Sparse, Smart Contours to Represent and Edit Images

Tali Dekel Chuang Gan Dilip Krishnan Ce Liu William T. Freeman
Google Research
{tdekel,chuangg,dilipkay,celiu,wfreeman}@google.com

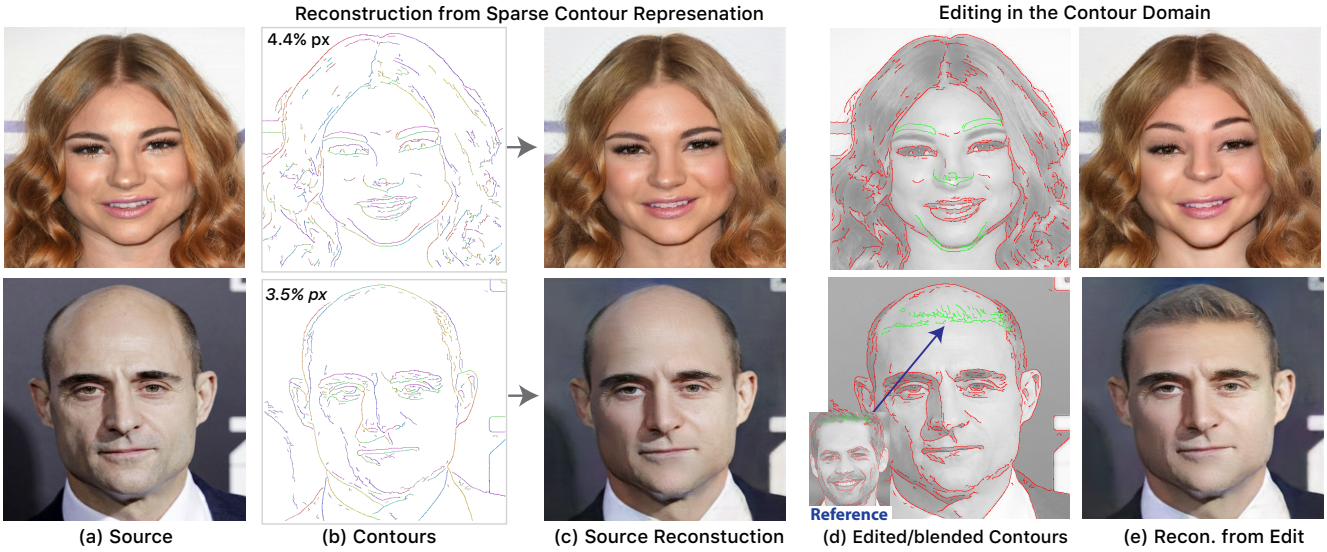


Figure 1. Our method produces high quality reconstructions of images from input representations in the form of values at sparse contour locations: a source (512×512) image in (a) is reconstructed in (c) from gradient information stored at the set of colored contours in (b)[2]. Less than 5% of pixels are non-zero. The model synthesizes hair texture, facial lines and shading even in regions where no input information is provided. Our model allows for semantically intuitive editing in the contour domain. Top-right: a caricature-like result (e) is created by moving and scaling some contours in (d). Bottom-right: hairs are synthesized by pasting a set of hair contours copied from a reference image. Edited contours are marked in green while the original contours in red.

Abstract

We study the problem of reconstructing an image from information stored at sparse contour locations. We show that high-quality reconstructions with high fidelity to the source image can be obtained from sparse input, e.g., comprising less than 6% of image pixels. This is a significant improvement over existing contour-based reconstruction methods that require much denser input to capture subtle texture information and to ensure image quality. Our model, based on generative adversarial networks, synthesizes texture and details in regions where no input information is provided. The semantic knowledge encoded into our model and the sparsity of the input allows to use contours as an intuitive interface for semantically-aware image manipulation: local edits in contour domain translate to long-range and coherent changes in pixel space. We can perform complex structural changes such as changing facial expression by simple edits of contours such as scaling and moving. Experiments on a variety of datasets verify the versatility and convenience of our models.

1. Introduction

Contours are a concise and perceptually meaningful representation of the image, as they encode "things" and not "stuff" [1]. This makes them appealing for image reconstruction and manipulation. As contours capture shape and object's boundaries, it is desirable to be able to maneuver them (e.g. translating, scaling, copying and pasting) and have the related pixels adapt accordingly, such that the edited images preserve the original image structure and texture details; just as artists use simple sketches to guide drawing sophisticated paintings. Therefore, faithfully reconstructing images from sparse contours, an open question that dates back to the seminal work of David Marr [24], is of great interest as it is the foundation for editing and processing.

A binary contour map is often insufficient to preserve fidelity for reconstruction (e.g. [18], Fig. 2(c)). Therefore, local image information such as gradients or color have been combined with contour locations and has been stud-

[2]First two principal components of the features is mapped to RGB [4].

ied extensively in the literature on diffusion-based methods [12, 13].

However, such diffusion based methods are not applicable for image editing because of their inability to synthesize texture and missing content. High-quality reconstruction often requires dense contours, which precisely forfeits the original purpose of conciseness and ease of manipulation (see Fig. 2(d-e)). When the contours are sparse, the reconstruction loses important image details such as texture (see Fig. 2(a-b)).

In this paper, we propose a new method, based on deep generative models, to resolve the conflict between high *fidelity* and high *sparsity*. Instead of forcing contours to model textures, details and fine structures, our model learns to hallucinate it appropriately, just from a sparse contour representation, even in large regions where no input information is provided (see Fig. 1(a-c)). Specifically, we assume that the correlation between contours and textures is well encapsulated in a class of images, such as faces, dogs and birds. For instance, knowing that a contour map is of a person's face, our model can fill in the details of hairs and facial expressions based on the statistical correlation trained on a set of facial images. To this end, we develop a cascade of two networks, splitting the overall task into two more tractable problems. The first network reconstructs the overall image structures and colors, while the second network recovers texture and fine details.

Extensive experiments show that with our model, high fidelity image reconstruction can be obtained from information stored at a small fraction of contour pixels, as small as 3% for a 512×512 image (see Fig. 1(a-c)). This essentially makes contours a powerful tool for image editing. Furthermore, our results demonstrate that our models encode semantic information about the training data. Hence, local edits in the contour domain are translated into coherent changes in the pixel space (e.g., dragging the eyebrows of a person up leads to changes in the facial lines that connect the eyebrow to the nose, see Fig. 1(d-e)). We show various image editing examples such as creating caricatures, changing facial expression or generating hair or fur texture.

2. Related Work

We briefly survey relevant image editing and reconstruction literature. Elder [12, 13] explored the completeness of contour representations, and their use in image editing tasks using diffusion based methods. Diffusion curves are vector-based primitives that have been suggested for creating smooth, shaded images [27]. Similar representations have been explored for compression of piece-wise smooth images such as depth maps or cartoon images [22].

In contrast to the above mentioned methods, a number of exemplar-based approaches for image editing have been proposed. Prominent among these are patch-based methods [5, 34] and seam carving [3]. These techniques copy patch information to create high-quality edited images. However,

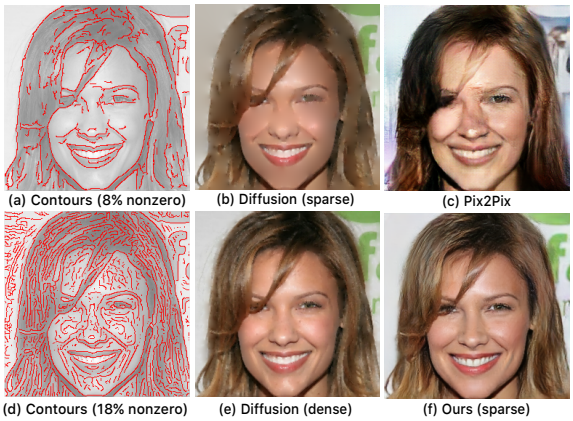


Figure 2. Reconstruction from sparse contours (marked in red in (a)) obtained by: (b) Homogeneous diffusion where the input comprises of RGB values sampled at both sides of each edge location; (c) Pix2pix [18] that takes a binary edge map as input; and (f) our result using gradient information stored at red locations in (a). The source image is shown in Fig. 3. (d-e) dense contours and corresponding reconstruction obtained by homogeneous diffusion. Our approach gives significantly superior reconstruction compared to (b-c) and (e) which is supplied with much denser input.

they are oblivious to the semantic content of the image, often failing to produce large, coherent changes. Edits often require human intervention in the form of geometric constraints.

Deep neural network based image synthesis approaches provide an alternative approach for image editing tasks [16, 14, 11, 29, 41]. Many of these works rely on autoencoder architectures and pixel losses, which have difficulty reconstructing textures. Superior results are possible with the use of generative adversarial networks (GANs) [15].

Unconditional GANs synthesize images from a stochastic latent variable. However, fine user control over the synthesized image is problematic. Several methods (e.g., [43, 7]) attempt to address this by performing image editing through the low-dimensional learned latent space of the generator network in a GAN. The idea is to optimize the latent representation of an image to satisfy user constrains (e.g., shape or color), while not deviating much from the latent representation of the original image. This approach requires solving a complicated optimization problem by back-propagation which is slow. User constraints are taken into account *implicitly* and so control over the generated image is limited. Finally, although various methods are being developed to stabilize GAN training [23, 2], synthesizing natural scenes from stochastic inputs is an open problem.

To combat this, methods have been proposed to condition the GAN on other kinds of inputs [26, 21, 17, 29, 25, 18, 39, 32, 40, 39]. Isola *et al.*[18] synthesize images from input label maps or edge maps. They consider only *binary* edges, which leads to low fidelity to the original image. Furthermore, they did not consider the task of image editing but rather focused on image translation from one domain to

another. Sangkloy *et al.*[32] took a step towards more *controllable synthesis* by learning to generate images from input hand-drawn sketches and additional input sparse color strokes. However, their input sketch is much denser than what we consider in this paper, and hence not suitable for complex geometric manipulation. In addition, because of the density of the input, their network is not required to synthesize texture in large regions as in our case. Furthermore, their edits consist of color changes, while leaving contours fixed, unlike our edits that modify the contour structure.

3. Overview

We represent an image by a sparse set of contours (computed using an off-the-shelf edge detector [10]), and an N-dimensional feature for each of the contour pixels. In this paper, we have experimented with three types of features: gradients, color and learned features (described further in Sec. 4). We reconstruct the source image from this input representation using a cascade of two networks, illustrated in Fig. 3.

The sparse contour representation is first fed into a network driven by an L_1 pixel loss that reconstructs the overall structure and colors of the output image (Fig. 3(a)). For example, when training on the VGG face dataset [28], this network recovers the face shape, skin tone, hair color and overall shading. We abuse notation slightly and call this network LFN ("Low Frequency Network"), although the reconstruction does contain some high frequency information, given by the input contours.

The second network takes as input the reconstruction produced by the LFN as well as the original sparse input, and outputs a much more textured and detailed reconstruction of the original image, using an adversarial loss; we call this HFN ("High Frequency Network"). Since we work with sparse contours ($\sim 6\%$ or less of total image pixels), significant textured regions are not represented in the input. The HFN learns to synthesize plausible texture and structure in these regions. For example, in the case of faces, the hair texture and fine facial lines are synthesized by the HFN, even though very few contours are detected in these regions.

We now describe the input representation and the model in detail.

4. Sparse Contour Representation

Given a contour map, an important consideration is what information to encode at each contour position. Color and gradients are common choices among diffusion-based methods (e.g., [27, 22]), and gradients have been a useful representation for image editing [30]. We therefore consider these two options, as well as a learned feature representation trained end-to-end with our reconstruction network. We define the feature $f(p)$ at each detected contour point p as follows:

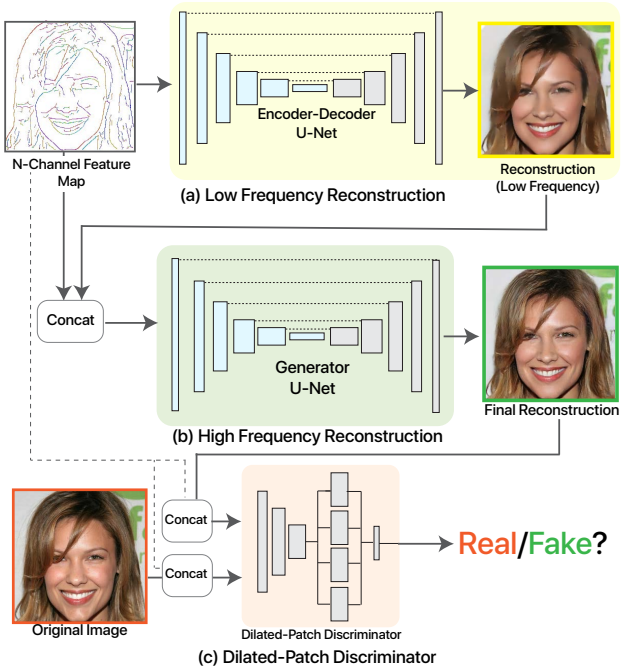


Figure 3. Our model reconstructs an image from a sparse N-channel feature map (typically N=3 or N=6) using a cascade of two "U-Nets" [31]: (a) a Low-Frequency Network (LFN), trained with an L_1 pixel loss, that recovers the overall structure and colors of the image; (b) High-Frequency Network (HFN) that is conditioned on the LFN output and the input feature map, produces a textured and detailed reconstruction; the HFN is trained with a combination of pixel loss and adversarial loss. (c) Our conditional discriminator, which incorporates dilated convolutions and aggregation across image patches to better capture high frequencies. "Concat" refers to concatenating channels of the same spatial resolution along the depth axis.

1. Color: The orientation of the contour map is computed and *R,G,B* values sampled at both sides of the contour for each edge pixel. That is, $f(p) = \{I_d^c, I_b^c\}_{c\in\{R,G,B\}}$, where I_d^c and I_b^c are values on either side of a contour for channel c. This results in a 6-value code per contour location.

2. Gradients: At each contour point, spatial image derivatives are computed for each of the color channels: $f(p) = \{G_x^c, G_y^c\}_{c\in\{R,G,B\}}$, where G_x^c, G_y^c are the x and y derivatives of the image channel c. This also results in a 6-value code per contour pixel.

3. Learned Features: An N channel feature map is learned end-to-end while training the reconstruction network (we found $N = 3$ to be a good balance between quality and complexity). We use a multi-scale representation to encode information from a larger neighbourhood around each edge pixel. We use a simple network that consists of a convolutional layer followed by a branch of *dilated convolution* filters with different sampling rates, employing an architecture similar to átrous spatial pyramid that presented in [8]. See Sec. 8 for more details.

Of the three choices above, we found that multi-scale

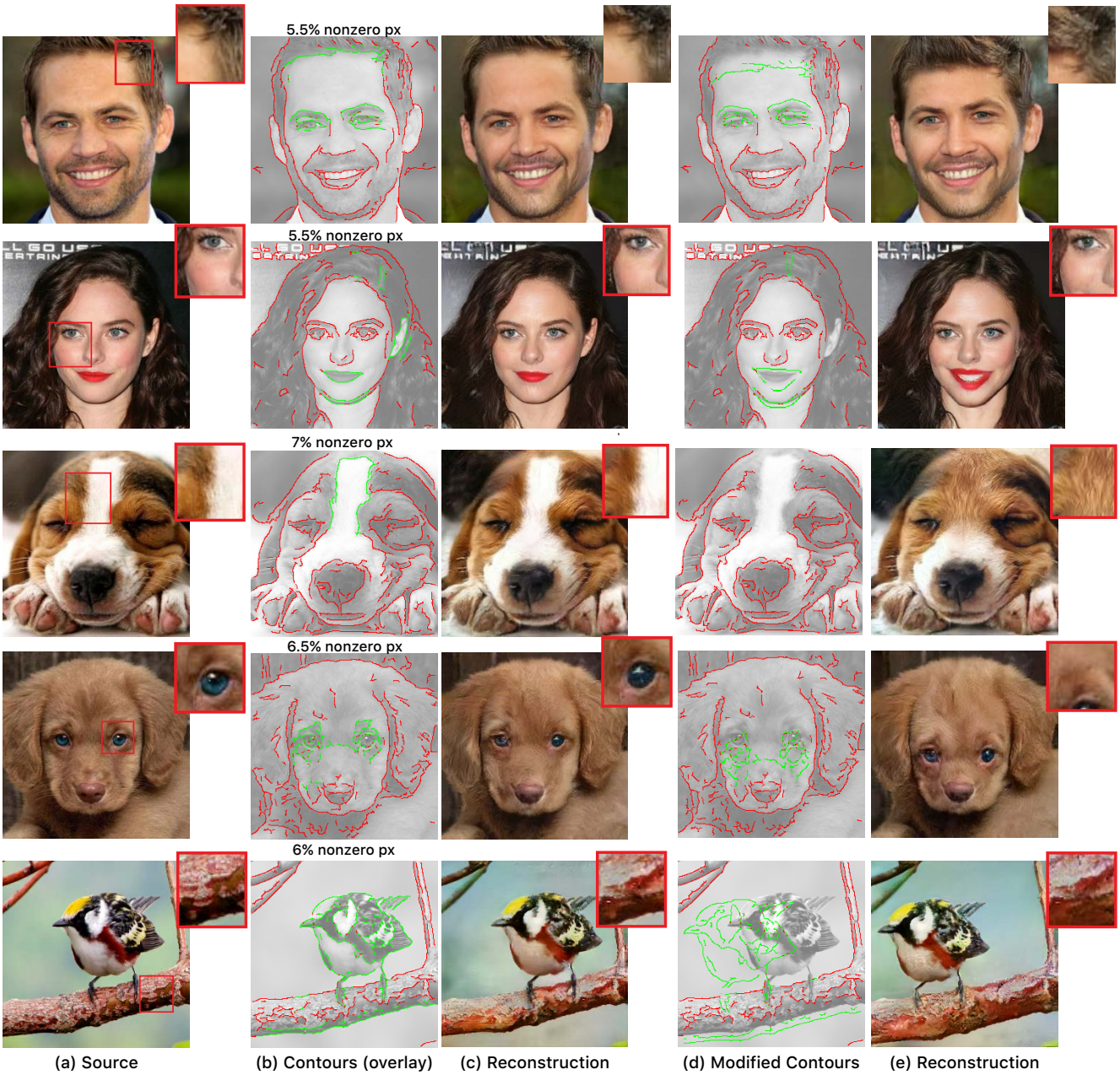


Figure 4. Reconstruction examples from sparse contour representation and manipulation results in the contour domain: (a) The source images; (b) The input sparse contour maps colored over the source images; (c) Reconstructions output of High Frequency Network (HFN) using sparse gradient information stored at colored (both red and green) contours (b); (d) Edited contours (the green colored contours in (b) are modified, red contours remain the same); (e) HFN reconstruction from edited contours. Notice how the edited results in are semantically meaningful and textures and details are reconstructed in regions with missing data.

learned features result in improved quality of reconstruction (see Fig. 5). However, for the application of image editing, we found gradient features to have the best trade-off between reconstruction accuracy of the original image and quality of image edits. Although gradient features are the most challenging to invert (the network needs to recover the absolute color values), they provide greater flexibility and robustness to image manipulation. For example, the use of gradients allows to blend two sets of contours taken from different images, as shown in Fig. 1. This aligns with the literature on image editing in the gradient domain, e.g. [30, 6].

Color features are more restrictive since they impose constraints on the absolute colors of the reconstruction. Learned features encode multi-scale information, hence the representation of one pixel can be highly correlated with

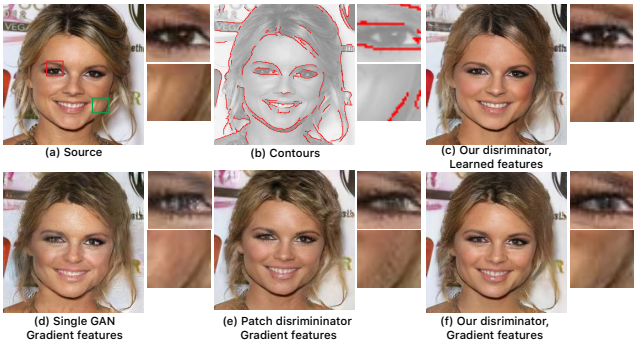


Figure 5. A source image (a) is reconstructed using information stored at 5% of edge pixels (b) with different network configurations. (c): Reconstruction using our cascaded model trained with our dilated-patch discriminator and end-to-end learned features. Second row: comparison using input gradient features with different models. (d): A single (non-cascaded) GAN. (e): Our cascaded network trained with patch discriminator [18]. (f): With dilated-patch discriminator. While the learned features are better for reconstruction, they are too restrictive for image editing (see discussion in Sec. 4).

the representation of another. Such correlations between features reduces flexibility to image edits (see Fig. 9(a)). Designing learned features which have the beneficial properties of gradients for image editing, while allowing higher-quality reconstructions, is an interesting area for future research.

5. Model

As mentioned in Sec. 3 and shown in Fig. 3, our model consists of a cascade of two networks: the first network (LFN) reconstructs the overall structure and colors of the output image from the sparse feature map, whereas the second one (HFN) recovers fine details and texture given the blurry (piecewise smooth) output of the LFN and the original sparse input. This is driven by the training losses. The LFN is trained with an L_1 pixel loss between the reconstructed output image and the ground-truth image, which encourages the overall structure and low-frequencies of the output to match the input but is insufficient to reconstruct fine textures and higher frequencies [19].

The HFN is conditioned on the sparse contours and the output of the LFN, and trained with a combination of L_1 pixel loss and an adversarial loss [15]. We use a conditional discriminator whose task is to distinguish between the *real* image, and the *fake* output of the HFN (the images are from the same source, i.e. the fake image is reconstructed from the contours of the real image). The weights of the LFN remains fixed during this training. For both the discriminator and generator adversarial losses, we use an L_2 loss between the logits of the real and generated samples, following the approach of Mao *et al.* [23].

The architecture of the LFN and the generator of HFN is a convolutional encoder and decoder with skip connections between layers of the encoder and decoder [31]. The architecture of our discriminator (Fig. 3(c)) is a combination of a "patch discriminator" [18] and a branch of dilated convolution filters that better capture higher frequencies (Fig. 5)(e-f). See SM for a detailed description.

The network cascade, our patch-dilated discriminator and U-net based decoder, together give high quality reconstructions for both 256×256 and 512×512 size images, as demonstrated in Fig. 1 and the supplementary materials. Overall performance is significantly improved over a non-cascaded single GAN (see Fig. 5), in line with the findings reported in [9, 42].

6. Experiments

We trained a model for each of the three publicly available datasets from different domains: VGG Faces [28], Caltech-UCSD Birds [38] and Stanford Dogs [20]. See Sec. 8 for more details. We performed the following experiments to measure the quality of our reconstructions and editing.

Reconstruction: Fig. 1, Fig. 2, Fig. 4(a-c), Fig. 6 and Fig. 8 show reconstruction examples that we achieve on these datasets, using gradient features at each contour location. Our models produce high quality reconstructions with respect to the original images from very sparse inputs (4%-7% non zero pixels). They reliably recover long range information and details that were not presented as input. For faces, for example, our models synthesize hair texture and recover fine details in the eyes, teeth and facial lines. In Fig. 8, we can see that texture of the foreground object (fur of a dog, feathers of a bird) is synthesized as well as the texture of the background such as ripples of water, grass, or wood and tree texture.

Editing: We show a number of editing results in Fig. 1 and Fig. 4(d-e), where the green colored contours in Fig. 4(b) are manipulated. Fig. 1 top row shows how we can easily create a caricature of a person by moving and scaling sets of contours (with gradient information being transferred as edges are moved around). In this example, the woman's eyebrows and nose are moved and the shape of her jaw is changed. The bottom row shows an example of generating plausible hair texture by blending in the contour domain: a set of "hair contours" (and their underlying gradient information) is copied from a reference image onto the target image. Our network convincingly inpaints the region that was originally bald. This effect of hair synthesis is also demonstrated in the second row in Fig. 4 where contours at the forehead boundary were dragged down.

The first row of Fig. 4 shows the creation of a smile effect by moving and scaling edges. Note the fine facial lines that are generated to accommodate the smile. Additional edits in this example include moving the hairline to the center of the head, and inpainting the ear region with hair by removing the ear contours. In the third example and fourth rows, from

Figure 6. The source image (a) is reconstructed from different sparse representations kept at the same edge pixels marked in red (b) using the following methods: (c) Diffusion [13] based solution that propagates RGB values sampled at both sides of the contour; (d) Pix2pix [18] which uses only binary contours as input; (e) Our LFN output using sparse gradients and (f) our final HFN output.

the Stanford Dogs dataset, we remove a portion of the white marking on the dog's forehead (third row); and move the position of the eyes (fourth row) to give the dog puppy-like proportions. In both cases, convincing fur-like texture is reconstructed, as can be seen from zoomed in portions.

In the bird example (last row of Fig. 4), we can make the tree trunk thinner by simply shifting up the contours that corresponds to the lower part of the tree trunk. In addition, this example demonstrates how easy it is to relocate the bird in the scene by pasting its contours in the new location, avoiding the need to accurately segment it from the background. The network is robust to missing edges and can reliably inpaint the holes that are generated in the original location of the bird.

These editing examples also show the necessity of a sparse contour representation to perform image edits. It would be more difficult to achieve these effects with denser contours (e.g., Fig. 2(d)). Furthermore, the use of a sparse representation has encouraged the network to learn semantic interpretations of a scene, giving it the ability to synthesize plausible texture and structure. Finally, in Fig. 9(a) we show one example of the effect of editing using learned features. This performs poorer than gradient features (compare to Fig. 4).

Model Components and Input Features We evaluate the performance our method using different types of input features as described Sec. 4. Specifically, we computed the average SSIM [37] between the real images, and the reconstructions, on 100 randomly sampled images from the VGG, CUB and Dogs datasets. Table 2 shows the computed scores using either gradient features, color (RGB) or learned features. The learned features, which capture multi-scale information, consistently give the best results, followed by color and then gradients. Note that in the case of gradients the network needs to recover the unknown absolute color values. This is clearly an ill-posed problem because adding any global constant to the true color values results in the same image gradients. Therefore, the reconstructions with gradients sometimes results in slight color shifts w.r.t. original image.

In Table 2 we also show the SSIM scores for gradient features using a single GAN instead of our cascaded network. The benefits of a cascade are clearly seen in the improved SSIM scores. This is unsurprising as the number of parameters are significantly larger and a coarse-to-fine approach has been shown to improve results in other works [42, 9].

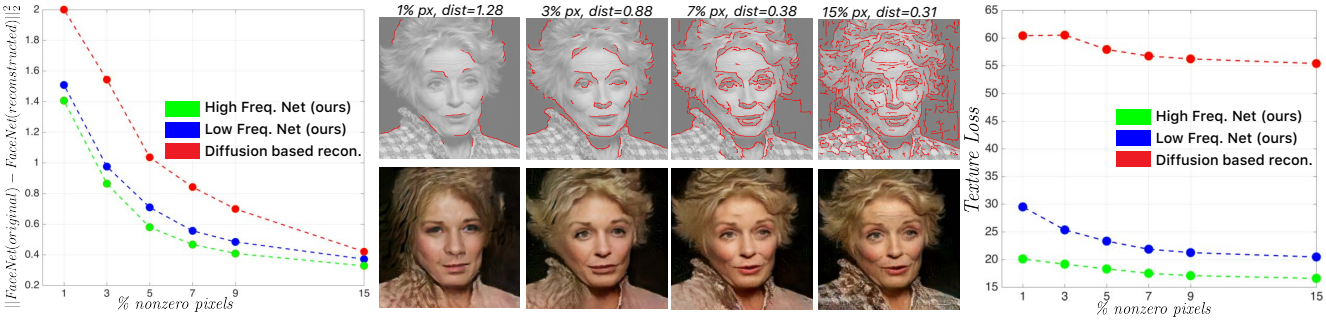
We also compare our final cascaded network output to the output of the LFN, and to that of a single (non-cascaded GAN). For all approaches, we use the same contour input,

Dataset	%Turkers Labeled Real	
	1 second	5 seconds
VGG 256x256 [28]	49.3	44.7
VGG 512x512 [28]	47.2	43.5
Stanford Dogs 256x256 [20]	48.1	46.1
CUB Birds 256x256 [38]	49.9	45.8

Table 1. AMT "real vs fake" test on different datasets. We show what fraction of the generated results were considered real by the workers, when the pairs were presented for 1 second or 5 seconds.

Dataset	GAN	LFN-HFN		
		Gradients	Color	Learned
VGG	0.786	0.812	0.859	0.867
CUB	0.697	0.740	0.763	0.783
Dogs	0.722	0.749	0.792	0.810

Table 2. Average SSIM scores on 100 randomly sampled images from the VGG test set. From left to right: reconstruction using a single GAN with gradients; and reconstruction using our LFN-HFN cascade network trained with gradients, RGB, and learned features, respectively. The input edge maps are identical for all network and have approximately 6% non-zero contour pixels.



(a) Average FaceNet Embedding Dist. (b) Our final reconstuction as a function of edge sparsity (c) Average Texture loss

Figure 7. Quantitative evaluation. We used FaceNet [33] to evaluate how close our reconstructions are to the original images in terms of recognition. (a): The computed distances averaged L_2 distances between the FaceNet embedding of the original image and our reconstruction, as a function of input contour sparsity, for the result of LFN, HFN and homogeneous diffusion (lower is better). A typical threshold for same/not same classification is $\sim$ 1.2 [33]. (b): The output of HFN for different sparsity levels (with distances shown above each image). (c): Texture loss [14] as a function of sparsity. HFN has the lowest texture loss compared to LFN or diffusion. For the images in (b), their texture losses are very similar although the face shape changes based on contours.

but the information at each contour location differs. A qualitative example of reconstruction using different network configurations in shown in Fig. 5.

Comparison with Baselines Fig. 2 and Fig. 6 show qualitative comparisons to two baselines: (i) a homogeneous diffusion approach which is a classic low-level method for image reconstruction from sparse contour representation, (see Sec. 2) and (ii) Pix2Pix [18].

For diffusion, we follow the approach of Elder [13], and use color (RGB) values sampled at either side of a contour location. It is seen that this results in piecewise smooth reconstructions, and fails to recover texture or details at missing contours. This is not surprising since diffusion merely interpolates the color values, without any semantic knowledge. The diffusion is sensitive to the location of the constrains and the sampling of the input values. Therefore, even when supplied with very dense information as in Fig. 2(c), the reconstruction (d) suffers from spikes and blurriness. This can also be seen be from the comparison to our low-frequency reconstruction that recovers significantly better the highlights and shading of the face (Fig. 6(f)).

For Pix2pix [18] that takes as input *binary* edge maps, we trained their network from scratch (using their original PyTorch implementation) on exactly the same data (input images and edge maps) that was used to train our network on the VGG and Caltech-UCSD datasets. Pix2pix, while recovering some texture, fails to recover the scene properties (e.g. skin tone, hair color, bird color) and results in poor quality reconstructions (e.g., tree texture in the background of the the bird image or facial artifacts). This is due to the lack of information at the contours, and demonstrates the importance of both location and value information.

Fig. 9 shows a comparison to Photoshop's state-of-the-art editing tools, for similar edits as Fig. 1 and Fig. 4. Since these tools do not have semantic awareness, the results can often be less than satisfactory.

6.1. Quantitative Evaluation

We performed a number of experiments to quantitative evaluate the quality of our reconstructions. Because our model hallucinates high frequency content that may not match exactly the original one, we go beyond measuring standard image similarity measures such as PSNR or SSIM that do not necessarily capture the perceptual quality [19].

Human Evaluation We evaluated our results using human raters on Amazon Mechanical Turk (AMT), following closely the protocol in [18]. Workers were presented with pairs of images corresponding to the source image and our reconstruction, and asked to label which one was "real". Each pair of images was presented for a limited time after which the rater makes their choice. As practice, the first 10 pairs of images were shown without a time limit. We evaluated the ranking of 10 rater, each was given 100 image pairs. Our reconstructions for this test were obtained using gradient based-features at 6% contour locations. The same test was repeated for 1 and 5 seconds presentation time. The results, reported in Table 1, show that for all datasets, our reconstructed images were hard to distinguish from the real images (a score of 50% would mean perfect confusion between real and fake). As expected. for a 5 second presentation time, it was slightly easier to spot reconstructions.

Face Recognition Evaluation We tested the extent to which our reconstructed faces capture the identity of a person. We used FaceNet [33], a well-known face recognition system and measured whether our reconstruction (using different sparsity levels of input) and the original image are classified as the same person. We followed [33] and computed the squared L_2-distance between the 128-dimensional embedding vectors of the original image and our reconstruction. Fig. 7(a) shows the average distance over 50 images randomly sampled from the VGG test set when applying our network using gradient input features (as this is the most challenging case for reconstruction) at different

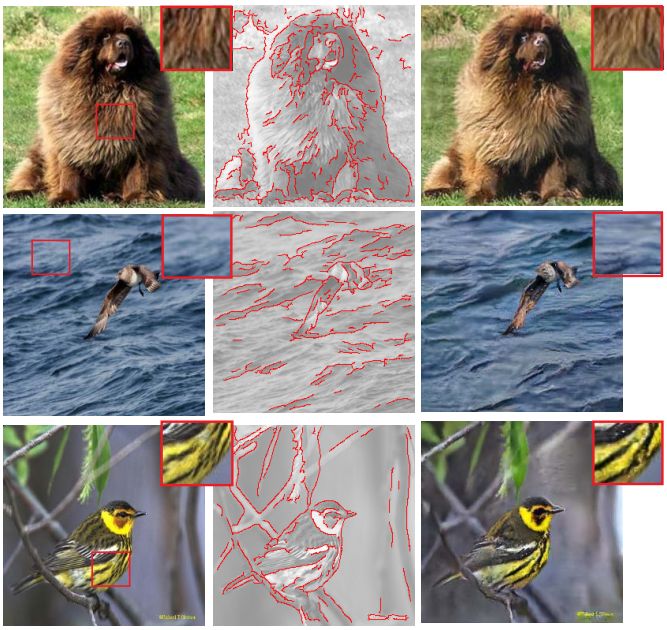


Figure 8. Reconstruction examples of test images from Stanford dogs and Caltech-UCSD datasets. Our network learns to synthesize the textures of the objects e.g., the fur of a dog, as well as the natural backgrounds in the scenes such as water and grass.

contour density levels. We report the error for LFN, HFN and homogenous diffusion. As expected, the performance of HFN is the highest regardless of the contour's sparsity, on average a relative 10% reduction in error over LFN and nearly 40% over homogenous diffusion. This shows that reconstructing details is helpful for a face recognition system.

An example of our reconstructions and the corresponding edge maps are shown in Fig. 7(b), where the resemblance to the source image gradually increase with density. There is not much loss of information between 12% and 6% nonzero pixels because the network recovers missing high frequency information. Note that even at a sparsity as low as 4% of pixels, Facenet recognizes the resulting face as being the same as the original (based on the thresholds given in [33]).

Texture/Style Evaluation: Recent style transfer methods have demonstrated that texture statistics can be captured by the Gram matrix of the activations at some layers in a pretrained recognition network (e.g. [35]). The work of [14] defines a *texture-loss* between two images as $\sum_{l=0}^{L} w_l \sum_{i,j}(G^l_{ij} - \hat{G}^l_{ij})^2$, where l network layers, and G^l_{ij} is the inner product (Gram matrix) between the vectorized feature map i and j at layer l: $G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$. G and $\hat{G}$ refer to the Gram matrices for two different images. We use this loss (using the same layers and network as [19]) to evaluate the quality of our synthesized texture compared to the source image: a lower loss means that the synthesized textures are closer to the source image. Fig. 7(c) shows the computed texture loss using LFN, HFN and diffusion based reconstructions, as a function of the contour's sparsity. The benefit of a GAN loss in reconstructing texture is clearly seen from the plots, as the loss of HFN is consistency the lowest and steady over different sparsity levels.

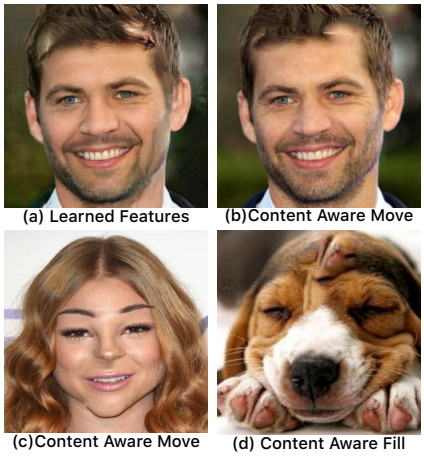


Figure 9. (a) Editing example using our model with learned features. (b-d) Comparison to patch-based Photoshop editing tools: content aware move and content aware fill. Our results using gradients on these images for shown in Fig. 1(e) and first and third row in Fig. 4)(e).

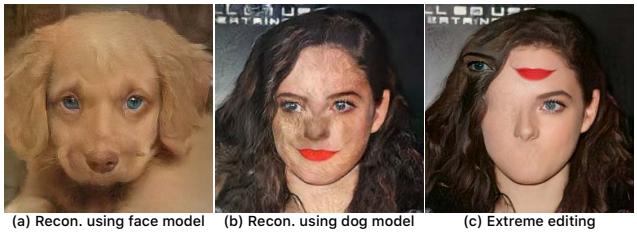


Figure 10. Limitations. (a) A *dog* image is reconstructed using our model when trained of *faces*; (b) shows the reverse. (c) Effect of an extreme edit whereby the result is not semantically meaningful due to the contour constraints.

7. Conclusions

We have presented a deep network model that produces high-quality reconstructions of images, and effective semantically-aware editing, from sparse contour representations. Due to the sparsity and the high level information encoded in our model, the representation has the significant benefit of being easy to manipulate for large, coherent edits. This is a significant improvement over existing work.

Our model is limited by the fact that domain specific textures and details do not transfer well between one domain to another. For example, applying a model trained on the dogs to a face results in a dog-like appearance (Fig. 10(b)), and vice-versa (Fig. 10(a)). In both cases, the input contours provides a strong constraint on the reconstruction, but the texture synthesized by our model is dominated by the training data. In some cases (e.g., extreme editing operations) can prevent semantically meaningful reconstructions (Fig. 10(c)).

8. Implementation Details

Our models were implemented in Tensorflow. We will release our trained models and code. To achieve the editing effects, we built a simple graphical user interface (GUI) that allows editing the contour map using simple operations such as moving, scaling or erasing sets of contours. Note that although the user edits the contour map, the underlying features change in the appropriate manner (for example, if some contours are removed, then the features associated with those contours are also removed).

Computing Input Representation: Edge probability maps were extracted using [10] followed by non-maximum suppression as a post processing. The computed maps were binarized by keeping x percentile of edges, where x is the desired percentage of nonzero pixels in the binary edge map. We group the edges into contours and filter out short contours (length less then 10 pixels). Gradients are computed by simple forward differences.

LFN and HFN: Both LFN and HFN are "U-Nets" [18]. We adopt the notations in [18] and denote a convolution layer with k filters followed by batch normalization by `Ck` for ReLU activation, and `Clk` for LeakyReLU (slop 0.2). For 256×256 images the number of filters in the encoder is given by: `C164-C1128-C1256-C1512-C1512-C1512`, hence the spatial resolution of the bottleneck is 4×4. The number of filters in each layer of the decoder is given by: `C512-C512-C256-C128-C64-C64` (filters in decoder are bigger because of the skip connections). All convolutional filters are size 4×4, and we use strides of size 2.

Dilated-Patch Discriminator Our discriminator (Fig. 3(c)) is a combination of a "patch discriminator" [18] and a branch of dilated convolution filters. Let $D_r k$ denote a dilated convolution with sampling rate r, leaky ReLU and k filters. Then our patch discriminator architecture is given by `C164-C1128-C1256` followed by 4 parallel dilated convolutions $\{\mathtt{D_2256}, \mathtt{D_4256}, \mathtt{D_8256}, \mathtt{D_{12}256}\}$, which are concatenated, and a final convolution layer with a single channel output.

The network used for learning the input representation (see Sec. 4) is similar and consists of a branch of dilated convolutions, each followed by regular a convolution layer. That is, $\{\mathtt{D_432-C3}, \mathtt{D_8256-C3}, \mathtt{D_{12}256-C3}, \mathtt{D_{16}256-C3}\}$, which are added to form the final output.

Training hyper-parameters and details For each dataset, LFN and HFN were trained from scratch (weights were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02). We used Adam optimizer with $\beta = 0.5, \epsilon = 1e^{-4}$, and batch size of 16 in all training runs except when training on 512×512 where we used size of 8. We used learning rate of 0.0002 for the generator, and 0.00002 for the discriminator, with decay rate of 0.98 every 10000 steps. During training we resized the images such that the small dimension is at the desired resolution and then randomly cropped to desired size. The relative weight of the adversarial loss vs. reconstruction loss was 100 in all our experiments.

During the HFN training, the weights of the LFN remained fixed, and we alternate between stepping the discriminator and generator in ratio of 2:1 in favor of the discriminator. When working with learned features, the weights feature network remained fixed during HFN training.

We trained on the VGG dataset at two spatial resolutions: 256×256 and 512×512. For Birds and Dogs, we used the original train/test splits, and for the VGG dataset we filtered out low resolution images from the train and test sets, to avoid reconstruction of JPEG artifacts. The VGG has 30227 samples and was trained for 210 epochs for both LFN and HFN; CUB Birds has 8855 samples and was trained for 720 epochs; Dogs dataset has 12000 training samples and was trained for 500 epochs.

Texture Loss We use the texture loss in Section 6.1 to evaluate our reconstructions. This loss was defined in [36].

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. 1991. 1

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2

[3] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transactions on graphics (TOG)*, 2007. 2

[4] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011. 1

[5] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24–1, 2009. 2

[6] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Transactions on Graphics (TOG)*, 2010. 4

[7] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 2

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 3

[9] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015. 5, 6

[10] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 3, 9

[11] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016. 2

[12] J. H. Elder. Are edges incomplete? *International Journal of Computer Vision*, 1999. 2

[13] J. H. Elder and R. M. Goldberg. Image editing in the contour domain. *PAMI*, 2001. 2, 6, 7

[14] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 7, 8

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 5

[16] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017. 2

[17] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and Locally Consistent Image Completion. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, 2017. 2

[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. 1, 2, 5, 6, 7, 9

[19] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 5, 7, 8

[20] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 5, 6

[21] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 2

[22] M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer. Edge-based compression of cartoon-like images with homogeneous diffusion. *Pattern Recognition*, 2011. 2, 3

[23] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016. 2, 5

[24] D. Marr and A. Vision. A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*, 1982. 1

[25] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[26] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016. 2

[27] A. Orzan, A. Bousseau, P. Barla, H. Winnemöller, J. Thollot, and D. Salesin. Diffusion curves: a vector representation for smooth-shaded images. *Communications of the ACM*, 2013. 2, 3

[28] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015. 3, 5, 6

[29] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[30] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on graphics (TOG)*, 2003. 3, 4

[31] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 3, 5

[32] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. *arXiv preprint arXiv:1612.00835*, 2016. 2

[33] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 7, 8

[34] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 8

[36] I. Ustyuzhaninov, W. Brendel, L. Gatys, and M. Bethge. What does it take to generate natural textures? 2016. 9

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[38] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 5, 6

[39] W. Xian, P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Texturegan: Controlling deep image synthesis with texture patches. *arXiv preprint arXiv:1706.02823*, 2017. 2

[40] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016. 2

[41] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016. 2

[42] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*, 2016. 5, 6

[43] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2