Semantic Photo Manipulation with a Generative Image Prior

DAVID BAU, MIT CSAIL and MIT-IBM Watson AI Lab
HENDRIK STROBELT, IBM Research and MIT-IBM Watson AI Lab
WILLIAM PEEBLES, MIT CSAIL
JONAS WULFF, MIT CSAIL
BOLEI ZHOU, The Chinese University of Hong Kong
JUN-YAN ZHU, MIT CSAIL
ANTONIO TORRALBA, MIT CSAIL and MIT-IBM Watson AI Lab

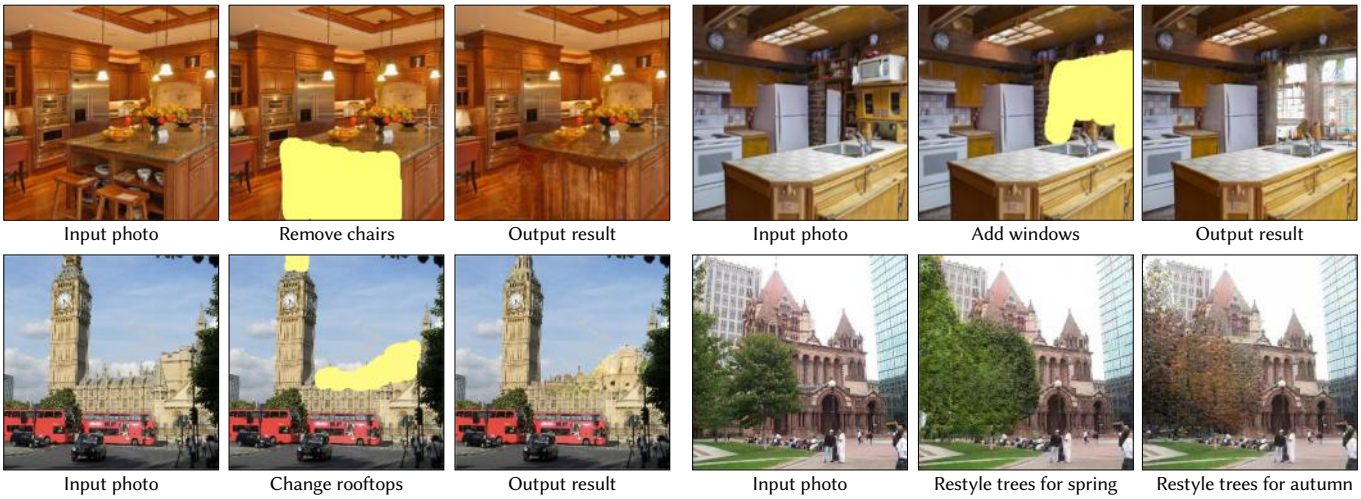

Fig. 1. Our proposed method enables several new interactive photo manipulations in which a user edits a photo with high-level concepts rather than pixel colors. Our deep generative model can synthesize new content that follows both the user's intention and the natural image statistics. *T*op: Given simple user strokes, our method can automatically synthesize and manipulate different objects while adjusting the surrounding context to match. *B*ottom: Our users can edit the visual appearance of objects directly, such as changing the appearance of rooftops or trees. Photos from the LSUN dataset [Yu et al. 2015].

Despite the recent success of GANs in synthesizing images conditioned on inputs such as a user sketch, text, or semantic labels, manipulating the high-level attributes of an existing natural photograph with GANs is challenging for two reasons. First, it is hard for GANs to precisely reproduce an input image. Second, after manipulation, the newly synthesized pixels often do not fit the original image. In this paper, we address these issues by adapting the image prior learned by GANs to image statistics of an individual image. Our method can accurately reconstruct the input image and synthesize new content, consistent with the appearance of the input image. We demonstrate our interactive system on several semantic image editing tasks, including synthesizing new objects consistent with background, removing unwanted objects, and changing the appearance of an object. Quantitative and qualitative comparisons against several existing methods demonstrate the effectiveness of our method.

CCS Concepts: • **Computing methodologies** → **Image representations**; **Neural networks**; **Image manipulation**.

Additional Key Words and Phrases: image editing, generative adversarial networks, deep learning, vision for graphics

ACM Reference Format:
David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. 2019. Semantic Photo Manipulation with a Generative Image Prior. *ACM Trans. Graph.* 38, 4, Article 59 (July 2019), 11 pages. https://doi.org/10.1145/3306346.3323023

Authors' addresses: David Bau, MIT CSAIL and MIT-IBM Watson AI Lab, davidbau@csail.mit.edu; Hendrik Strobelt, IBM Research and MIT-IBM Watson AI Lab, hendrik.strobelt@ibm.com; William Peebles, MIT CSAIL, wisp@csail.mit.edu; Jonas Wulff, MIT CSAIL, jwulff@csail.mit.edu; Bolei Zhou, The Chinese University of Hong Kong, bzhou@ie.cuhk.edu.hk; Jun-Yan Zhu, MIT CSAIL, junyanz@csail.mit.edu; Antonio Torralba, MIT CSAIL and MIT-IBM Watson AI Lab, torralba@csail.mit.edu.

PREPRINT
to appear in ACM Trans. of Graphics (TOG) SIGGRAPH 2019

1 INTRODUCTION

The whirlwind of progress in deep learning has produced a steady stream of promising generative models [Goodfellow et al. 2014; Karras et al. 2018] that render natural scenes increasingly indistinguishable from reality and provide an intuitive way to generate realistic imagery given high-level user inputs [Bau et al. 2019; Wang et al. 2018].

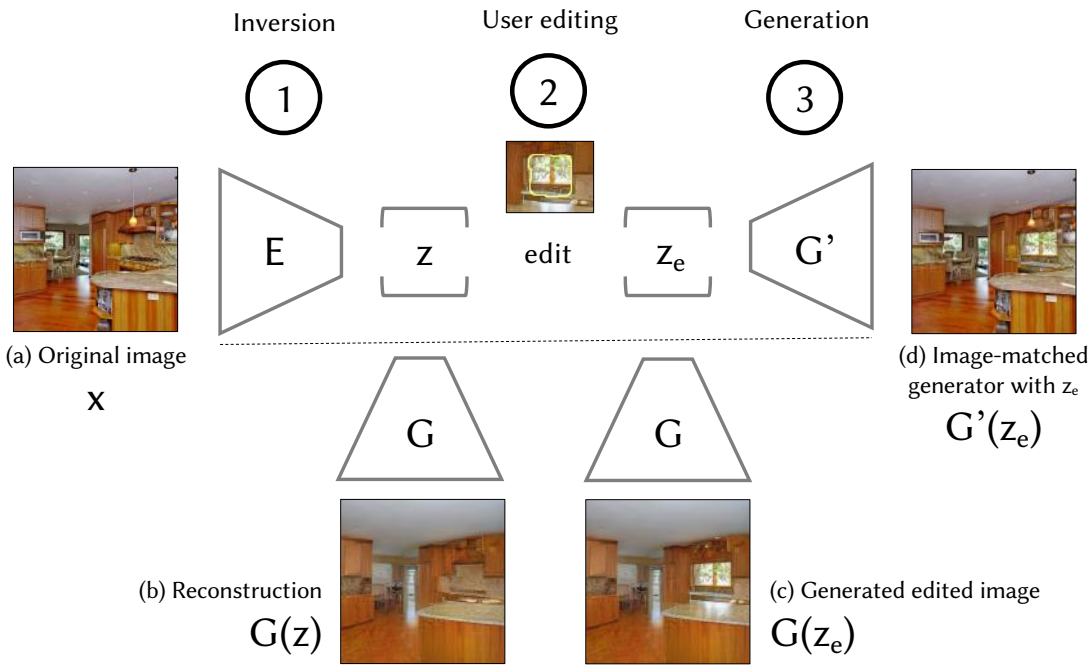

Fig. 2. *Overview.* To perform a semantic edit on an image x, we take three steps. (1) We first compute a latent vector $z = E(x)$ representing x. (2) We then apply a semantic vector space operation $z_e = \texttt{edit}(z)$ in the latent space; this could add, remove, or alter a semantic concept in the image. (3) Finally, we regenerate the image from the modified z_e. Unfortunately, as can be seen in (b), usually the input image x cannot be precisely generated by the generator G, so (c) using the generator G to create the edited image $G(x_e)$ will result in the loss of many attributes and details of the original image (a). Therefore to generate the image we propose a new last step: (d) We learn an image-specific generator G' which can produce $x'_e = G'(z_e)$ that is faithful to the original image x in the unedited regions. Photo from the LSUN dataset [Yu et al. 2015].

We seem to be on the verge of using these generative models for *semantic manipulation of natural photographs*, which will combine the simplicity of pixel-level editing with flexible semantic manipulations of a scene. The role of deep generative models will be to provide latent semantic representations in which concepts can be directly manipulated and then to preserve image realism when semantic changes are made. This editing scheme will allow users to manipulate a photograph not with physical colors, but with abstract concepts such as object types and visual attributes. For example, a user can add a lit lamp into a bedroom (Figure 1 bottom left) or change the color of a tree's leaves (Figure 1 bottom right) with a few scribbles.

Despite the promise of this approach, two technical challenges have prevented these generative models from being applied to natural photograph manipulation. First, it is extremely difficult to find a latent code z to reproduce a given photograph x with a deep generative model G: $x \approx G(z)$. As shown in Figure 2b, the reconstructed image $G(z)$ roughly captures the visual content of the input image x, but visual details are obviously different from the original photo. Second, after manipulation, the newly synthesized pixels from generative models are often incompatible with the existing content from the real image, which makes stitching the new content into the context of the original image challenging (Figure 2c).

In this paper, we address the above two issues using an image-specific adaptation method. Our key idea is to learn an *image-specific generative model* $G' \approx G$, that produces a near-exact solution for our input image x, so that $x \approx G'(z)$ outside the edited region of the image. We construct our image-specific G' to share the same semantic representations as of the original G. Importantly, our image-specific G' produces new visual content, consistent with the original photo while reflecting semantic manipulations (Figure 2d).

We build `GANPaint editor`, an interactive interface that supports a wide range of editing tasks, from inserting new objects into a natural photo (Figure 1 top) to changing the attributes of existing objects (Figure 1 bottom). We show that our general-purpose editing method outperforms compositing-based methods as measured in human perception studies. Finally, we perform an ablation study demonstrating the importance of our image-specific adaptation method compared to previous reconstruction methods. Our code, models, and data are available at our website ganpaint.csail.mit.edu.

2 RELATED WORK

Generative Adversarial Networks. (GANs) [Goodfellow et al. 2014] learn to automatically synthesize realistic image samples [Karras et al. 2018; Miyato et al. 2018]. GANs have enabled several user-guided image synthesis tasks such as generating images from user sketches [Isola et al. 2017; Sangkloy et al. 2017], creating face animation [Geng et al. 2018; Nagano et al. 2018], synthesizing photos from language description [Zhang et al. 2017a], and interactively manipulating objects in a generated scene [Bau et al. 2019; Park et al.

2019]. While most of the prior work focuses on generating a new image *from scratch* given user controls, little work has used GANs for interactively manipulating an existing natural photograph. The key challenge is the mismatch between the GAN-generated content and existing content in the image. Several lines of work [Brock et al. 2017; Perarnau et al. 2016; Zhu et al. 2016] propose to manipulate a photo using GANs but only work with a single object (e.g., handbag) at low resolutions (64x64) and often involve post-processing steps. In this work, our method can directly generate a final result and allow semantic manipulations of an entire natural scene.

Interactive Photo Manipulation. Manipulating a natural photograph is a central problem in computer graphics and computational photography, where a piece of software manipulates an image to achieve a user-specified goal while keeping the result photorealistic. Example applications include color adjustment [An and Pellacini 2008; Levin et al. 2004; Reinhard et al. 2001; Xue et al. 2012], tone mapping [Durand and Dorsey 2002], image warping [Avidan and Shamir 2007], image blending [Pérez et al. 2003; Tao et al. 2010], or image reshuffling [Barnes et al. 2009], to name just a few. These editing tools can modify the low-level features of an image with simple interactions (e.g. scribbles) and work well without external knowledge. On the contrary, to edit the high-level semantics of an image such as the presence and appearance of objects, prior work often demands manual annotations of the object geometry [Kholgade et al. 2014] and scene layout [Karsch et al. 2011], choice of an appropriate object [Lalonde et al. 2007; Pérez et al. 2003], or RGBD data [Zhang et al. 2016a]. Different from those systems, our method allows complex high-level semantic manipulations of natural photos with a simple brush-based user interface. We use natural image statistics learned by a deep generative model to connect simple user interaction to the complex visual world.

Deep Image Manipulation. Deep learning has achieved compelling results in many image editing tasks. Recent examples include image inpainting [Iizuka et al. 2017; Pathak et al. 2016; Yu et al. 2018], image colorization [Iizuka et al. 2016; Zhang et al. 2016b], photo stylization [Gatys et al. 2016; Li et al. 2018; Zhu et al. 2017] and photo enhancement [Gharbi et al. 2017; Kim and Park 2018]. Learning-based systems allow real-time computation and require no hand-crafted heuristics. Recent work further integrates user interaction into end-to-end learning systems, enabling interactive applications such as user-guided colorization [Zhang et al. 2017b] and sketch-based face editing [Portenier et al. 2018]. These methods can achieve high-quality results, but the editing task is fixed at training time and requires specific training data. In this work, we propose an alternative, task-agnostic approach. We learn the natural image statistics using a generative model and allow many different editing applications in the same framework. This enables new visual effects where training data is not available, such as adding objects and changing the appearance of objects.

3 METHOD

Overview: We propose a general-purpose semantic photo manipulation method that integrates the natural image prior captured by a GAN generator. Figure 2 shows our image editing pipeline. Given a natural photograph as input, we first re-render the image using an image generator. More concretely, to precisely reconstruct the input image, our method not only optimizes the latent representation but also adapts the generator. The user then manipulates the photo using our interactive interface, such as by adding or removing specific objects or changing their appearances. Our method updates the latent representation according to each edit and renders the final result given the modified representation. Our results look both realistic and visually similar to the input natural photograph.

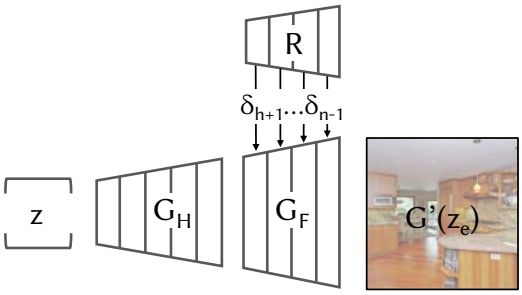

Fig. 3. Image-specific model adaptation through training a small network R that produces perturbations δ_i that influence the later, fine-grained layers G_F of the GAN. We add a regularization term to encourage the perturbations to be small.

Below, we first review recent GAN-based methods for generating visual content given semantic user inputs in Section 3.1. In theory, we could apply these methods to edit an existing photo as long as we can reproduce an image using a learned GAN generator. In Section 3.2, we show that a rough reconstruction is possible, but a precise reconstruction is challenging and has eluded prior work. In Section 3.3, we present our new image-specific adaptation method to bridge the gap between the original photo and generated content. Finally, we describe several image manipulation operations powered by our algorithm in Section 3.4.

3.1 Controllable Image Synthesis with GANs

Deep generative models [Goodfellow et al. 2014; Kingma and Welling 2014] are proficient at learning meaningful latent representation spaces. An encoder-based generative model is a function $G : \mathrm{z} \rightarrow \mathrm{x}$ that generates an image $\mathrm{x} \in \mathbb{R}^{H \times W \times 3}$ from a latent representation $\mathrm{z} \in \mathbb{R}^{h \times w \times |z|}$. This representation can be a low-dimensional vector from a Gaussian distribution (i.e., $1 \times 1 \times |z|$) or an intermediate feature representation in the generator. In this work, we use the intermediate representation for flexible spatial control.

Using a generative model G for photo editing is powerful for two reasons. First, arithmetic vector operations in the latent representation space often result in interesting semantic manipulations. For example, given a generated image from the model $G(\mathrm{z})$, latent vector operations $\mathrm{z}_e = \texttt{edit}(\mathrm{z})$ can adjust the species of an animal [Miyato et al. 2018], the orientation of an object [Chen et al. 2016], or the appearance [Zhu et al. 2016] or presence [Bau et al. 2019] of objects in a scene. Second, the edited image $G(\texttt{edit}(\mathrm{z}))$ will still lie on the natural image manifold as G is trained to produce a natural image given any latent code.

Fig. 4. GANBrush user interface. A new image is first uploaded and inverted. The toolbar (left of image) then allows users to select the mode of operation (draw or erase), select the semantic feature, and select brush-size and feature strength (low, med, high). The history panel (right of image) shows a stack of modifications in chronological order. The corresponding edits are highlighted in the image when users hover over previous edits. Each edit can also be removed from the list. Photo being edited is of St Luke the Evangelist, Cwmbwrla, Swansea courtesy Jaggery, via Geograph (cc-by-sa/2.0).

Our method can be applied to various generative models and different latent representation editing methods. In this work, we focus on manipulating semantic object-centric representations that can add, remove, and alter objects such as trees and doors within a natural scene. For that, we build our work on a state-of-the-art model, progressive GANs [Karras et al. 2018], and a recent latent representation editing method [Bau et al. 2019]. Figure 1 shows several editing examples on real input images. The details of these latent space edits are discussed in Section 3.4.

3.2 Reproducing a Natural Image with a Generator

Applying a semantic edit as described above on a natural image requires reconstructing the input image x in the latent space (i.e. finding a z so that $\mathrm{x} \approx G(\mathrm{z})$), applying the semantic manipulation $\mathrm{z}_e = \texttt{edit}(\mathrm{z})$ in that space, and then using the generator $\mathrm{x}_e = G(\mathrm{z}_e)$ to render the modified image (Figure 2). Formally, we seek for a latent code z that minimizes the reconstruction loss $\mathcal{L}_r(\mathrm{x}, G(\mathrm{z}))$ between the input image x and generated image $G(\mathrm{z})$:

$$\mathcal{L}_r(\mathrm{x}, G(\mathrm{z})) = \|\mathrm{x} - G(\mathrm{z})\|_1 + \lambda_{\mathrm{VGG}} \sum_{i=1}^{N} \frac{1}{M_i} \|F^{(i)}(\mathrm{x}) - F^{(i)}(G(\mathrm{z}))\|_1, \tag{1}$$

where we use both color pixel loss and perceptual loss, similar to prior image reconstruction work [Dosovitskiy and Brox 2016; Zhu et al. 2016]. Here $\lambda_{\mathrm{VGG}} = 10$ and $F^{(i)}$ is the i-th layer with M_i features in the VGG network [Simonyan and Zisserman 2015]. To speed up the reconstruction, we follow the prior work [Zhu et al. 2016] and train an encoder $E\colon \mathrm{x} \rightarrow \mathrm{z}$ that predicts the latent code directly from the image. We use a similar training loss $\arg\min_E \mathbb{E}_{\mathrm{x}\sim p_{data}(\mathrm{x})} \mathcal{L}_r(\mathrm{x}, G(E(\mathrm{x})))$. During test time, we use $E(\mathrm{x})$ as the initialization for the optimization of Eqn. 1.

Unfortunately, this method fails to produce visually appealing edits when G is not able to generate images resembling x. Finding a latent code z that can reproduce an *arbitrary image* x is hard because, for many images, the range of the generator G does not include any image sufficiently similar to x in appearance. As shown in Figure 2, existing reconstruction methods [Dosovitskiy and Brox 2016; Zhu et al. 2016] can only roughly re-generate the color and shape of objects in the scene and fail to reproduce the visual details of input images faithfully. Given an inaccurate reconstruction, subsequent edits introduce further artifacts. Besides, the unedited regions of the final result may look different from the input photo due to the reconstruction error.

3.3 Image-Specific Adaptation

The image reconstruction problem is hard as it would require the same generator G to be able to reproduce every single detail of every possible input image. And to be useful for incremental editing, new results $G(\texttt{edit}(\mathrm{z}))$ need to be compatible with the input image x as well. To address these issues, we propose to use an image-specific generator G' that can adapt itself to a particular image. First, this image-specific generative model G' can produce a near-exact match for our input image x. Second, our image-specific G' should be close to G so that they share an underlying semantic representation. Learned with the above two objectives, G' can preserve the visual details of the original photo during semantic manipulations.

More precisely, to perform a successful edit, G' does not strictly need to have x itself in its range; rather, we find G' to exactly reproduce only the unedited regions of the input image. Given a user stroke binary mask, $\texttt{mask}_e$:

$$\texttt{mask}_e = \begin{cases} 1 & \text{where the stroke is present} \\ 0 & \text{outside the stroke} \end{cases} \tag{2}$$

When adapting G', this constraint can be approximated by minimizing a simple difference between the input image x and those generated by $G'(\mathrm{z}_e)$, summed over the image regions outide of the strokes.

$$\mathcal{L}_{\mathrm{match}} \equiv ||(G'(\mathrm{z}_e) - x) \odot (1 - \texttt{mask}_e)||_1, \tag{3}$$

where we set $\mathrm{z}_e = \texttt{edit}(\mathrm{z})$, and $\odot$ is the elementwise Hadamard product. The operation $\texttt{edit}(\mathrm{z})$ expresses the user's intent to apply a particular semantic manipulation on the deep latent structure of G; it assumes that we can find a G' with a similar latent structure as G. Otherwise, the editing operation $\texttt{edit}(\mathrm{z})$ may not work well for newly constructed G'.

Preserving Semantic Representation. To ensure that the image-specific generator G' has a similar latent space structure as the original generator G, we construct G' by preserving all the early layers of G precisely and applying perturbations only at the layers of the network that determine the fine-grained details.

This is done by exploiting the internal structure of modern image generator networks: a generator G has a layered structure consisting of a series of convolutions at increasing resolutions, where the final layer g_n is close to pixels and captures the fine-grained details, while the first layer g_1 is closest to the latent representation z and captures high-level information:

$$G(z) = g_n(g_{n-1}(\cdots(g_1(\mathrm{z})).\cdots)) \tag{4}$$

The early layers of a generator represent high-level semantics such as the presence and layout of objects, while later layers encode lower-level pixel information such as edges and colors, as observed by Bau et al. [2019]. Therefore, to leave the semantic structure of G unchanged, we divide it into a group of high-level layers G_H containing layers 1 through h and fine-grained layers G_F containing layers $h+1$ through n, so that $G(\mathrm{z}) \equiv G_F(G_H(\mathrm{z}))$. This division is illustrated in Figure 3. Only G_F are adjusted when creating G'. The early layers G_H that decode the high-level structure of the image remain unchanged. In detail, we define G_H and G_F as:

$$\begin{aligned} \mathrm{z}_h \equiv\ & G_H(\mathrm{z}) \equiv g_h(g_{h-1}(\cdots g_1(\mathrm{z})\cdots)) \\ & G_F(\mathrm{z}_h) \equiv g_n(g_{n-1}(\cdots(g_{h+1}(\mathrm{z}_h)\cdots)) \end{aligned} \tag{5}$$

The choice of h can be tuned experimentally; we have found it to be effective to choose $h = n-5$ so that the fine-grained section contains a pyramid of four scales of convolutions. To create a generator that faithfully reconstructs the target image x, we will update G_F. However, directly updating the weights of fine-grained layers G_F to match output to x causes overfitting: when changed in this way, the generator becomes sensitive to small changes in z and creates unrealistic artifacts.

Instead, we train a small network R to produce small perturbations δ_i that multiply each layer's output in G_F by $1+\delta_i$. Each δ_i has the same number of channels and dimensions as the featuremap of G_F at layer i. This multiplicative change adjusts each featuremap activation to be faithful to the output image. (Similar results can be obtained by using additive δ_i.) Formally, we construct G'_F as follows:

$$\begin{aligned} & G'_F(\mathrm{z}_h) \equiv g_n((1+\delta_{n-1}) \odot g_{n-1}(\cdots((1+\delta_{h+1}) \odot g_{h+1}(\mathrm{z}_h)\cdots))) \\ & G'(\mathrm{z}) \equiv G'_F(G_H(\mathrm{z})). \end{aligned} \tag{6}$$

The perturbation network R learns to produce δ_i starting from a random initialization; R takes no input. Figure 3 illustrates the architecture of this perturbation network.

To further prevent overfitting, we add a regularization term to penalize large perturbations:

$$\mathcal{L}_{\text{reg}} \equiv \sum_{i=h+1}^{n-1} ||\delta_i||^2 \tag{7}$$

.

Overall optimization. The overall optimization of G' can now be summarized. To learn G', we fix the edited semantic representation z_e and the unedited pixels $\mathrm{x} \odot \mathtt{mask}_e$ that must be matched. Furthermore, we fix all pre-trained layers of the generator G.

Our objective is to learn the parameters of perturbation network R to minimize the following loss:

$$\mathcal{L} = \mathcal{L}_{\text{match}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}. \tag{8}$$

We learn the network R using the standard Adam solver [Kingma and Ba 2015] with a learning rate of 0.1 for 1000 steps. The weight λ_{reg} balances the magnitude of perturbations of G' and closeness of fit to the target pixels. We empirically set it to 0.1.

The optimization takes less than 30 seconds on a single GPU. Fewer steps can be used to trade off quality for speed. While 1000 steps achieve an average PSNR of 30.6 on unedited reconstruction of a sample, it takes 100 steps to achieve PSNR of 24.6.

The computational form of adapting a network to a single image is inspired by previous work on deep image prior [Ulyanov et al. 2018] and deep internal learning [Shocher et al. 2018], which have been shown to be effective at solving inpainting and super-resolution without any training data beyond a single image. However, our application is different; we initialize G with a generative model trained on a distribution of images in order to synthesize a semantic change encoded in the latent vector z_e, and then we train on a single target image in order to blend the semantic edits with the original image.

3.4 Semantic Editing Operations: GANPaint

Our user interface (Figure 4) enables interactive editing of images that can be uploaded by users. Although computing $G'(\mathrm{z}_e)$ requires some time, the interactive editor provides real-time previews $G'_w(\mathrm{z}_e)$ using a G'_w that is fast to apply. The function G'_w is set to a perturbation of the generator G with weights that are optimized so that $G'_w(\mathrm{z}) \approx \mathrm{x}$. Since G'_w is derived only from the pixels of the unedited image x, it can be computed once when the image is uploaded and applied quickly for each edit. Although $G'_w(\mathrm{z}_e)$ introduces more visual artifacts compared to $G'(\mathrm{z}_e)$, it gives a good indication of what the final rendering will look like at interactive rates. Currently, our web-based interface requires a server-side GPU for inference.

To demonstrate semantic editing, we use Progressive GAN models [Karras et al. 2018] that can generate realistic scene images trained on the LSUN dataset [Yu et al. 2015]. Once trained, this generator consists of 15 convolutional layers and produces images at a resolution of 256×256 pixels. We closely follow methods of GAN Dissection [Bau et al. 2019] to paint a photo by editing middle-level latent representations located on the feature maps between the 4th and 5th layer. The representation $\mathrm{z} \in \mathbb{R}^{8\times8\times512}$ is a 512-channel tensor with featuremaps of size 8×8. The GAN dissection toolkit [Bau et al. 2019] can be used to identify a set of object types and concepts C which we allow the user to insert, remove or change the appearances of objects in their images, as detailed below.

Adding and removing objects. Using our GANPaint editor Figure 4, users can select regions of an image they would like to edit. To insert or remove an object belonging to class c in a user-selected region $U \in \mathbb{R}^{8\times8}$, we form a channel mask $\alpha_c = (i_c \otimes U) \in \mathbb{R}^{8\times8\times512}$, to select specific features in z in the feature channels and locations that the user wishes to edit. The feature channels $i_c \in \mathbb{R}^{512}$ relevant to class c are obtained by analyzing the generator using [Bau et al. 2019], and $\otimes$ indicates a broadcasted outer product.

The edited representation $\mathrm{z}_e = \mathtt{edit}(\mathrm{z})$ is computed as follows:

$$\mathrm{z}_e := \underbrace{(1-\alpha_c) \odot \mathrm{z}}_{\text{activations retained from } z} + \underbrace{\alpha_c \odot (s\, p_c)}_{\text{edited activations}} \tag{9}$$

In the above, the representation z is blended with the feature vector sp_c, where $p_c \in \mathbb{R}^{8\times8\times512}$, is a spatially-expanded vector that represents the average activation of the object class c over all images, constant in each 8×8 channel, and s is a scalar which controls how much to shrink or boost edited activations. Setting $s = 0$ corresponds to removing class c from the representation, whereas setting $s > 0$ corresponds to adding class c.

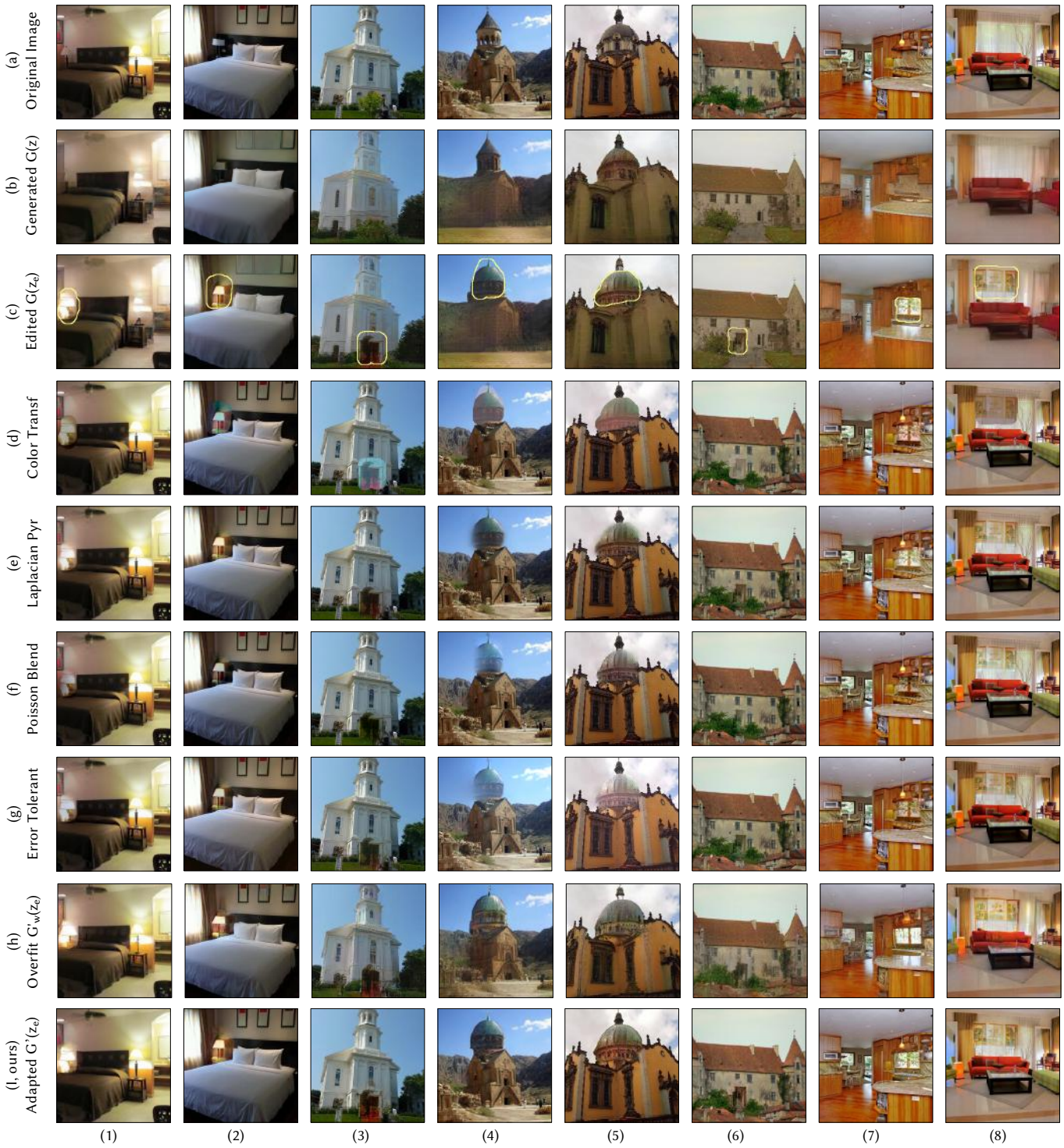

Fig. 5. Comparing our method to several compositing methods. From top to bottom: (a) the original unmodified image (b) its reconstruction by the unmodified generator; (c) the generated image $G(z_e)$ after applying a semantic edit to the latent vector z; (d) compositing the edited region from the edited generated (c) into the original image (a) using simple color transfer; (e) Laplacian pyramid blending; (f) Poisson blending; (g) Error-tolerant image compositing; (h) reconstruction of z_e using a G'_w where weights have been overfit to the unedited original image x; (i) our method: reconstruction of z_e using a G' with activations adapted to match image x outside the editing region. Photos from the LSUN dataset [Yu et al. 2015].

Changing the appearance of objects. The same system can also alter the appearance of an object of class c. To do this, we simply use equation (9) with a value of p_c derived from a *reference image* that has an appearance the user wishes to mimic. A simple way to accomplish this is to set the i-th component of p_c to be the mean of the positive activations in the i-th channel of the latent features z of the reference image for channels related to class c; all other elements of p_c are set to zero. See Figure 8 for an example of changing object appearances. Taking means of positive activations is only one possible way to incorporate information about the reference image's latent representation z into p_c; we leave exploring other ways of copying attributes to future work.

4 EXPERIMENTAL RESULTS

We evaluate each step of our method. In Section 4.1 we compare our image-specific adaptation method to several compositing methods, and in Section 4.2 we compare our method to two simplified ablations of our method. In Section 4.3, we show qualitative results applying our image-specific adaptation G' on both training set and in-the-wild test images. In Section 4.4 we demonstrate our method for changing the appearance of objects. In Section 4.5, we evaluate our method for recovering the latent vector z from an image x.

4.1 Comparing Image-Specific Adaptation to Compositing

By adapting the generator to the original image, our method blends the generated edited image with the unedited parts of the photograph. This scenario is similar to the task of image compositing, so we evaluate the performance of our method in comparison to several compositing approaches. We compare our method to Color transfer [Reinhard et al. 2001], Laplacian pyramid blending, Poisson editing [Pérez et al. 2003], and Error-Tolerant Image Compositing [Tao et al. 2010]. In each case, we use the traditional compositing method to insert the edited image pixels generated by $G(z_e)$ from the edited region into the target photograph.

We asked 206 Amazon MTurk workers to compare the realism of the results of each compositing algorithm with our results. For each method, 1200 pairwise quality comparisons were collected on 20 different edited images. For each comparison, workers are shown the real photograph and the edited region as generated by $G(\mathrm{x}_e)$, then asked which image from a pair is the most realistic insertion of the object into the photo. Top rows of Table 1 summarize the results. Qualitative comparisons are shown in Figure 5(defg).

Workers find that our method yields more realistic results than traditional image blending approaches on average. For example, in the case Figure 5(c4), our method blends the new dome into the old building while maintaining edges, whereas traditional blending methods do not differentiate between boundary pixels that should be crisp edges from those that require smooth blending.

4.2 Ablation Studies

To study the necessity of adapting G' by perturbing activations, we compare our method $G'(\mathrm{z}_e)$ to two simpler generator-based approaches. First, we compare our method to $G(\mathrm{z}_e)$ itself without any perturbations in G. Second, we compare to an adapted $G'_w(\mathrm{z}_e)$ in which the image is rendered by an adapted G'_w with weights that have been optimized to fit the unedited image, so that $G'_w(\mathrm{z}) \approx \mathrm{x}$. Although G'_w is adapted to x, it is adapted to all pixels of the unedited image without regard to the editing region. Results are summarized in bottom rows of Table 1, and in Figure 5(ehi). Note that the question asked for AMT raters shows the edited $G(\mathrm{z}_e)$ as the example of the foreground edit that should be inserted in the photo. Thus raters are primed to look for the editing changes that are demonstrated in image; this might result in ratings for $G(\mathrm{z}_e)$ that could be higher than they would without this prompt.

Table 1. AMT evaluation of compositing methods compared to our method: we report the percentage of users that prefer various other methods over ours. Our method is also compared to the unadapted generator G as well as a directly adapted generator G'_w in which the weights have been fitted so $G'_w(z) \approx x$.

Method	% prefer vs ours
Color transfer [Reinhard et al. 2001]	16.8%
Error-tolerant image compos. [Tao et al. 2010]	43.6%
Poisson blending [Pérez et al. 2003]	44.2%
Laplacian pyramid blending	47.2%
Our method	50.0%
$G(z_e)$ without adaptation	37.4%
$G'_w(z_e)$, weights are fitted so $G'_w(z) \approx x$	33.1%

Compared to the two ablations, our method produces results that are with fewer visible artifacts in the output image that are rated as much more realistic on average. In particular, artifacts introduced by G'_w are seen as high-frequency differences between the target image and the generated image, as well as additional color artifacts near the edited region. These artifacts can be seen by zooming into the examples in Figure 5(h).

4.3 Qualitative Results

In this section, we show qualitative results of applying image-specific adaptation on natural photographs.

Editing LSUN images. In Figure 6, we show several edits on images from the LSUN datasets [Yu et al. 2015]. In the first column, we show the original unedited image, which is inverted (column two). The segmentation mask in column three indicates the edits requested by the user and the resulting GAN-generated image is shown in column four ($G(\mathrm{z}_e)$).

To the right (column five) we show the results of applying image-specific adaptation on edits using the same user requests. All examples are taken from the LSUN dataset on which the GANs were trained (living rooms, kitchens, outdoor church images).

Editing In-the-wild images. In Figure 7, we examine editing results on newly collected images that are not in the training set. All edits are applied using a GAN trained on 'outdoor church' images.

As can be seen in the second column of Figure 7, rendering of $G(z)$ reveals parts of images such as building shapes, doors, windows, and surfaces that are modeled by G. Many parts of the new church images are modeled, and our image-matching method is able to

Fig. 6. Examples of editing work-flow. From left to right: input image x is first converted to GAN image $G(z)$, edited by painting a mask, the effect of this mask edit can be previewed at interactive rates as $G(z_e)$. It can be finally rendered using image-specific adaption as $G'(z_e)$. Photos from LSUN [Yu et al. 2015].

apply edit to these parts of the image in the presence of objects such as cars and lampposts that are not generated by $G(z)$.

4.4 Style Variants

We demonstrate varying styles of inserted objects in Figure 8. The variants are produced by using different source images ((a) and (b)) and by using the same image but different strengths of style adaptation (c).

4.5 Recovering the Latent Vector z

Our method depends on starting with a latent vector z that is able to approximate the user's photo. Here we report our results in recovering an inverse $E \approx G^{-1}$. We find z to optimize the latent vector x $\approx G$(z) in the following two steps.

First, we train a network E_{net}(x) to minimize the mean loss Eqn. 1 when inferring z from x. We find that ResNet-18 is effective at computing a good estimate for z. We can quantitatively evaluate the accuracy of E_{net} at recovering the correct z by testing it on images

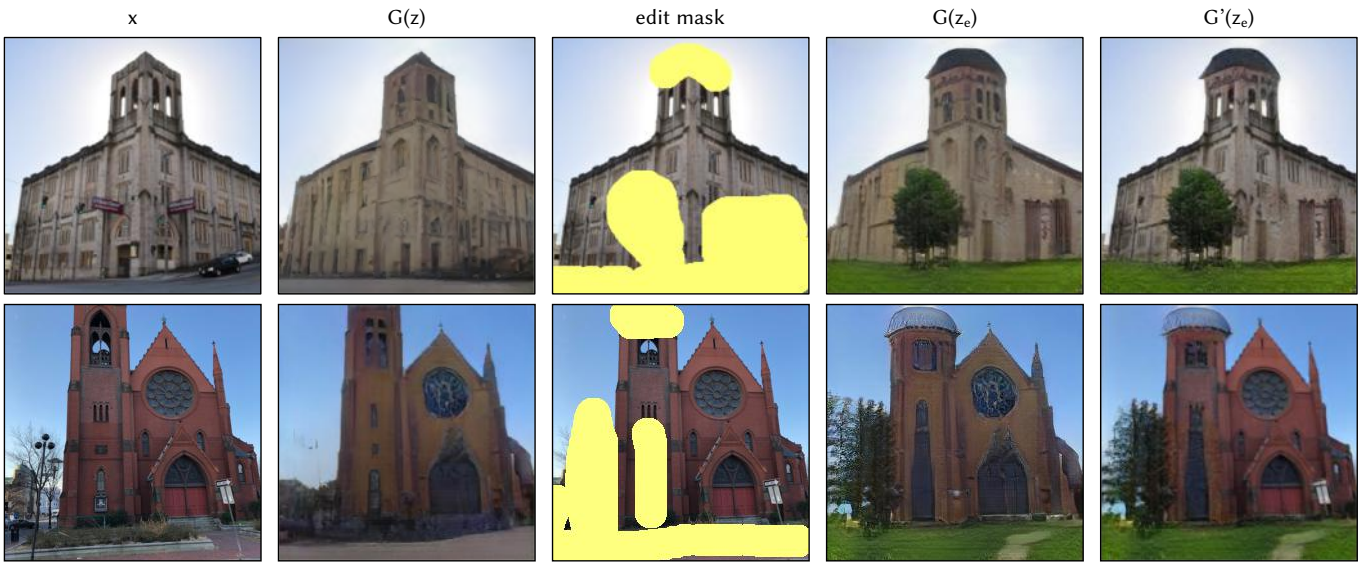

Fig. 7. Applying our method on 'in the wild' images. These images are newly collected images that are not in the training set. Images are edited using a GAN model trained on churches: new church images respond well to editing, even if they contain a few idiosyncratic objects (such as cars and lampposts) not modeled by the generator. Photo of Urban Grace Church courtesy Visitor7 via Wikipedia (cc-by-sa/3.0); photo of First Baptist Church by the authors.

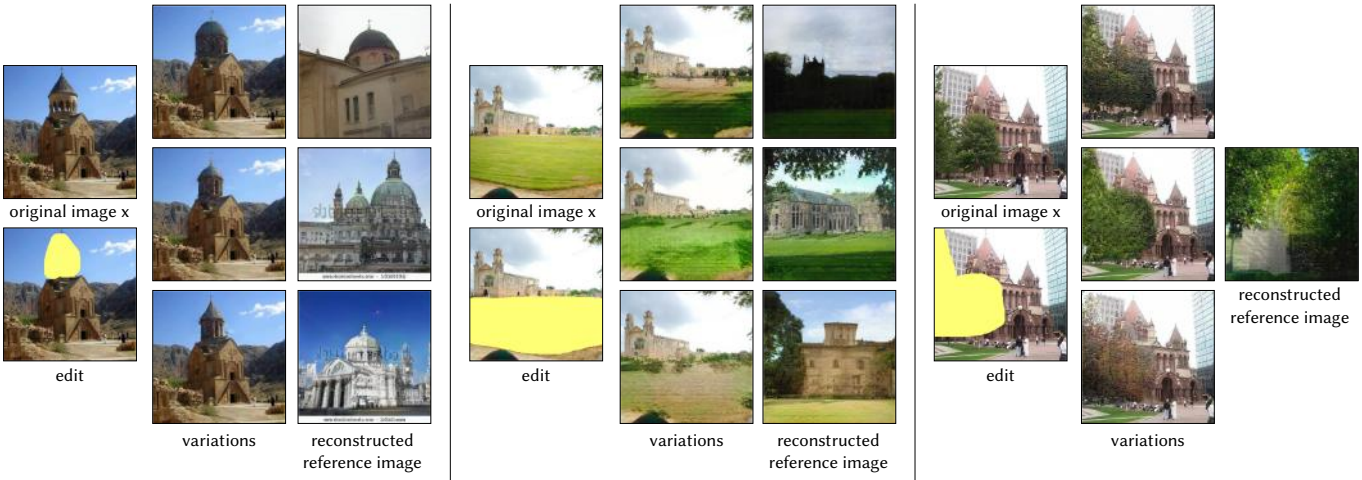

Fig. 8. Changing the appearance of domes, grass, and trees. In each section, we show the original image x, the user's edit overlayed on x and three variations under different selections of the reference image. Additionally, we show reconstructions of the reference image from G. In (c), we fix the reference image and only vary the strength term s. Photos from the LSUN dataset [Yu et al. 2015].

$x = G(z_{true})$ for which the true latent vector z_{true} is known. The results are shown in Figure 9 (a).

Next, we apply gradient descent to optimize the latent vector z using the same objective function (Eqn. 1). Our case, $z = z_4$ is a fourth-layer representation of an underlying progressive GAN, which means that we do not have explicit statistics for regularizing this representation. We have found that an effective implicit regularizer for z_4 can be obtained by optimizing over the earlier layers $z_1, z_2,$ *and* z_3 rather than z_4 itself. Using this approach results in the recovery of nearly perfect z_4, as shown in Figure 9 (b). We use $E(x)$ to denote the result of both steps of the optimization. Figure 10 shows a qualitative comparison of reconstructions $G(E(x))$ with input images x for two pairs of images. We observe that, for x that are generated by G, the inversion is nearly perfect: the reconstructed image is indistinguishable from the original. Figure 10 also shows reconstructions $G(E(x))$ for real images x that are *not* generated by G. In this setting, the inversions are far from perfect. Furthermore, the failure of E on other images reveals the types of images that G cannot reconstruct, and provide insight into the limitations of the generator. For example, in a model trained on outdoor scenes

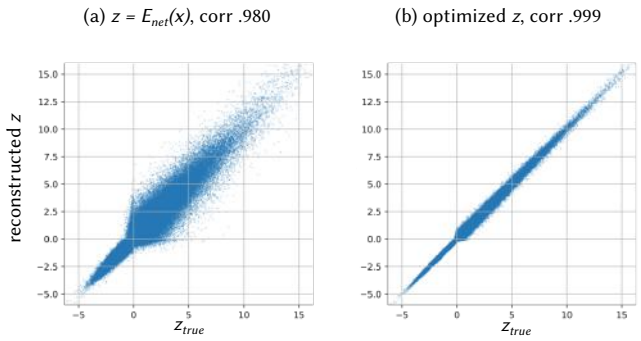

Fig. 9. A generator G can be inverted accurately within its range. Two inversion methods are tested on images generated by $x = G(z_{true})$, and the components of the predicted $z = E(x)$ are plotted against the true components of the known z. In (a), a Resnet-18 network is trained to calculate G^{-1} and achieves good precision. In (b), the network-computed results are used to initialize an optimization that further refines the calculation of z. The result is a recovery of the true z above a Pearson's correlation about 99.9%. Thus inversion works well for generated images. However, this does not imply that these methods can invert real photos. Images from the LSUN dataset [Yu et al. 2015].

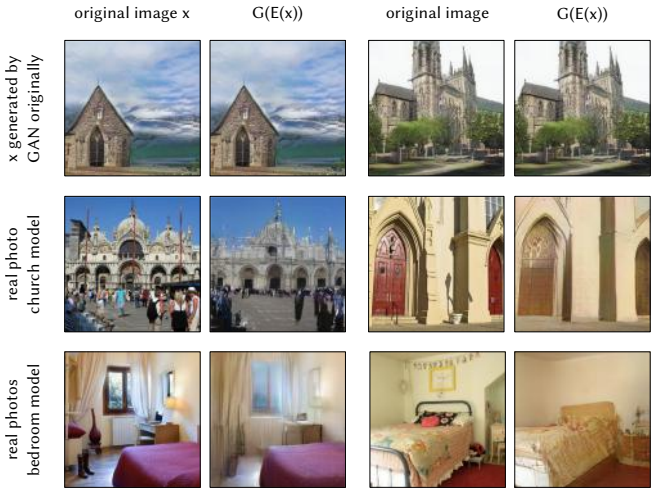

Fig. 10. Recovery of latent vectors z from images x: results can be compared qualitatively by rendering $G(z)$ and comparing to the original image x. For images that were originally generated by the generator, our algorithm can recover z that is quantitatively and qualitatively identical to its true value. However, this is not true for many real photos x. Inverting and regenerating the image reveals details that the generator cannot represent; for the two models shown here, these details can include people and objects that are not usually seen in the data set. Photos from LSUN [Yu et al. 2015].

of churches, the inversions cleanly drop out people and vehicles, suggesting that this model cannot represent these types of objects well. The model trained on bedrooms drops out certain types of decorations.

5 LIMITATIONS AND DISCUSSION

Although our method for adapting a generative network allows various semantic photo manipulations, many challenges remain.

Fig. 11. *Failure cases.* Some latent vector space operations are difficult to disentangle from undesired effects. For example, in this sequence, a user has attempted to use a latent space manipulation to remove all the chairs from the image. In the generated version of the image, it can be seen that when the chairs are removed, visual remnants remain mixed with the space. Disentanglement of latent vector directions remains a core challenge for this family of methods. Photo from the LSUN dataset [Yu et al. 2015].

First, our method requires an optimization be run after each edit, which takes about 30 seconds on a modern GPU. This time can be reduced by applying fewer optimization steps and incrementally optimizing during editing, or by using a full-image adaptation G'_w of the generator that provides a adaptation of the generator that does not depend on the editing region. Although these faster methods introduce more artifacts than our full method, they can provide useful preview of the results at interactive speeds.

Second, another challenge is that latent spaces learned by deep neural networks are not fully disentangled, so finding vector operations to express a user intent can remain challenging. Adding or removing an object from a scene may have some interactions with unrelated objects. Some examples of undesirable interaction are illustrated in Figure 11; in these examples, chairs are reduced by zeroing chair-correlated components in the representation, but the rendered results do not cleanly remove all parts of the chairs, and distorted parts that resemble chair legs remain in the result. We also note that our method for varying the appearances of objects can be brittle; certain classes of objects (such as trees) vary more than other classes (such as domes or skies). For example, in Figure (10) we found that dome shape, but not color, could be varied.

Lastly, the quality and resolution of our current results are still limited by the deep generator that we are using [Karras et al. 2018]. For example, in Figure 7, the GAN-synthesized images omit many details such as cars and signage; our method cannot add such objects to an image if they are not modeled by the generator. But our method is not coupled with a specific generative model and will improve with the advancement of models with better quality and resolution (e.g., concurrent work [Brock et al. 2019; Karras et al. 2019]).

Nevertheless, we have found that in many computer graphics applications the learned natural image prior can help produce more realistic results with less user input. Our work presents a small step towards leveraging the knowledge learned by a deep generative model for semantic photo manipulation tasks.

ACKNOWLEDGMENTS

We are grateful for the support of the MIT-IBM Watson AI Lab, the DARPA XAI program FA8750-18-C000, NSF 1524817 on Advancing Visual Recognition with Feature Visualizations, and a hardware donation from NVIDIA.

REFERENCES

Xiaobo An and Fabio Pellacini. 2008. AppProp: all-pairs appearance-space edit propagation. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 40.

Shai Avidan and Ariel Shamir. 2007. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, Vol. 26. ACM, 10.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG)* 28, 3 (2009), 24.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Zhou Bolei, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2019. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. In *ICLR*.

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large scale gan training for high fidelity natural image synthesis. (2019).

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2017. Neural photo editing with introspective adversarial networks. In *ICLR*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*.

Alexey Dosovitskiy and Thomas Brox. 2016. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*.

Frédo Durand and Julie Dorsey. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *ACM transactions on graphics (TOG)*, Vol. 21. ACM, 257–266.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. *CVPR* (2016).

Jiahao Geng, Tianjia Shao, Youyi Zheng, Yanlin Weng, and Kun Zhou. 2018. Warp-guided GANs for single-photo facial animation. In *SIGGRAPH Asia*. 231.

Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. 2017. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 118.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM TOG* 35, 4 (2016).

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 107.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *CVPR*.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*.

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *CVPR*.

Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. 2011. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG)* 30, 6 (2011), 157.

Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 2014. 3D object manipulation in a single photograph using stock 3D models. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 127.

Tae-Hoon Kim and Sang Il Park. 2018. Deep context-aware descreening and rescreening of halftone images. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 48.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR* (2014).

Jean-François Lalonde, Derek Hoiem, Alexei A Efros, Carsten Rother, John Winn, and Antonio Criminisi. 2007. Photo clip art. *ACM transactions on graphics (TOG)* 26, 3 (2007), 3.

Anat Levin, Dani Lischinski, and Yair Weiss. 2004. Colorization using optimization. In *ACM transactions on graphics (tog)*, Vol. 23. ACM, 689–694.

Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. 2018. A closed-form solution to photorealistic image stylization. In *ECCV*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, Hao Li, Richard Roberts, and others. 2018. paGAN: real-time avatars using dynamic textures. In *SIGGRAPH Asia*. 258.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic Image Synthesis with Spatially-Adaptive Normalization. In *CVPR*.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context Encoders: Feature Learning by Inpainting. *CVPR* (2016).

Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. 2016. Invertible conditional gans for image editing. In *NIPS Workshop on Adversarial Training*.

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Transactions on graphics (TOG)* 22, 3 (2003), 313–318.

Tiziano Portenier, Qiyang Hu, Attila Szabó, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. 2018. Faceshop: Deep Sketch-based Face Image Editing. *ACM Transactions on Graphics (TOG)* 37, 4 (July 2018), 99:1–99:13.

Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. 2001. Color transfer between images. *IEEE Computer graphics and applications* 21, 5 (2001), 34–41.

Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. In *CVPR*.

Assaf Shocher, Nadav Cohen, and Michal Irani. 2018. âĂIJZero-ShotâĂİ Super-Resolution using Deep Internal Learning. In *CVPR*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Michael W Tao, Micah K Johnson, and Sylvain Paris. 2010. Error-tolerant image compositing. In *ECCV*.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2018. Deep image prior. In *CVPR*.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *CVPR*.

Su Xue, Aseem Agarwala, Julie Dorsey, and Holly Rushmeier. 2012. Understanding and improving the realism of image composites. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 84.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365* (2015).

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. 2018. Generative Image Inpainting With Contextual Attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Edward Zhang, Michael F Cohen, and Brian Curless. 2016a. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 174.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. 2017a. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *ICCV*.

Richard Zhang, Phillip Isola, and Alexei A Efros. 2016b. Colorful Image Colorization. In *ECCV*.

Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. 2017b. Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Transactions on Graphics (TOG)* 9, 4 (2017).

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative Visual Manipulation on the Natural Image Manifold. In *ECCV*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *ICCV*.