USING LATENT SPACE REGRESSION TO ANALYZE AND LEVERAGE COMPOSITIONALITY IN GANS

Anonymous authors
Paper under double-blind review

ABSTRACT

In recent years, Generative Adversarial Networks have become ubiquitous in both research and public perception, but how GANs convert an unstructured latent code to coherent, high quality output is still an open question. In this work, we investigate regression into the latent space as a probe to understand the compositional properties of GANs. We find that combining the regressor and a pretrained generator provides a strong image prior, allowing us to create composite images from a collage of random image parts at inference time and leverage the generator to rectify the inconsistencies. To compare compositional properties across different generators, we measure the trade-offs between reconstruction of the unrealistic input and image quality of the regenerated samples. We find that the regression approach enables more localized editing of individual image parts compared to direct editing in the latent space, and we conduct experiments to quantify this independence effect. Our method is agnostic to how image parts are extracted, and does not require explicit labelled knowledge of concepts during training. Because this approach only uses a single feed-forward pass, it can operate in real-time and also perform a number of related applications, such as image inpainting or example-based image editing. We demonstrate its effectiveness on several GANs and datasets.

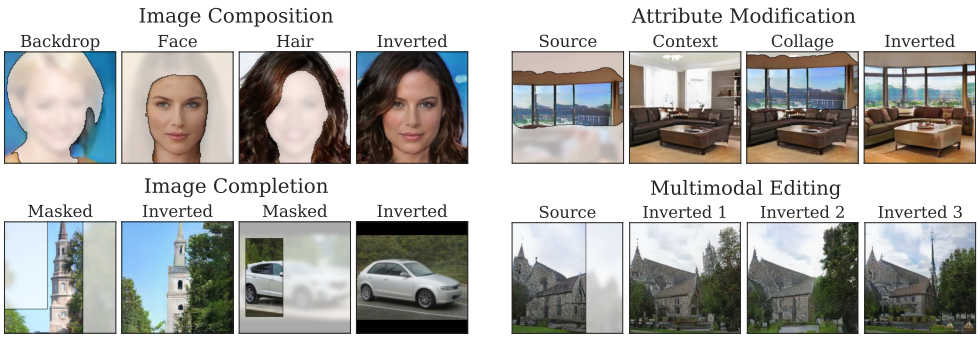


Figure 1: Simple latent regression on a fixed, pretrained generator can perform a number of image manipulation tasks without needing labelled concepts or changes to the generator. We use this to probe how GANs understand and compose scenes from image priors – using image patches and images with missing pixels as inputs.

1 INTRODUCTION

Unconditional generative adversarial networks are the current state-of-the-art in image generation (Karras et al., 2019b;a; 2017; Radford et al., 2015), but like any deep model, our understanding of this complex mapping from random noise to images remains limited. How do GANs create realistic scenes from seemingly uninterpretable noise? What kinds of representations do they learn during training? And how interchangeable are these learned concepts? Here, we use a latent space regressor model to probe what GANs know about the world. Given a pretrained, fixed generator network, we train a regressor to approximate the inverse mapping from image to noise sample. We then use this mapping to investigate the priors that a GAN enforces and the degrees of freedom that it allows.

For example, from a given church image, we can explore alternative ways that the generator can imagine constructing that scene. Can the generator maintain the same church but add clouds to the sky? What if we swap the foreground tree for a different one? Given only parts of the building, can it realistically fill in the missing portion? We show that in fact, the pretrained generator – without any additional interventions – already represents these compositional properties in its latent code. In addition, it applies certain rules that it learns from the dataset, ensuring that the output is always a coherent scene regardless of inconsistencies in the input.

Our approach is simple – using a fixed pretrained generator, we subsequently train a regressor to predict the latent code from an input image, while adding a masking modification to learn to handle missing pixels. To investigate the GAN's ability to produce alternative versions of a given scene, we input a rough template of the scene we desire, and use the regressor and generator networks to convert it into a realistic image. Even though our regressor is never trained on these unrealistic templates, the encoder bottleneck projects each image into a reasonable part of the latent space, which the generator maps onto the image manifold. The approach requires no labels or clustering of attributes; all we need is a single example of approximately how we want the generated image to look. It only requires a forward pass of the regressor and generator, so there is low latency in obtaining the output image, unlike iterative optimization approaches that can require upwards of a minute to reconstruct each image.

We use the regressor and generator to investigate the compositional capabilities of pretrained GANs across different dataset domains. Using automatically generated input collages, we leverage the generator to recombine this unrealistic content into a coherent image. We then investigate the GAN's ability to independently vary localized portions of a given image. In summary, our contributions are:

- We propose a latent regression model that learns to perform image reconstruction even in case of incomplete images and missing pixels.
- There is no use of labelled attributes nor test-time optimization, so we can modify images based on a single example of the desired modification and reconstruct in real-time. The reconstruction respects the style of the modified context, and not just the object identity.
- Together, the regressor and generator form a prior that can create coherent scene composites from rough collages of extracted image parts.
- We use the regressor to probe what parts of a scene can vary independently without explicit conceptual labels, and investigate the difference between image mixing using the encoder and interpolation in latent space.
- The same regressor setup can be used for a variety of other applications, such as multimodal editing, scene completion, or dataset rebalancing.

2 Related Work

Image Inversion. Given a target image, either the output of the generator network or a real image, the inversion problem aims to recover the latent code of the generator that best corresponds to the target. Image inversion comes with a number of challenges, including 1) the complex optimization landscape and 2) the generator's inability to reconstruct out-of-domain images. To relax the domain limitations of the generator, one possibility is to invert to a more flexible intermediate latent space (Abdal et al., 2019). However, inverting to an intermediate code may allow the generator to become overly flexible, thus requiring regularizers to ensure that the recovered latent code does not deviate too far from the latent manifold (Pividori et al., 2019; Zhu et al., 2020; Wulff & Torralba, 2020). An alternative to increasing the flexibility of the generator is to learn an ensemble of latent codes that approximate a target image when combined (Gu et al., 2019a). Due to the challenging optimization, the quality of inversion depends on good initialization. A number of approaches propose a hybrid of a latent regression network to provide an initial guess of the latent code with subsequent optimization of the latent code (Bau et al., 2019; Guan et al., 2020) or the generator weights (Zhu et al., 2016; Bau et al., 2020; Pan et al., 2020), while Huh et al. (2020) investigates gradient-free approaches for optimization. Another use case of image inversion through a generator may be to complete partial scenes: using optimization, this is achieved in optimization methods by reconstructing only the known pixels (Bora et al., 2017; Gu et al., 2019a; Abdal et al., 2020), whereas in feed-forward methods, this missing region must be provided explicitly to the model. Rather than inverting into pre-trained

generator, one can train the generator and encoder jointly; based on modifications to the Variational Autoencoder (Kingma & Welling, 2013); Donahue et al. (2016); Donahue & Simonyan (2019); Dumoulin et al. (2016) use this setup to investigate the properties of latent space representations learned during training, while Pidhorskyi et al. (2020) demonstrates a joint learning method that can achieve comparable image quality to recent GAN models. Specifically in our work, we investigate the emergent priors of a pretrained GAN using a masked latent regression objective as an approximate image inverter; the regression network allows us to investigate these GAN priors with a tradeoff of an more approximate reconstruction but much lower latency compared to optimization-based techniques.

Composition in Image Domains. To join segments of disparate image sources into one cohesive output, early works use hand-designed features, such as Laplacian pyramids for seamless blending (Burt & Adelson, 1983). To automate this, Isola & Liu (2013) and Hays & Efros (2007) employ nearest-neighbor approaches for scene composition and completion. More recently, a number of deep network architectures have been developed for compositional tasks. For discriminative tasks, Tabernik et al. (2016) and Kortylewski et al. (2020) train CNNs with modified compositional architecture for purposes of model interpretability and reasoning about object occlusion in classification. For image synthesis, Mokady et al. (2019) and Press et al. (2020) use an autoencoder to encode, disentangle, and swap properties between two sets of images, while Shocher et al. (2020) proposes mixing images in deep feature space while training the generator. Rather than creating models specifically for the image composition or scene completion objective, we investigate the ability of a pre-trained generator to mix-and-match parts of its generated images. Related to our work, Besserve et al. (2018) estimates the modular structure of GANs by learning a casual model of latent representations, whereas we investigate the generator's compositional properties using image inversion.

Image Editing. A recent topic of interest is the ability to edit images using generative models. A number of works propose linear attribute vector editing directions to perform image manipulation operations (Goetschalckx et al., 2019; Jahanian et al., 2019; Shen et al., 2019; Kingma & Dhariwal, 2018; Karras et al., 2019a; Radford et al., 2015). One may also identify concepts learned in the generator's intermediate layers by clustering intermediate representations, either using segmentation labels (Bau et al., 2018) or unsupervised clustering (Collins et al., 2020), and change these representations to edit the desired concepts in the output image. Suzuki et al. (2018) use a spatial feature blending approach which mixes properties of target images in the intermediate feature space of a generator. On faces, editing can be achieved using a 3D parametric model to supervise the modification (Tewari et al., 2017; 2020). In our work, we do not require clusters or concepts in intermediate layers to be defined a priori, nor do we need distinct input or output domains as in image translation tasks (Zhu et al., 2017; Almahairi et al., 2018). Furthermore, unlike image manipulation using semantic maps (Park et al., 2019; Gu et al., 2019b), our approach respects the style of the manipulation (e.g. the specific color of the sky), which is lost in the semantic map representation. Our method shares commonalities with Richardson et al. (2020), although we focus on investigating composition rather than image-to-image translation. In our approach, we only require a single example of the approximate target property we want to modify and use regression into the latent space as a fast image prior to create a coherent output. This allows us to create edits that are not contingent on learned concepts, and we do not need to modify or train the generator network.

3 METHOD

3.1 LATENT CODE RECOVERY IN GANS

While GANs provide a simple mapping from a predetermined input distribution to a complex output distribution, e.g. from a standard normal $\mathcal{Z}$ to the image manifold $\mathcal{X}$, they are not easily invertible. In other words, given an image sample from the output distribution, it is not trivial to recover the sample from the input distribution that generated it. The image inversion objective aims to find the latent code z that best recovers the desired target image x:

$$z^* = \arg\min_z (\mathrm{dist}(G(z),\ x)), \tag{1}$$

using some metric of image distance dist, such as pixel-wise L1 error or a metric based on deep features. This objective can be solved iteratively, using LBFGS (Liu & Nocedal, 1989) or other

optimizers. However, iteration incurs a substantial time cost – it takes a large number of iterations to converge, is prone to local minima, and must be performed for each target image x independently.

An alternative way of of recovering the latent code z is to train a neural network to amortize the optimization costs. In this case, the recovered latent code is simply the result of a feed-forward pass through a trained regressor network, $z^* = E(x)$, where E can be used for any $x \in \mathcal{X}$.

3.2 Learning a latent regressor function

Here, we investigate using latent regressors to probe properties of the generator latent space, such as the ability to combine scenes and complete missing regions. Our latent regression objective for the regressor network E is:

$$\mathcal{L} = \mathbb{E}_{z \sim N(0,1),\ x=G(z)} \left[||x - G(E(x))||_2^2 + \mathcal{L}_p(x, G(E(x))) + \mathcal{L}_z(z, E(x)) \right]. \tag{2}$$

We sample z randomly from the latent distribution, and pass it through a pretrained generator G to obtain the target image $x = G(z)$. Between the target image x and the recovered image $G(E(x))$, we use a mean square error loss to guide reconstruction, and a perceptual loss $\mathcal{L}_p$ (Zhang et al., 2018) to recover details. Between the original latent code z and the recovered latent code $E(x)$, we use a latent recovery loss $\mathcal{L}_z$. We use mean square error or a variant of cosine similarity for latent recovery, depending on the GAN's input normalization. We provide additional details in Supp. Sec. A.1.1.

In all cases the generator is frozen, and we only optimize the weights of the latent space encoder E. Using ProGAN (Karras et al., 2017), we train the encoder network to directly invert to the latent code z. For StyleGAN (Karras et al., 2019b), we encode to an expanded $\mathcal{W}^+$ latent space. Once trained, the output of the latent regressor yields a latent code such that the reconstructed image looks perceptually similar to the target image and is much faster than iterative optimization methods.

3.3 Image composition using latent regression

The regressor network E and generator G form an image prior to project any input image x_{input} onto the manifold of generated images $\mathcal{X}$, even if $x_{\text{input}} \notin \mathcal{X}$. We take advantage of this to investigate the compositional properties of the latent code. We extract parts of images (either generated by G or from real images), and combine them to form a collaged image x_{clg}. This extraction process does not need to be precise and can have obvious seams, so x_{clg} is often not realistic. We then use the regressor/generator pair to blend the seams and produce a realistic composite output. To create x_{clg}, we sample base images x_i and masks mask_i, and combine them via union:

$$x_{\text{clg}} = \bigcup_i \text{mask}_i \otimes x_i. \tag{3}$$

To automate the process of extracting masked images, we use a pretrained segmentation network (Xiao et al., 2018) and sample from the output classes (additional details in Supp. Sec. A.1.2). However, the masking network is not specific to segmentation – we also experiment with a saliency network (Liu et al., 2018) as well as user-defined masks in Supp. Sec. A.2.4. Once we have formed the collage x_{clg}, we reproject via the regressor generator to obtain the recovered composite image $x_{\text{rec}} = G(E(x_{\text{clg}}))$. Note that the encoder is trained solely for the latent recovery objective (Eqn. 2) and has never seen collaged images.

3.4 Learning with missing data

When automatically creating composite images, not every pixel in x_{comp} will be filled with part of a base image x_i. In optimization approaches using Eqn. 1, this can be handled by optimizing only over the known pixels. However, a regressor network cannot handle this naively - it cannot distinguish between unknown pixels and known pixels, and will try to fit the values of the unknown pixels. This can be mitigated with a slight modification to the regression network, by indicating which pixels are known versus unknown as input (Fig. 2), yielding a modified regression objective:

$$\mathcal{L} = \mathbb{E}_{z \sim N(0,1),\ x=G(z)} ||x - G(E(x_m, m))||_2^2 + \mathcal{L}_p(x, G(E(x_m, m))) + \mathcal{L}_z(z, E(x_m, m)) \tag{4}$$

Instead of taking an image x as input, the encoder takes as input a masked image x_m and a mask m, where $x_m = x \otimes m$, and m is an additional channel of input. Again, we hold the generator

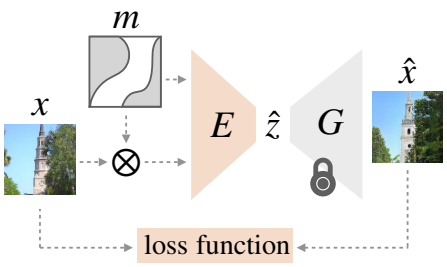


Figure 2: We train a latent space regressor E to predict the latent code $\hat{z}$ that, when passed through a fixed generator, reconstructs input x. At training and test time, we can also modify the encoder input with additional binary mask m (Eqn. 4). Inference is only a forward pass of the networks, and the input x can be realistic, as the encoder and generator serve as a prior to map the image back to the image manifold.

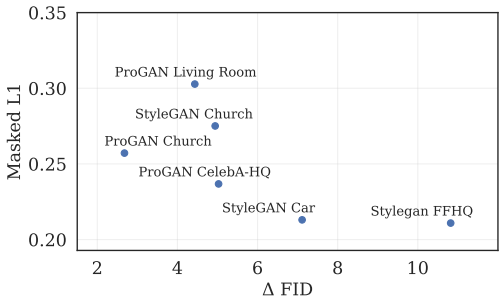


Figure 3: Composition of unrealistic input collages is a balance of two factors: we want to reconstruct the input (low L1), but still retain a realistic image (low FID). Using automatic collages of synthesized image parts, we plot this tradeoff of Masked L1 error for reconstruction and difference FID score between the unrealistic composites and re-encoded images on different image domains.

fixed and only optimize the weights of the encoder using image samples from the generator. Similar to before, we train only for the latent recovery objective and not for the image composition task. Intuitively, this masking operation on the encoder operates similarly to "dropout" Srivastava et al. (2014) on pixels – it encourages the encoder to learn a flexible way of recovering a latent code that still enables the generator to reconstruct the image. Thus, given only partial images as input, the encoder is encouraged to map from these known pixels into a part of the latent space that is semantically consistent with the rest of the image (as deeper layers of the generator focus on semantic concepts (Bau et al., 2018)). This allows the generator to reproduce an image that is both likely under its prior and consistent with the observed pixels.

To obtain the masked image during training we take a small patch of random uniform noise u, upsample to create smooth boundaries, and sample a threshold t from the uniform distribution to create the binary mask. However, at test time, the exact form of the mask does not matter – the mask simply indicates where the generator should try to reconstruct or inpaint, and does not distinguish between the different image parts of the input (i.e., the masks used to extract image parts in Eqn. 3 are not available to the encoder). We provide additional details in Supp. Sec. A.1.1 and A.2.3.

4 Experiments

Using pre-trained Progressive GAN (Karras et al., 2017) and StyleGAN2 (Karras et al., 2019b) generators, we conduct experiments on CelebA-HQ and FFHQ faces and LSUN cars, churches, living rooms, and horses to investigate the compositional elements that the generators learn from data.

4.1 Image Composition from Approximate Collages

By training a latent regressor with a masked objective, we find that the encoder and generator pair forms an image prior which can convert unrealistic input images into realistic scenes that maintain properties of the input image. By mixing parts of different images together and encoding through the generator, we obtain a coherent output that is consistent with each part of the input while filling in the missing regions. How well can the generator recombine concepts that it creates, and how does this ability vary across different generator architectures and dataset domains? In this experiment, we automate the process of creating collages from image parts by randomly sampling the generated images, extract different semantic categories using a pretrained segmentation network, and overlaying these patches using Eqn. 3. In Fig. 4, we show qualitative examples of these input collages and the corresponding generator output across a variety of domains; note that the input collages are not realistic, and often contain missing detections or misalignments. However, the learned image prior from the generator and encoder bottleneck fixes these issues to create realistic outputs.

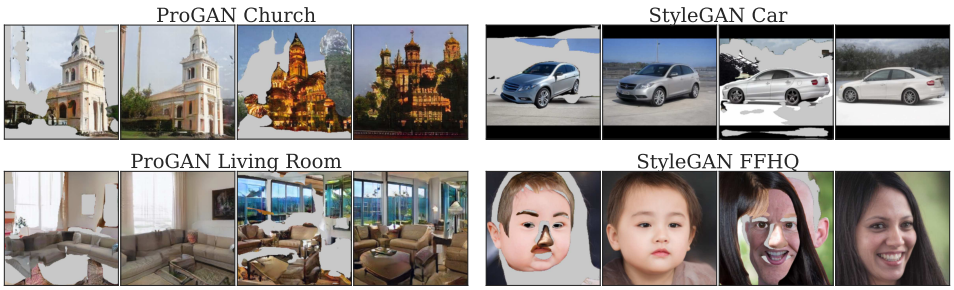


Figure 4: Trained only on a masked reconstruction objective, a regressor into a the latent space of a pretrained generator allows the generator to recombine components of its generated images, despite strong misalignments and and missing regions in the input. Here, we show automatically generated collaged input from extracted image parts, and the corresponding output of the generator.

When creating composites from random images, composition is a tradeoff between two factors. First, we desire the networks to reconstruct the input, which are recombinations of images that each generator creates. However, due to the automatic nature of the collages, a perfect reconstruction will not be realistic due to the lack of alignment between image parts, so the second criteria is that the reconstructed images should have good quality. In Fig. 3, we visualize the tradeoff between these two desirable properties of composition by measuring how well each generator can reconstruct randomly mixed parts of images from its own domain using masked L1 distance (lower is better):

$$\text{avg}(m \otimes |x - G(E(x, m))|) \tag{5}$$

and the image quality of the synthesized result (FID score; Heusel et al. (2017). To isolate the realism of the composite image from the encoder's reconstruction capability, we compare the difference FID between the reconstructed composites $G(E(x))$, where $x = \cup_i \text{mask}_i \otimes G(z_i)$, and re-encoded images $G(E(G(z))$. In general, encoding to the StyleGAN $\mathcal{W}^+$ latent space results in better reconstructions of the collaged input, but at the tradeoff of higher FID, potentially due to the increased flexibility of the expanded $\mathcal{W}^+$ latent code compared to the more restricted z latent code in ProGAN. However, while the StyleGAN generator on FFHQ faces has the largest increase in FID, we note that the synthesized composites have high density but lower coverage (Naeem et al., 2020), indicating that while the generated composites are close to the real-image manifold, they are missing full variation of natural images. We show full results on FID, density, coverage, and L1 reconstruction error and additional random samples of input collages and generator output in Supp. Sec. A.2.4.

While here we use a pretrained segmentation network to automatically extract parts, our reconstruction method is agnostic to the way in which the parts are extracted. In Supplementary Sec. A.2.1 and A.2.4, we show that these image collages can also be created by extracting output from a saliency network, approximate rectangles, or user-drawn masks.

4.2 Comparing Compositional Properties Across Architectures

While using a trained encoder network gives us a fast and simple way of creating a composite image from an input collage, it is not the only way of finding images, in our case the collaged input, within a pretrained generator network. Is it only pretrained generators that have this property, and is it contingent on having a feed-forward encoder? Here, we compare the the different image priors enforced by different image reconstruction methods, using both pretrained generators as well as separately trained generator architectures, where inference involves either optimization or a feed-forward pass. For comparison, we use the same set of collages to compare the methods, generated from parts of random real images of the church and face domains. As some methods take several minutes to reconstruct a single image, we use 200 collages for each domain (see Tab. 4 for optimization times of each). Due to the smaller sample size, we use density as a measure of realism (higher is better), which measures proximity to the real-image manifold (Naeem et al., 2020).

For the church domain, we compare to methods in which the generator and encoder are trained jointly: DIP (Ulyanov et al., 2018), CycleGAN (Zhu et al., 2017), and SPADE (Park et al., 2019) (additional details in Supplementary Sec. A.2.4). Next, we experiment with different ways of inverting into the latent space of pretrained GAN networks: (1) our method of using a trained encoder network, (2)

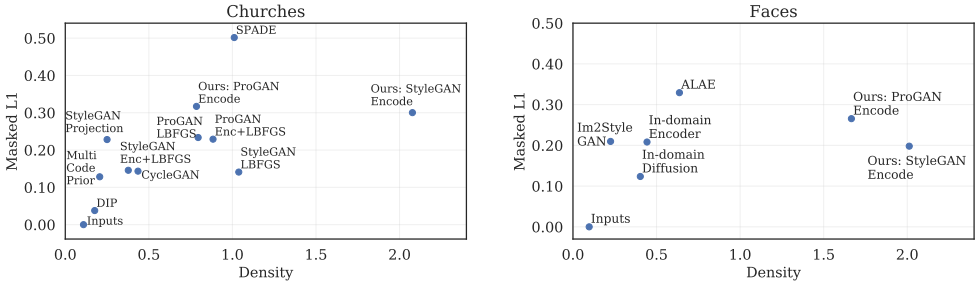


Figure 5: Comparing reconstruction of an image collage (masked L1) to realism of the generated output on random church collages (left) and face collages (right) across different image reconstruction methods – these methods include both feed-forward and iterative optimization approaches, as well as architectures in which a generator and encoder is trained jointly rather than using a pretrained generator. An ideal composition has low L1, but high density (close to real image manifold).

using LBFGS optimization into a pretrained generator, (3) or a hybrid of the encoder and LBFGS. We also evaluate the compositional properties of two additional iterative optimization methods: Gu et al. (2019a) which optimizes for multiple latent codes at an intermediate layer of the GAN, and a perceptual-loss based projection method demonstrated in Karras et al. (2019b). Qualitatively, the different methods have varying degrees of realism when trying to reconstruct unrealistic input collages (we show examples in Supp. Fig. 17); optimization-based methods such as DIP tend to overfit and lead to unrealistic reconstructions with lower L1 error but also low density (unrealistic), whereas segmentation-based methods such as SPADE are not trained to reconstruct the input, leading to high L1. Fig 5 illustrates this compositional tradeoff, where the encoder based methods perform slightly worse in L1 reconstruction compared to optimization approaches, but maintain more realistic output and can reconstruct with much lower computational time.

On the face domain, we compare composite outputs generated from ProGAN and StyleGAN using the regressor network to the ALAE architecture (Pidhorskyi et al., 2020), which jointly learns the encoder and decoder, and the Im2StyleGAN (Abdal et al., 2019) algorithm which does not use an encoder but directly optimizes the latent code. We also compare to In-domain inversion (Zhu et al., 2020), which uses an unmasked encoder to initialize and regularize a secondary masked optimization step. We show qualitative examples in Supplementary Fig. 18 and a comparison of reconstruction L1 and density in Fig. 5: the our ProGAN and StyleGAN masked encoders can maintain closer proximity to the real image manifold (higher density) compared to the alternative methods, with much faster inference time compared to Im2StyleGAN. While the In-domain encoder is also a feed-forward method, it cannot handle incomplete inputs, so both the encoder and subsequent optimization step have low density. On these same inputs, ALAE exhibits interesting compositional properties and is qualitatively able to correct misalignments in face patches, but the density of generated images lower than that of the pretrained GANs.

4.3 How does composition differ from interpolation?

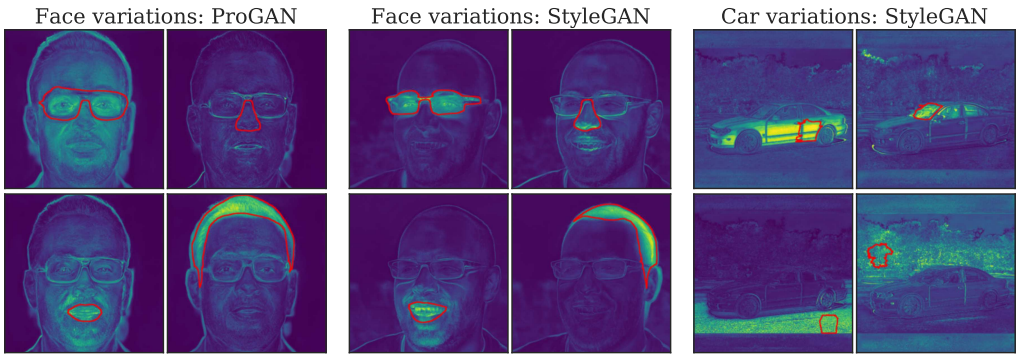
Combining images in pixel space and using the encoder bottleneck to rectify the input is only one way that a generator can mix different images. Another commonly used method is to simply use a linear interpolation between two latent codes to obtain an output that has properties of both input images. Here, we investigate how these two approaches differ. When collaging parts of two images, we desire that part of a context image stays the same while the remaining portion changes to match a chosen target: to achieve this *composition*, we replace the pixels of our context image with the desired modified target, and pass the result through the encoder bottleneck to obtain the blended image. This can be described using: $G(E(m \otimes x_1 + (1-m) \otimes x_2))$. Does directly interpolating in latent space give us the same result? To compare, we compute the *latent* α-blending by performing a weighted average of the context and target latent codes: $G(\alpha * E(x_1) + (1-\alpha) * E(x_2))$. Third, we also compare to *pixel* α-blending the inputs in pixel space and using the encoder bottleneck to make the output more realistic: $G(E(\alpha * x_1 + (1-\alpha) * x_2))$. We select the weight α to be proportional to the area of the target modification.

Figure 6: Comparing properties of composition and interpolation. We aim to apply the same target modification (white windows; Target), to two context sources (Context), where the collage of the two images is shown in the third column (Collage). Inverting a collaged image into latent space via the encoder (Composition) yields a more disentangled interpolation that blends the inconsistency between the two target images. Compared to alternative Latent and Pixel α-blending, the Composition approach better matches the context and target regions.

Qualitatively, the disentangled interpolation is better able to change the target region while keeping the context region fixed, e.g., add white windows while reducing changes in the fireplace or couch in Fig. 6, whereas the other two interpolation methods are less faithful in preserving image content. To quantify this effect, we compute the masked L1 distance (Eqn. 5) of the interpolated images to (1) the original masked context region, (2) to the original masked target region, and (3) to the input collage over 500 random samples. The disentangled interpolation has lower distance from the target and context, and is also closer to the desired collage. Unlike interpolations using attribute vectors, these distentangled directions do not need to be learned from data - they are based on a single context and target image, and also allow multiple possible directions of manipulation. We show additional interpolations and comparison to learned attribute vectors in Supp. Sec. A.2.2, and additional applications such as dataset rebalancing and one-shot image editing in Supp. Sec. A.2.1.

4.4 Using regression to investigate independence of image components

One property of the regressor and generator pair is that it enforces global coherence: When we modify part of the input, the generator will create an output that is still globally consistent. While global consistency is desirable, we also want to minimize leakage of the edits. For example, if we change a facial expression from a frown to a smile, the hair style should not change. Here, we seek to quantify the independence of parts under the global coherence constraint imposed by the regressor / generator pair. We parse a given face image x into separate semantic components (such as eyes, nose, etc.) represented as masks m_c. For each component, we generate N new faces x_n, and replace the component region m_c in x by the corresponding region in x_n, yielding (after regression and generation) for each component c a set of N images $x_{c,n} = G(E(m_c \otimes$



(a) Face variation, ProGAN (b) Face variation, StyleGAN (c) Part discovery for cars

Figure 7: Image variation when replacing single parts. In ProGAN (a), replacing single parts leads to change beyond the part that is being changed. Changes in StyleGAN (b) are more localized. This method can be used to find correlated regions even without semantic labels, shown for cars (c).

$x_n + (1 - m_c) \otimes x))$. We can now measure how much changing component c changes each pixel location by computing the normalized pixel-wise standard deviation of $x_{c,n}$ across the n different replacements as $v_c = \sigma_c / \sum_c \sigma_c$, where $\sigma_c = \sqrt{\mathbb{E}_n[(x_{n,c} - \mathbb{E}_n[x_{n,c}])^2]}$. For a given component c, we measure independence as the average variation outside of c that results from modifying c as $s_c = \mathbb{E}[(1 - m_c) \otimes v_c]$ (a lower s_c means higher independence). We repeat this experiment 100 times and use $N = 20$ samples.

Table 1: Face part independence across models (lower means more independent), measured as the variation outside of the replaced part, computed over difference replacements.

	Background	Skin	Eyes	Ears	Nose	Mouth	Hair	Avgerage
ProGAN	0.167	0.177	0.043	0.024	0.030	0.038	0.170	0.093
StyleGAN	0.129	0.163	0.034	0.020	0.038	0.025	0.119	0.075

Table 1 shows the variations of ProGAN and StyleGAN. With the exception of the nose, StyleGAN achieves better independence, indicating increased part localization and less leakage of edits. Figure 7(a) and (b) qualitatively show examples for the variation maps v_c for different parts for ProGAN (a) and StyleGAN (b); the replaced part is marked in red. It can be seen that in StyleGAN the variation is more concentrated in the region that is being replaced. Lastly, this method can be utilized for unsupervised part discovery, as shown in Fig. 7(c). Here, changing the color of the rear door (top left) changes the appearance of the whole car body; changing the windshield appearance (top right) also changes the windows, and the foreground (bottom left) and background (bottom right) are large parts varying together, but distinct from the car.

5 Conclusion

Using a simple latent space regression model, we investigate the compositional properties of pretrained GAN generators. We train a regressor model to predict the latent code given an input image with the additional objective of learning to complete a scene with missing pixels. With this regressor, we can probe various properties and biases that the GAN learns from data. In creating scenes, the GAN allows for local degrees of freedom but maintains an overall degree of global consistency. This allows us to input approximate templates to the regressor model, such as partially occluded scenes or collages extracted from parts of images, and use the regressor and generator as image prior to regenerate a realistic output. Inverting the collaged images yields disentangled interpolation directions where part of the input changes while the remaining context remains fixed. The latent regression approach allows us to investigate how the GAN enforces independence of image parts, and is learned without knowledge of objects or explicit attribute labels. It only requires a single forward pass on the models, which enables real time image editing based on single image examples.

References

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4432–4441, 2019.

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8296–8305, 2020.

Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. *arXiv preprint arXiv:1802.10151*, 2018.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4502–4511, 2019.

David Bau, Hendrik Strobelt, William Peebles, Bolei Zhou, Jun-Yan Zhu, Antonio Torralba, et al. Semantic photo manipulation with a generative image prior. *arXiv preprint arXiv:2005.07727*, 2020.

Michel Besserve, Arash Mehrjou, Rémy Sun, and Bernhard Schölkopf. Counterfactuals uncover the modular structure of deep generative models. *arXiv preprint arXiv:1812.03253*, 2018.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. *arXiv preprint arXiv:1703.03208*, 2017.

Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

Edo Collins, Raja Bala, Bob Price, and Sabine Süsstrunk. Editing in style: Uncovering the local semantics of gans. *arXiv preprint arXiv:2004.14367*, 2020.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pp. 10542–10552, 2019.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5744–5753, 2019.

Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. *arXiv preprint arXiv:1912.07116*, 2019a.

Shuyang Gu, Jianmin Bao, Hao Yang, Dong Chen, Fang Wen, and Lu Yuan. Mask-guided portrait editing with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3436–3445, 2019b.

Shanyan Guan, Ying Tai, Bingbing Ni, Feida Zhu, Feiyue Huang, and Xiaokang Yang. Collaborative learning for faster stylegan embedding. *arXiv preprint arXiv:2007.01758*, 2020.

James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4–es, 2007.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pp. 6626–6637, 2017.

Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. *arXiv preprint arXiv:2005.01703*, 2020.

Phillip Isola and Ce Liu. Scene collaging: Analysis and synthesis of natural images with semantic layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3048–3055, 2013.

Ali Jahanian, Lucy Chai, and Phillip Isola. On the"steerability" of generative adversarial networks. *arXiv preprint arXiv:1907.07171*, 2019.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019a.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*, 2019b.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Adam Kortylewski, Ju He, Qing Liu, and Alan L Yuille. Compositional convolutional neural networks: A deep architecture with innate robustness to partial occlusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8940–8949, 2020.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3089–3098, 2018.

Ron Mokady, Sagie Benaim, Lior Wolf, and Amit Bermano. Mask based unsupervised content transfer. *arXiv preprint arXiv:1906.06558*, 2019.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. *arXiv preprint arXiv:2002.09797*, 2020.

Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. *arXiv preprint arXiv:2003.13659*, 2020.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2337–2346, 2019.

Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14104–14113, 2020.

Marcos Pividori, Guillermo L Grinblat, and Lucas C Uzal. Exploiting gan internal capacity for high-quality reconstruction of natural images. *arXiv preprint arXiv:1911.05630*, 2019.

Ori Press, Tomer Galanti, Sagie Benaim, and Lior Wolf. Emerging disentanglement in auto-encoder based unsupervised image content transfer. *arXiv preprint arXiv:2001.05017*, 2020.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. *arXiv preprint arXiv:1907.10786*, 2019.

Assaf Shocher, Yossi Gandelsman, Inbar Mosseri, Michal Yarom, Michal Irani, William T Freeman, and Tali Dekel. Semantic pyramid for image generation. *arXiv preprint arXiv:2003.06221*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Ryohei Suzuki, Masanori Koyama, Takeru Miyato, Taizan Yonetsuji, and Huachun Zhu. Spatially controllable image synthesis with internal representation collaging. *arXiv preprint arXiv:1811.10153*, 2018.

Domen Tabernik, Matej Kristan, Jeremy L Wyatt, and Aleš Leonardis. Towards deep compositional networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 3470–3475. IEEE, 2016.

Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1274–1283, 2017.

Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6142–6151, 2020.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9446–9454, 2018.

Jonas Wulff and Antonio Torralba. Improving inversion and generation diversity in stylegan using a gaussianized latent space. *arXiv preprint arXiv:2009.06529*, 2020.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 418–434, 2018.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. *arXiv preprint arXiv:2004.00049*, 2020.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pp. 597–613. Springer, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A APPENDIX

A.1 SUPPLEMENTARY METHODS

A.1.1 ADDITIONAL TRAINING DETAILS

The loss function of the encoder contains image loss terms to ensure that the output of the generator approximates the target image, and a latent recovery loss term to ensure that the predicted latent code matches the original latent code. On the image side, we use mean square error loss in conjunction with LPIPS perceptual loss (Zhang et al., 2018). The latent recovery loss depends on the type of GAN. Due to pixel normalization in ProGAN, we use a latent recovery loss based on cosine similarity, as the exact magnitude of the recovered latent code does not matter after normalization:

$$L_z = 1 - \frac{z}{||z||_2} \cdot \frac{E(x)}{||E(x)||_2}. \tag{6}$$

For StyleGAN, we invert to the intermediate latent space, as it is known that in this space semantic properties are better disentangled than in $\mathcal{Z}$. Furthermore, allowing the latents to differ on different scales has been shown to better capture the variability of real images (Abdal et al., 2019). During training, we therefore generate different latents for different scales, and train the encoder to estimate different styles, i.e. estimate $w \in \mathcal{W}^+$. Unlike the latent space of ProGAN, however, $w \in \mathcal{W}^+$ is not normalized to the hypersphere. Instead of a cosine loss, we therefore use a mean square error loss as the latent recovery loss:

$$L_w = ||w - E(x)||_2. \tag{7}$$

We train the encoders using a ResNet backbone (ResNet-18 for ProGAN, and ResNet-34 for Stylegan), modifying the output dimensionality to match the number of latent dimensions for each GAN. The encoders are trained with the Adam optimizer with default parameters, learning rate $lr = 0.001$. We use batch size 16 for the 256 resolution generators, and train for 500K batches (training takes around two days on a single GPU). For the 1024 resolution generator, we use batch size 4 and 400K batches.

When training with masks, we take a small 6x6 patch of random uniform noise u, upsample to the generator's resolution, and sample a threshold t from the uniform distribution in range $[0.3, 1.0]$ to create the binary mask.

$$\begin{aligned} m &= \mathbb{1}\left[\text{Upsample}(u) > t\right] \\ x_m &= m \odot x \end{aligned} \tag{8}$$

At inference time, the exact shape of the mask does not matter: we can use hard coded rectangles, hand-drawn masks, or masks based on other pretrained networks. Note that the mask does not distinguish between input image parts – it is a binary mask with value 1 where the generator should try to reconstruct, and 0 where the generator should fill in the missing region.

A.1.2 ADDITIONAL DETAILS ON COMPOSITION

When creating automated collages from image parts, we use a pretrained segmentation network (Xiao et al., 2018) to extract parts from randomly sampled individual images. We manually define a set of segmentation class for a given image class, and, to handle overlap between parts, specify an order to these segmentation classes. To generate a collage, we then sample one random image per class. For church scenes, we use an ordering of (from back to front) sky, building, tree, and foreground layers – this ensures that a randomly sampled building patch will appear in front of the randomly sampled sky patch. For living rooms, the ordering we use is floor, ceiling, wall, painting, window, fireplace, sofa, and coffee table – again ensuring that the more foreground elements are layered on top. For cars, we use sky, building, tree, foreground, and car. For faces we order by background, skin, eye, mouth, nose, and hair. In Sec. A.2.4 we investigate using other methods to extract patches from images, rather than image parts derived from segmentation classes.

A.2 SUPPLEMENTARY RESULTS

A.2.1 ADDITIONAL APPLICATIONS

In the main text, we primarily focus on using the regression network as a tool to understand how the generator composes scenes from missing regions and disparate image parts. However, because the regressor allows for fast inference, it enables a number of other real-time image synthesis applications.

Image Completion From Masked Inputs. Using the masked latent space regressor in Eqn. 4, we investigate the GAN's ability to automatically fill in unknown portions of a scene given incomplete context. We show qualitative examples of this property in Fig. 8. These reconstructions are done on parts of real images to ensure that the regressor is not simply memorizing the input, as could be the case with a generated input (the regressor is never trained on real images). For example, when a headless horse is shown to the regressor, the rest of the horse can be filled in by the GAN. In contrast, a vanilla latent regressor (Eqn. 2; RGB) overfits on the missing pixels and is unable to produce a realistic output. The image completion also depends on the exposed context; in the living room example, the white couch is completed in both cases but the window is only added based on known context, whereas the background is different when the window is not shown. This suggests that the networks inherently learn an unsupervised object representation, allowing them to complete objects from only partial hints, even though the generator and regressor are never provided with structured labels or concepts during training.

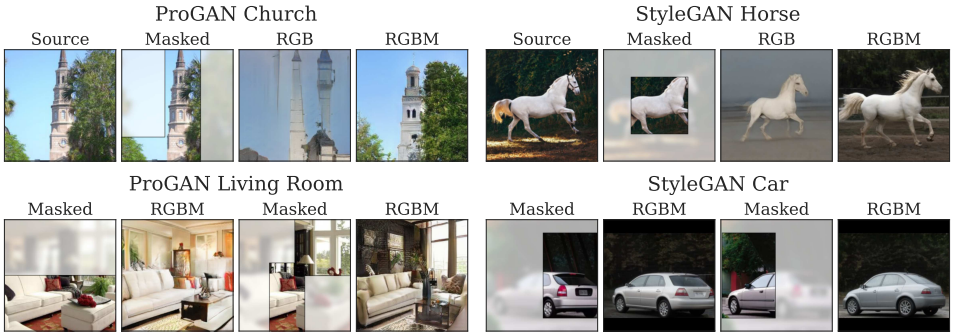


Figure 8: Image completions using the latent space regressor. Given a masked real image, a regressor without knowledge of masked images is unable to realistically reconstruct the scene, while the regressor trained on masks inputs inpaints in the unknown region in a way that is consistent with the given context (top). The completion can vary depending on the exposed context region of the same input (bottom).

Multimodal Editing. Because the regressor only requires a single example of the property we want to reconstruct, it is possible to achieve multimodal editings simply by overlaying different desired properties on a given context image. Here, we demonstrate an example of adding different styles of trees to a church scene. In each of the context images, there is originally no tree on the right-hand side. We can add different styles of trees to each context image simply by overlaying a tree from a different image, and performing image inversion to create the composite. Here, we use a rectangular region as a mask, rather than following the boundary of the tree precisely. However, note that, after inversion, the color of the sky remains consistent in the composite image, and so does the building color in second tree example. This image editing approach does not required learning of clusters or having labelled attributes, and therefore can be done with a single pair of images without known cluster definitions, unlike the methods in Bau et al. (2018); Collins et al. (2020). Furthermore, unlike methods based on segmentation maps (Park et al., 2019; Gu et al., 2019b), styles within each individual semantic category, e.g. the color of the sky, is also changeable based on the provided input to the encoder.

Attribute Editing With A Single Unlabeled Example. Typically in attribute editing using generative models, one requires a collection of labeled examples of the attribute to modify; for example, we take a collection of generated images with and without the attribute based on an attribute classifier, and find the average difference in their latent codes (Jahanian et al., 2019; Radford et al., 2015;

Figure 9: Using latent regression, we can perform multimodal editing to a context image. Each editing operation uses a single pair of images. Here, our editing mask is a rough rectangular box. Note how in case of *Tree 2*, the pasted region includes part of the church; the regressor attempts to include this part, and the resulting re-generated building therefore looks different from the building in the other examples.

Goetschalckx et al., 2019). Here, we demonstrate an approach towards attribute editing without using attribute labels (Fig. 10). We overlay a single example of our target attribute (e.g. a smiling mouth or a round church tower), on top of the context image we want to modify, and encode it to get the corresponding latent code. We then subtract the difference between the original and modified latent codes, and add it to the latent code of the secondary context image: $z_{1,\text{modified}} - z_1 + z_2$. This bypasses the need for an attribute classifier, such as one that identifies round or square church towers.

Figure 10: We can use latent regression to perform multimodal editing to a context image. From this single pair of images, we can compute a manipulation vector and apply it to a secondary image, which edits the secondary image accordingly.

Dataset Rebalancing. Because the latent regression inverter only requires a forward pass through the network, we can quickly generate a large number of reconstructions. Here, we use this property to investigate rebalancing a dataset (Fig. 11). Using pretrained attribute detectors from Karras et al. (2019a), we find that there is a smaller fraction of smiling males than smiling females in CelebA-HQ (first panel), although the detector itself is biased and tends to overestimate smiling in general compared to ground truth labels (second panel). The detections in the GAN generated images mimic this imbalance (third panel). Next, using a fixed set of generated images, we randomly swap the mouth regions among them in accordance to our desired proportion of smiling rates – we give each

face a batch of 16 possible swaps and taking the one that yields the strongest detection as smiling/not smiling based on our desired characteristic (if no swaps are successful, the face is just used as is). We use a hardcoded rectangular box region around the mouth, and encode the image through the generator to blend the modified mouth to the face. After performing swapping on generated images, this allows us rebalance the smiling rates, better equalizing smiling among males and females (fourth panel). Finally, we use the latent regression to perform mouth swapping on the dataset, and train a secondary GAN on this modified dataset – this also improves the smiling imbalance, although the effect is stronger in males than females (fifth panel). We note that a limitation of this method is that the rebalanced proportion uses the attribute detector as supervision, and therefore a biased or inaccurate attribute detector will still result in biases in the rebalanced dataset.

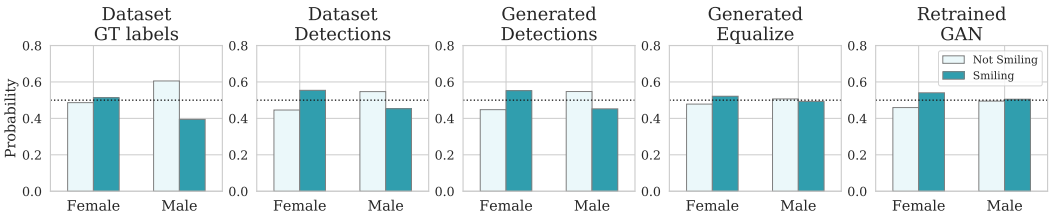


Figure 11: We perform dataset rebalancing using latent space regression. From pretrained attribute detectors, a smaller fraction of males smile than females in CelebA-HQ, and the GAN samples mimic this trend. By swapping mouths in the GAN samples, we can equalize male and female smiling rates. Next, we do this swap and invert operation on the dataset and retrain a GAN on the modified dataset, which also improves balance in smiling rates although the effect is stronger for the male category.

A.2.2 Comparing composition with latent space interpolation

In the main text, we compare different interpolation examples – disentangled, latent, and pixel interpolations – on a living room generator. Here, we show equivalent examples in church scenes. In Fig. 12, we demonstrate a "tree" interpolation, in which we want the background church to remain the same but trees to be added in the foreground. Compared to the latent and pixel interpolations, the disentangled interpolation better preserves the church while adding trees, which we quantify using masked perceptual distance. Similarly, we can change the sky of context scene - e.g. by turning a clear sky into a cloudy one in Fig. 13, where again, doing the disentangled interpolation better preserves the details of the details of the church while the sky changes.

In Fig 14, we show the result on changing the smile on a generated face image. For the smile attribute, we can compare to a learned smile edit vector using labelled images from a pretrained smile classifier. We also measure the the facial embedding distance using a pretrained face-identification network[1] – the goal of the interpolation is to change the smile of the image while reducing changes to the rest of the face identify, and thus minimize embedding distance. Because the mouth region is small, choosing the interpolation weight α by the target area minimally changes the interpolated image ($\alpha > 0.99$), so instead we use an $\alpha = 0.7$ weight so that all methods have similar distance to the target. While the disentangled approach and the learned attribute vector approach perform similarly, learning the attribute vector requires labelled examples, and cannot perform multimodal modifications such as applying different smiles to a given face.

[1] https://github.com/timesler/facenet-pytorch

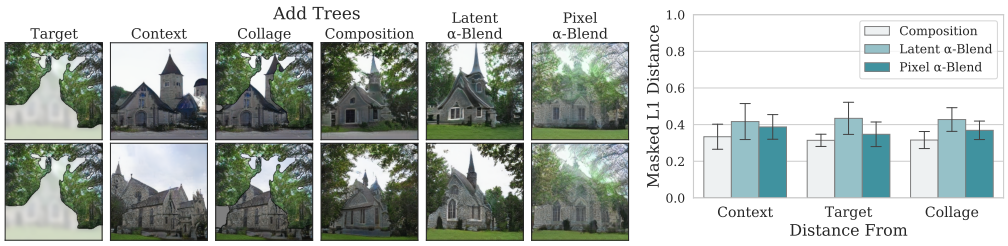


Figure 12: Comparison of disentangled, latent, and pixel interpolations when adding trees (Target) to different scenes (Context). The overlay of these two components is shown in Collage, where we use a mask to indicate missing regions. We compare (1) a Composition approach using the encoder and generator to blend the images together to (2) interpolation in the latent codes (Latent α-blend) and (3) interpolation in pixel space, which then passes through the encoder to pull the image closer to the image manifold. We find that the composition approach is better able to match properties of both the context and target images compared to the two alpha-blending techniques.

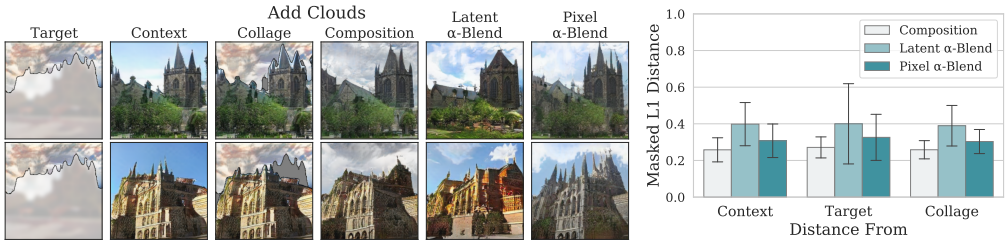


Figure 13: Comparison of disentangled, latent, and pixel interpolations when changing the sky of a scene, using similar encodedr-based Composition, Latent α-blend, and Pixel α-blend methods as above.

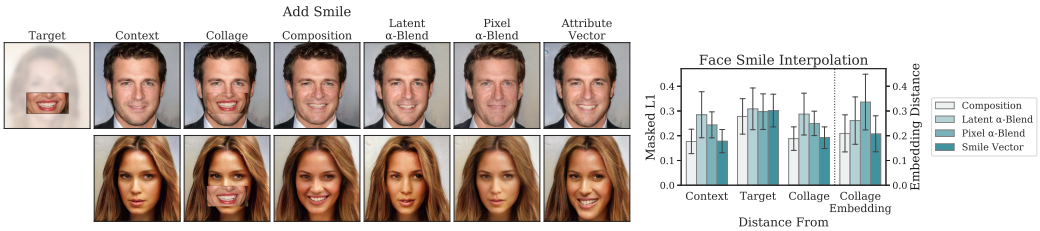


Figure 14: Comparison of disentangled, latent, and pixel interpolations using the same encoder-based Composition, Latent α-blend, and Pixel α-blend methods as above when adding a smile to a face. We also compare these approaches to the modifications produced by edit vector learning from attribute labels.

A.2.3 Loss Ablations

When training the regression network, we use a combination of three losses: L2 pixel loss, perceptual loss, and a latent recovery loss. In this section, we investigate the reconstruction result using different combinations of losses. In the first case, we do not enforce latent recovery, so the encoded latent code does not have to be close to the ground truth latent code but the encoder and generator just need to reconstruct the masked input image. In the second case, we investigate omitting the perceptual loss, and simple use an L2 loss in image space. Since the encoders are trained with L2 loss and the VGG variant of perceptual loss, we evaluate with L1 reconstruction and the AlexNet variant of perceptual loss. Training with all losses leads to reconstructions that are more perceptually similar compared to the two other variants, while per-pixel L1 reconstruction is not greatly affected (Tab. 2). We show qualitative examples of the reconstructions on masked input in Fig. 15.

Table 2: Table of masked L1 and LPIPS-Alexnet reconstruction errors for encoder networks trained with all losses, no latent loss, and no perceptual loss.

	All Losses	**No Latent**	**No LPIPS**
L1	0.194	0.194	0.197
LPIPS(alex)	0.291	0.305	0.318

Figure 15: Qualitative examples of reconstructions from masked inputs on encoders trained with different combinations of loss terms.

A.2.4 Additional Composition Results

Comparing different composition approaches. In Fig 16, we show examples of extracted image parts, the collage formed by overlaying the image parts, and the result of poisson blending the images according to the outline of each extracted part. We further compare to variations of the encoder setup, where (1) RGB: the encoder is not aware of missing pixels in the collaged input, (2) RGB Fill: we fill the missing pixels with the average color of the background layer, and (3) RGBM: we allow the encoder and generator to inpaint on missing regions. Table 3 shows image quality metrics of FID (lower is better) and density and coverage (higher is better) and L1 reconstruction (lower is better) for each generator and domain. To get feature representations for the density and coverage metrics, we use pretrained VGG features. While the composite input collages are highly unrealistic with high FID and low density/coverage, the inverted images are closer to the image manifold. Of the three inversion methods, the RGBM inversion tends to yield lower FID and higher density and coverage, while also minimizing L1 reconstruction error.

Comparing different image reconstruction generators. How do image priors different across different image reconstruction pipelines? Our encoder method relies on a pretrained generator, and trains an encoder to the latent space of that generator. Therefore, it can take advantage of the image priors learned by the generator, where the encoder keeps the result close to the image manifold. Here, we compare to different image reconstruction approaches, such as autoencoder architectures or optimization methods rather than feed-forward inference. We construct the same set of input collages using parts of real images, where inference times of each method is listed in Table 4.

For the church domain, we compare to three autoencoding methods, where the generator and encoder is learned jointly. (1) DIP (Ulyanov et al., 2018) learns an autoencoder to reconstruct a single image,

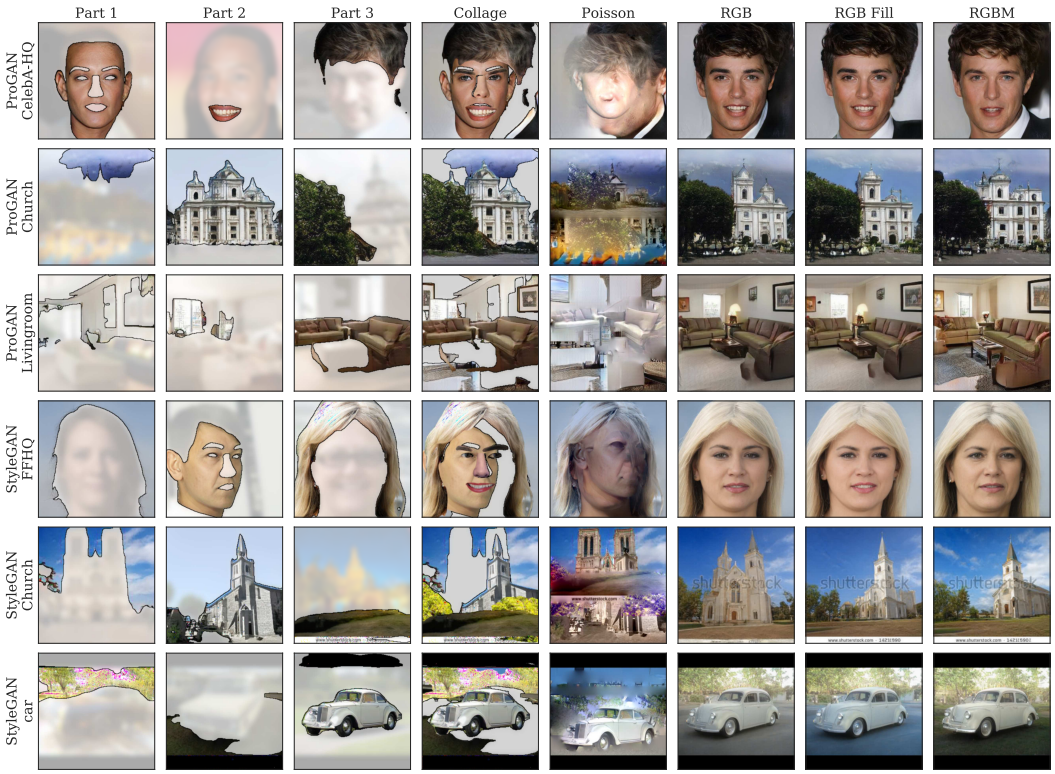


Figure 16: Qualitative examples of automatic image composition. We extract parts of sampled images and overlay them to form a rough collage. We compare poisson blending the image parts to encoder based methods that invert the collage through the generator, either without knowledge of missing pixels (RGB and RGB Fill methods), or with the objective of inpainting the missing regions (RGBM).

where a masked loss allows it to inpaint missing regions. (2) We train a CycleGAN (Zhu et al., 2017) between image collage and real image domain, creating a network that is explicitly trained for image composition. (3) We use a pretrained SPADE (Park et al., 2019) network which creates images from segmentation maps, but information about object style is lost in the segmentation input. Next, we use the ProGAN and StyleGAN2 pretrained generators, and experiment with different ways of inverting into the latent code, using LBFGS optimization, a trained encoder network, or both. We also investigate two additional iterative optimization approaches: Multi-Code Prior (Gu et al., 2019a) for image inpainting, and the StyleGAN2 projection method using perceptual loss (Karras et al., 2019a), which we modify to use a masked perceptual loss. Fig 17 shows qualitative results, where in particular the third example is a challenging input where the lower part of the tower is missing. The two encoders demonstrate an image prior in which the bottom of the tower is reconstructed on output. While DIP can inpaint missing regions, it cannot leverage learned image prior. CycleGAN can fill in missing patterns, but with visible artifacts, whereas SPADE changes the style of each input component. Iteratively optimizing on a pretrained generator may fail to reconstruct the the input collage if an appropriate initialization is not found, and it can also lose semantic priors as it optimizes towards an unrealistic input.

On the face domain we find that the optimization-based Im2StyleGAN Abdal et al. (2019) algorithm is not able to realistically inpaint missing regions in the input collage. While the ALAE autoencoder (Pidhorskyi et al., 2020) exhibits characteristics of blending the collage into a cohesive output image, the reconstruction error is higher than that of the GAN-based approaches. The In-domain encoder method (Zhu et al., 2020) is negatively impacted by the missing pixels in the input, resulting in low density, although the subsequent optimization step is able to reduce L1 distance (Fig. 18).

Composing images using alternative definitions of image parts. In the main text, we focus on creating composite images using a pretrained segmentation network. However, we note that the exact

Table 3: Quantitative comparison of automated collaging and latent regression variations. The latent regressor and generator pulls the unrealistic composite images closer to the real manifold, yielding high density and coverage and lower FID on output, compared to the collaged inputs and poisson blending.

Model	Metric	GAN Samples	GAN Reconstructions	Collage	Collage Filled	Poisson Blended	RGB Inverted	Filled Inverted	RGBM Inverted
ProGAN Church	FID	8.01	6.72	21.20	27.35	24.12	10.85	12.90	9.40
	Density	0.94	1.04	0.15	0.15	0.45	1.02	0.92	1.23
	Coverage	0.78	0.82	0.22	0.22	0.41	0.71	0.74	0.78
	Masked L1	–	–	–	–	–	0.26	0.28	0.26
ProGAN Living Room	FID	13.17	10.46	72.82	69.95	47.84	19.50	17.70	14.90
	Density	0.69	0.86	0.01	0.03	0.28	0.98	0.81	1.05
	Coverage	0.63	0.69	0.02	0.02	0.26	0.60	0.59	0.64
	Masked L1	–	–	–	–	–	0.33	0.34	0.30
ProGAN CelebA-HQ	FID	10.43	12.38	81.82	82.09	53.76	16.24	15.35	17.41
	Density	1.27	1.55	0.11	0.14	0.28	1.14	1.19	1.69
	Coverage	0.83	0.84	0.03	0.03	0.15	0.72	0.74	0.78
	Masked L1	–	–	–	–	–	0.26	0.26	0.24
StyleGAN Church	FID	3.90	6.73	15.66	22.89	16.92	13.69	12.55	11.68
	Density	0.88	1.48	0.15	0.14	0.41	1.59	1.58	2.06
	Coverage	0.85	0.86	0.26	0.25	0.50	0.75	0.79	0.82
	Masked L1	–	–	–	–	–	0.29	0.31	0.28
StyleGAN Car	FID	2.31	8.67	17.44	141.21	15.38	17.37	93.43	15.78
	Density	1.07	1.68	0.37	0.42	0.61	1.82	1.48	1.66
	Coverage	0.94	0.86	0.44	0.36	0.54	0.80	0.73	0.80
	Masked L1	–	–	–	–	–	0.23	0.30	0.21
StyleGAN FFHQ	FID	2.86	36.85	92.15	92.00	36.54	53.16	55.21	47.67
	Density	1.17	2.40	0.16	0.17	0.26	1.65	1.54	2.01
	Coverage	0.90	0.68	0.02	0.02	0.24	0.51	0.49	0.56
	Masked L1	–	–	–	–	–	0.23	0.23	0.21

Table 4: Inference times of different image reconstruction methods Our ProGAN and StyleGAN encoder methods only rely on a forward pass for inference. DIP, CyleGAN, and SPADE are autoencoder-based networks where the decoder and encoder are trained jointly; DIP requires test-time optimization, while CycleGAN and SPADE do not. The remaining methods involve iterative optimization on the latent code of a pretrained generator, and thus are substantially slower.

	ProGAN Encoder	Stylegan Encoder	DIP	CycleGAN	SPADE	ProGAN LBFGS	ProGAN Enc+LBFGS	StylegGAN LBFGS	Stylegan Enc+LBFGS	Multi-Code Prior	StyleGAN Projection
Time(s)	0.02	0.07	98.90	0.10	0.07	263.04	66.55	387.38	176.96	477.93	86.87

process of extracting image parts does not matter, as the encoder and generator form an image prior that will remove inconsistencies in the input. In Fig. 19 we show composites generated using the output of a pretrained segmentation network Liu et al. (2018), and in Fig. 20 we show compositions using hand-selected compositional parts, where the parts extracted from each image do not have to correpond precisely with object boundaries.

Random composition samples. We show additional random samples of the generated composites across the ProGAN and StyleGAN2 generators for a variety of image domains in Fig. 21 and Fig. 22.

Figure 17: Qualitative examples of image composition comparing different encoder-decoder or generator-only setups. For each example, we show the input collage created from randomly selected parts of images (Input, and result of our forward-pass only method of encoding into a pretrained generator (ProGAN Encode and StyleGAN Encode) – while this is a less precise match because there is no per-image optimization, inference is fast and these reconstruction have higher density (closeness to image manifold) than the unrealistic inputs. In the remainder of the top row are three autoencoder-based methods (DIP, CycleGAN, and SPADE), in which the encoder and decode is trained jointly; as these are fully convolutional they lack global semantic priors. In the bottom row, we show methods that iteratively optimize a latent code of a pretrained generator to match a target image. While these can better match the target collage after optimizing for reconstruction, they are orders of magnitude slower, making real-time interaction infeasible, and also lose semantic priors (as in the "floating tower" in the third example).

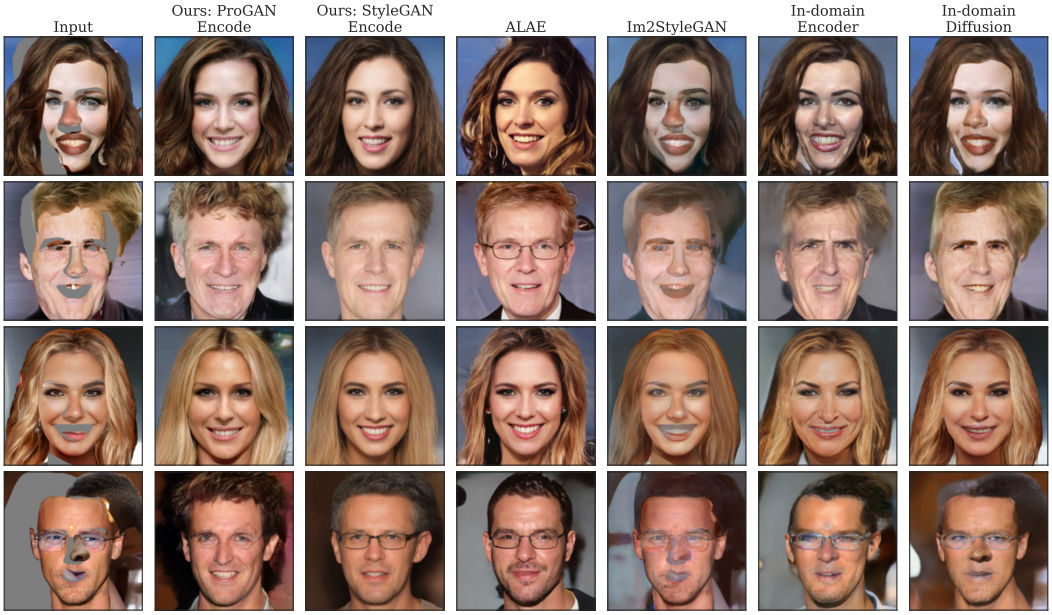


Figure 18: Qualitative examples of face composites across GAN-based generators. We show the input image, a collage from patches of random images (Input). We compare our masked-encoder approaches (ProGAN Encoder and StyleGAN encoder) to the ALAE autoencoder where the generator and encoder is trained jointly, the Im2StyleGAN method which iteratively optimizes for a latent code, the In-domain encoder which does not use masks during learning, and the In-domain diffusion step which subsequently optimizes the output of the encoder.

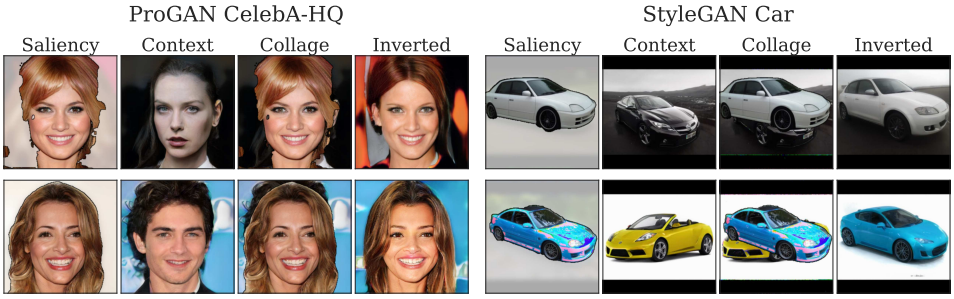


Figure 19: Qualitative examples of automatic image composition using a pretrained saliency network to generate input collages rather than a segmentation network. Note that in the second car example, the collage overlays the blue car with the yellow car still visible, but inversion via the encoder corrects this inconsistency.

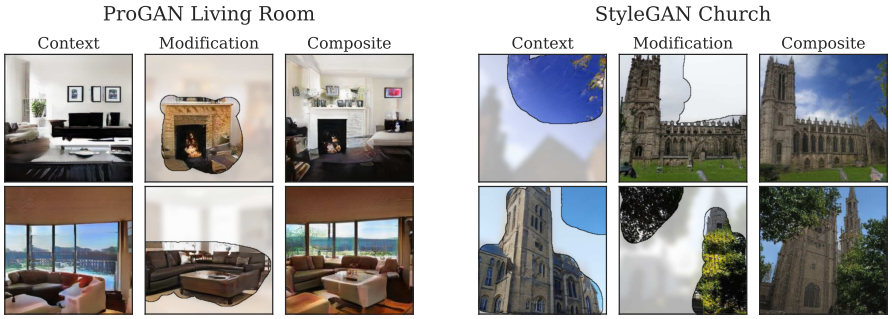


Figure 20: Qualitative examples of collages based on user-selected regions and the corresponding generator output. Note that the selected regions do not have to coincide neatly with object boundaries, and the generator will blend inconsistencies in the inputs together.

(a) ProGAN CelebA-HQ

(b) ProGAN Church

(c) ProGAN Living Room

Figure 21: Random samples of automatically generated colleges using a pretrained ProGAN generator.

(a) StyleGAN FFHQ

(b) StyleGAN Church

(c) StyleGAN Car

Figure 22: Random samples of automatically generated colleges using a pretrained StyleGAN generator.