# Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation

Arnab Ghosh[1] Richard Zhang[2] Puneet K. Dokania[1]
Oliver Wang[2] Alexei A. Efros[2,3] Philip H. S. Torr[1] Eli Shechtman[2]

[1]University of Oxford [2]Adobe Research [3]UC Berkeley

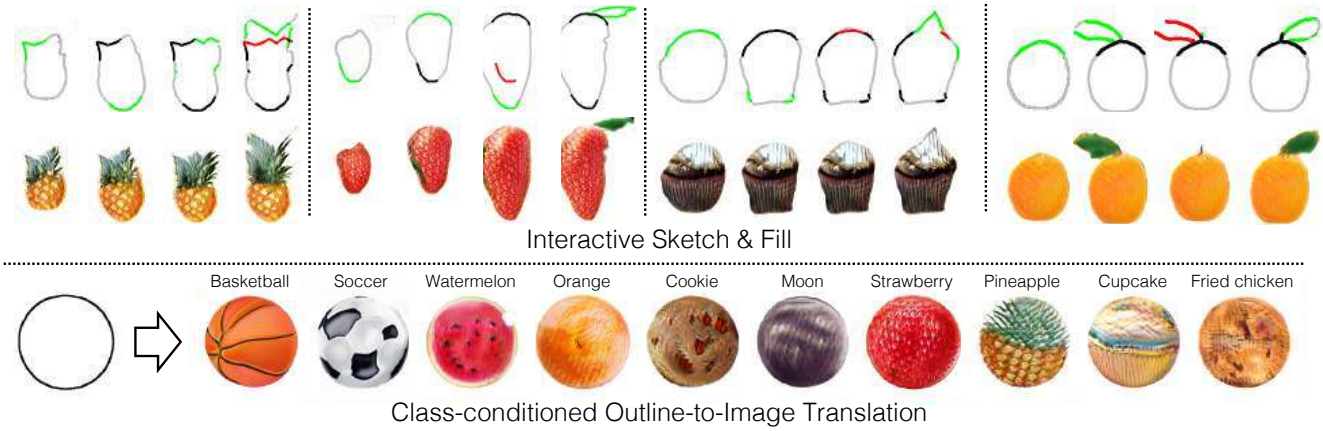



Figure 1: (**Top**) Given sparse user input (first row), our model estimates the complete shape and provides this as a recommendation to the user (shown in gray), along with the final synthesized object (second row). These estimates are updated as the user adds (green) or removes strokes (red) over time – previous edits are shown in black. (**Bottom**) This generation is class-conditioned, and our method is able to generate distinct multiple objects for the same outline (*e.g.* 'circle') by conditioning the generator on the object category.

## Abstract

*We propose an interactive GAN-based sketch-to-image translation method that helps novice users easily create images of simple objects. The user starts with a sparse sketch and a desired object category, and the network then recommends its plausible completion(s) and shows a corresponding synthesized image. This enables a feedback loop, where the user can edit the sketch based on the network's recommendations, while the network is able to better synthesize the image that the user might have in mind. In order to use a single model for a wide array of object classes, we introduce a gating-based approach for class conditioning, which allows us to generate distinct classes without feature mixing, from a single generator network.*

## 1. Introduction

Conditional GAN-based image translation [25, 43, 61] models have shown remarkable success at taking an abstract user input, such as an edge map or a semantic segmentation map, and translating it to a real image. These methods run at interactive rates, and combining them with a user interface allows the user to quickly create fun (but usually, unrealistic) images. A few limitations prevent them from being used as a true interactive tool that assists the user in generating an image of an object they have in mind. First, the user is required to provide an entire abstract map as input (full edge or label map). This may prove difficult for many, as untrained practitioners generally struggle at free-hand drawing of accurate proportions of objects and their parts [6], 3D shapes and perspective [45]. It is much easier with current image translation methods to obtain realistic looking images by editing existing images [8, 40] than creating images from scratch. Second, current GAN-based image translation methods are limited to a single class of images. For example, switching from a cat to a dog requires loading (or storing in memory) a new model per class.

We propose a new GAN-based interactive image generation system for drawing objects that: 1) generates full images given only *sparse* and *partial* user strokes (or

sketches); 2) serves as a *recommender system* that suggests or helps the user *during* their creative process, in order to generate a desired image; and 3) uses a single conditional GAN for *multiple* image classes with an effective gating mechanism. Such a system allows for creative input to come from the user, while the challenging task of getting exact object proportions correct is left to the model, which constantly predicts a plausible completion of the user's sketch (Fig. 1).

We use sparse object outlines/sketches/simplified-edges instead of dense edge maps as the user input, as these are closer to the lines that novice users tend to draw [7]. Our model first completes the input, which could be partial outlines or edges, and then generates the image conditioned on the completed shape. There are several advantages to this two-stage approach. For one, we are able to give the artist feedback on the general object shape in our interactive interface (similar to ShadowDraw [31]), allowing them to quickly refine the completed shape until it is satisfactory. Second, we found this to work better than going directly from partial outlines to images, as the additional intermediate supervision on full outlines/sketches breaks the problem into two easier sub-problems – first recover the geometric properties of the object (shape, proportions) and then fill in the appearance (colors, textures).

For the second stage, multi-class conditional generation, we use a gating mechanism conditioned on the input class label. Briefly, gating allows the network to focus on the important parts (activations) of the network specific to the conditioning class. Such an approach allows for a clean separation of classes, enabling us to train a single generator and discriminator across *multiple* object classes.

To demonstrate the potential of our method as an interactive tool for stroke-based image generation, we collect a new image dataset of ten simple object classes (pineapple, soccer, basketball, etc.) with white background. In order to stress test our gating mechanism, six of the object classes have similar round outlines, so the model is truly conditioned on the class label and cannot figure out the class only from the stroke. Fig. 2 shows a short video of an interactive editing session using our system. Along with these simple objects, we also demonstrate the potential of our method on complicated ones such as faces and shoes.

## 2. Related Work

**Interactive Generation** Interactive interfaces for free-hand drawing go all the way back to Ivan Sutherland's Sketchpad [48]. The pre-deep work most related to us, ShadowDraw [31], introduced the concept of generating multiple shadows for novice users to be able to draw sketches. PhotoSketcher [13] introduces a retrieval based method for obtaining real images from sketches. More recently, deep recurrent networks have been used to generate sketches [18, 14]. Sketch-RNN [18] provides a completion of partial strokes, with the advantage of intermediate stroke information via the Quickdraw dataset at training time. SPIRAL [14] learns to generate digits and faces using a reinforcement learning approach. Zhu et al. [60] train a generative model, and an optimization-based interface to generate possible images, given color or edge constraints. The technique is limited to a single class and does not propose a recommendation for the completion of the shape. SketchyGAN [3] also aimed at generating multi-class images but lacks interactive capability. In contrast to the above, our method provides interactive prediction of the shape and appearance to the user and supports multiple object classes.

**Generative Modeling** Parametric modeling of an image distribution is a challenging problem. Classic approaches include autoencoders [21, 54] and Boltzmann machines [47]. More modern approaches include autoregressive models [12, 51], variational autoencoders (VAEs) [28], and generative adversarial networks (GANs). GANs and VAEs both learn mappings from a low-dimensional "latent" code, sampled stochastically, to a high-dimensional image through a feedforward pass of a network. GANs have been successful recently [9, 41, 1], and hybrid models feature both a learned mapping from image to latent space as well as adversarial training [10, 11, 30, 4].

**Conditioned Image Generation** The methods described above can be conditioned, either by a low-dimensional vector (such as an object class, or noise vector), a high-dimensional image, or both. Isola et al. [25] propose "pix2pix", establishing the general usefulness of conditional GANs for image-to-image translation tasks. However, they discover that obtaining multimodality by injecting a random noise vector is difficult, a result corroborated in [33, 38, 62]. This is an example of mode collapse [16], a phenomenon especially prevalent in image-to-image GANs, as the generator tends or ignore the low-dimensional latent code in favor of the high-dimensional image. Proposed solutions include layers which better condition the optimization, such as Spectral Normalization [58, 35], modifications to the loss function, such as WGAN [2, 17] or optimization procedure [20], or modeling proposals, such as MAD-GAN [15] and MUNIT [24]. One modeling approach is to add a predictor from the output to the conditioner, to discourage the model from ignoring the conditioner. This has been explored in the classification setting in Auxiliary-Classifier GAN (ACGAN) [36] and regression setting with InfoGAN [4] and ALI/BiGAN ("latent regressor" model) [11, 10], and is one half of BicycleGAN model [62]. We explore a complementary approach of architectural modification via gating.

**Gating Mechanisms** Residual networks [19], first intro-

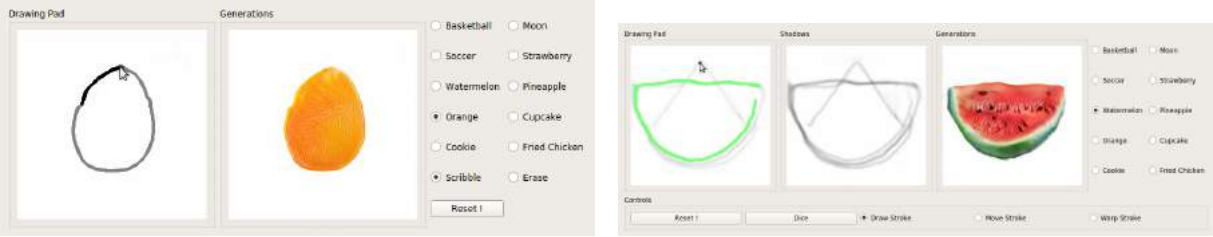



Figure 2: **Video of our interface** We can see two versions of our interface. The left side shows how a user can quickly generate multiple objects using a few strokes, while the right side shows the utility of multimodal completions where the user can quickly explore different possible shape generations while drawing. **Please view with Acrobat Reader.**

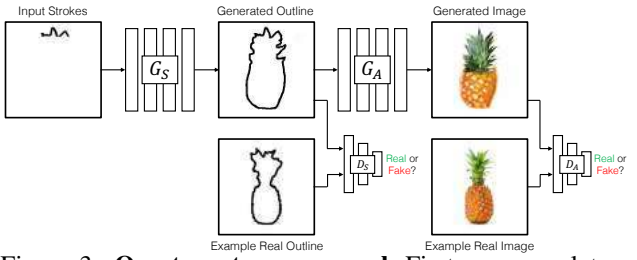



Figure 3: **Our two-stage approach** First, we complete a partial sketch using the shape generator $G_S$. Then we translate the completed sketch into an image using the appearance generator $G_A$. Both generators are trained with their respective discriminators $D_S$, and $D_A$.

duced for image classification [29], have made extremely deep networks viable to train. Veit et al. [53] find that the skip connection in the architecture enables test-time removal of blocks. Follow-up work [52] builds in block removal during training time, with the goal of subsets of blocks specializing to different categories. Inspired by these results, we propose the use of gating for image generation and provide a systematic analysis of gating mechanisms.

The adaptive instance normalization (AdaIn) layer has similarly been used in arbitrary style transfer [23] and image-to-image translation [24], and Feature-wise Linear Modulation (FiLM) [39]. Both methods scale and shift feature distributions, based on a high-dimensional conditioner, such as an image or natural language question. Gating also plays an important role in sequential models for natural language processing: LSTMs [22] and GRU [5]. Similarly, concurrent work [27], [37] use a AdaIN-style network to modulate the generator parameters.

# 3. Method

We decouple the problem of interactive image generation into two stages: object shape completion from sparse user sketches, and appearance synthesis from the completed shape. More specifically, as illustrated in Fig. 3 we use the Shape Generator $G_S$ for the automatic shape (outline/sparse-sketch/simplified-edge) generation and the Appearance Generator $G_A$ for generating the final image as well as the adversary discriminators $D_S$ and $D_A$. Example usage is shown in our user interface in Fig. 2.

## 3.1. Shape completion

The shape completion network $G_S$ should provide the user with a visualization of its completed shape(s), based on the user input, and should keep on updating the suggested shape(s) interactively. We take a data-driven approach for this whereby, to train the network, we simulate partial strokes (or inputs) by removing random square patches from the full outline/ full sparse sketch/ full simplified edges. The patches are of three sizes (64×64, 128×128, 192×192) and placed at a random location in the image of size 256×256 (see Fig. 5 for an example). To extend the technique beyond outlines and generate more human-like sketches, we adopt the multistage procedure depicted in Fig. 6. We refer to these generated sketches as "simplified edges". We automatically generate data in this manner, creating a dataset where for a given full outline/sketch or a simplified edge-map, 75 different inputs are created. The model, shown in Fig. 3, is based on the architecture used for non-image conditional generations in [34]. We modify the architecture such that the conditioning input is provided to the generator and discriminator at multiple scales as shown in Fig. 4. This makes the conditioning input an active part of the generation process and helps in producing multimodal completions.

## 3.2. Appearance synthesis

An ideal interactive sketch-to-image system should be able to generate multiple different image classes with a single generator. Beside memory and time considerations (avoiding loading/using a separate model per class, reducing overall memory), a single network can share features related to outline recognition and texture generation that are common across classes, which helps training with limited examples per class.

As we later show, class-conditioning by concatenation can fail to properly condition the network about the class information in current image translation networks [25, 62].

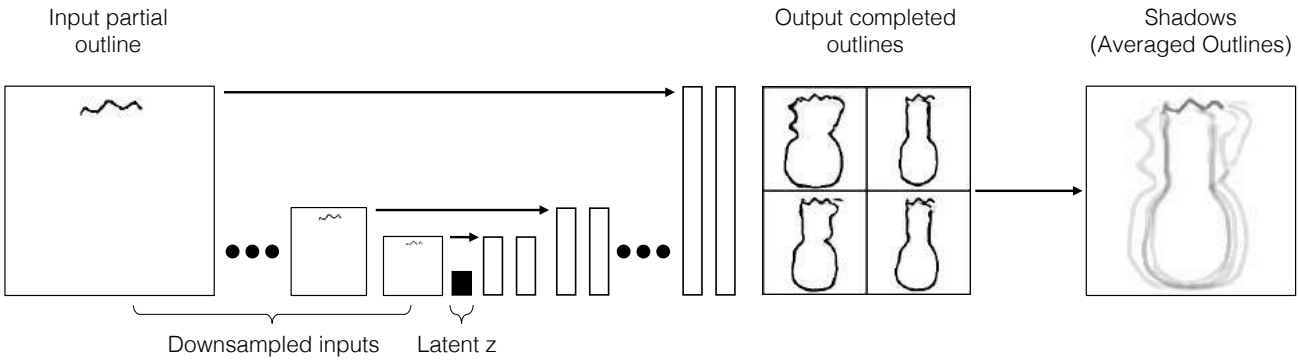
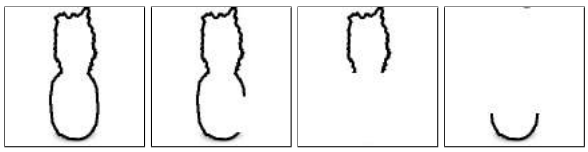



Figure 4: **First stage (Shape Generator)** To achieve multi-modal completions, the shape generator is designed using inspiration from non-image conditional model [34] with the conditioning input provided at multiple scales, so that the generator network doesn't ignore the partial stroke conditioning.



Outline Simulated Partial Inputs

Figure 5: **Simulated Inputs** Three sizes of occluders were used to simulate partial outlines.

Figure 6: **Simplified Edges** The 2nd edgemap is obtained using the technique of [25], while the 3rd is the intermediate edgemap using [32] and further simplified using [46] which looks closer to what a human would sketch.

To address this, we propose an effective soft gating mechanism, shown in Fig. 7. Conceptually, our network consists of a small external gating network that is conditioned on the object class (encoded as a 1-hot vector). The gating network outputs parameters that are used to modify the features of the main generator network. Given an input feature tensor $X_l$, "vanilla" ResNet [19] maps it to

$$X_{l+1} = X_l + \mathcal{H}_l(X_l). \tag{1}$$

Changes in resolution are obtained by upsampling before or downsampling after the residual block. Note that we omit $l$ subscript from this point forward to reduce clutter. Our gating network augments this with a predicted scalar $\alpha$ for each layer of the network using a learned network $\mathcal{F}(\mathbf{y})$, where $\mathbf{y}$ is the conditioning vector:

$$X + \alpha\, \mathcal{H}(X), \text{where } \alpha \in [0, 1] \tag{2}$$

If the conditioning vector $\mathbf{y}$ has no use for a particular block, it can predict $\alpha$ close to zero and effectively switch off the layer. During training, blocks within the main network can transform the image in various ways, and $\mathcal{F}$ can modulate such that the most useful blocks are selected. Unlike previous feature map conditioning methods such as AdaIn [50], we apply gating to *both* the generator and discriminator. This enables the discriminator to select blocks which effectively judge whether generations are real or fake, conditioned on the class input. Some blocks can be shared across regions in the conditioning vector, whereas other blocks can specialize for a given class.

A more powerful method is to apply this weighting channel-wise using a vector $\boldsymbol{\alpha}$:

$$X + \boldsymbol{\alpha} \odot \mathcal{H}(X), \text{where } \boldsymbol{\alpha} \in [0, 1]^c, \tag{3}$$

where $\odot$ represents channel-wise multiplication. This allows specific channels to be switched "on" or "off", providing additional degrees of freedom. We found that this channelwise approach for gating provides the strongest results. AdaIn describes the case where an Instance Normalization [50] (IN) operation is applied before scaling and shifting the feature distribution. We constrain each element of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in $[-1, 1]$. We additionally explored incorporating a bias term after the soft-gating, either block-wise using a scalar $\beta \in [-1, 1]$ per layer, or channel-wise using a vector $\boldsymbol{\beta} \in [-1, 1]^c$ per layer but we found that they did not help much, and so we leave them out of our final model. Refer Fig. 8 for pictorial representation of various gatings.

Finally, we describe our network architecture, which utilizes the gated residual blocks described above. We base our architecture on the proposed residual **Encoder-Decoder** model from MUNIT [24]. This architecture is comprised of 3 `conv` layers, 8 residual blocks, and 3 `up-conv` layers. The residual blocks have 256 channels. First, we deepen the network, based on the principle that deeper networks have more valid disjoint, partially shared paths [53], and add 24 residual blocks. To enable the larger number of residual

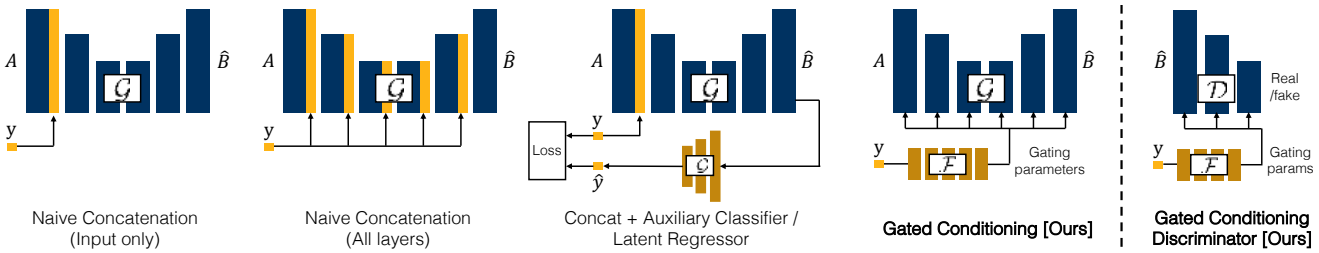



Figure 7: **Conditioning variants for the Appearance Generator** Our model uses gating on all the residual blocks of the generator and the discriminator, other forms of conditioning such as (naive concatenation in input only, all layers, AC-GAN like latent regressor [36]) are evaluated as well.

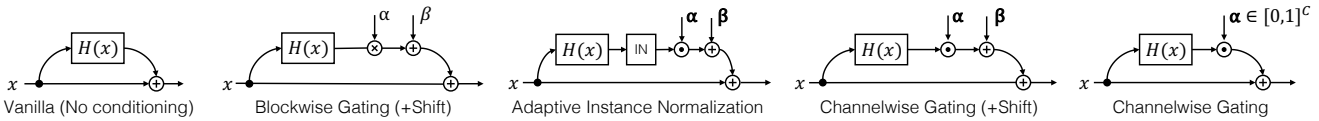



Figure 8: **Injecting conditioning with modified residual layers (Left)** A "vanilla" residual block without conditioning applies a residual modification to the input tensor. **(Mid-left)** The $\mathcal{H}(X)$ block is softly-gated by scalar parameter $\alpha$ and shift $\beta$. **(Mid)** Adaptive Instance Normalization [23] applies a channel-wise scaling and shifting after an instance normalization layer. **(Mid-right)** Channel-wise gating adds restrictions to the range of $\boldsymbol{\alpha}$. **(Right)** We find that channel-wise gating (without added bias) produces the best results empirically.

| Trained task | FID |
|---|---|
| **Faces** | |
| Partial Simplified Edges → Image | 383.02 |
| Partial Simplified Edges → Simplified Edges → Image | 374.67 |
| **Shoes** | |
| Partial Simplified Edges → Image | 170.45 |
| Partial Simplified Edges → Simplified Edges → Image | 154.32 |

Table 1: **Single-class generation, 2-stage vs 1-stage**. We evaluate the result quality from different task pipelines.

| Trained task | Avg Acc |
|---|---|
| Partial edges → Image | 73.12 % |
| Partial outline → Image | 88.74 % |
| Partial outline → Full outline → Image [Ours] | **97.38%** |

Table 2: **Multi-class generation, 2-stage vs 1-stage**. We evaluate the result quality from different task pipelines. Accuracy is computed by a fixed, pretrained classification network, on the resulting images.

blocks, we drastically reduce the width to 32 channels for every layer. We refer to this network as **SkinnyResNet**. Additionally, we found that modifying the downsampling and upsampling blocks to be residual connections as well improved results, and also enables us to apply gating to *all* blocks. When gating is used, the gate prediction network, $\mathcal{F}(\mathbf{y})$, is also designed using residual blocks. Additional architecture details are in the supplementary material.

# 4. Experiments

We first compare our 2 step approach for interactive image generation on existing datasets such as the UTZappos Shoes dataset [57] and CelebA-HQ [26]. State-of-the-art techniques such as pix2pixHD [55] are used to generate the final image from the autocompleted sketches. We finally evaluate our approach on a multi-class dataset that we collected to test our proposed gating mechanism.

## 4.1. Single Class Generation

**Datasets** We use the edges2shoes[25], CelebA-HQ[26] datasets to test our method on single class generation. We simplify the edges to attempt to more closely resemble how humans would draw strokes by first using the preprocessing code of [32] further reducing the strokes with a sketch simplification network [46].

**Architecture** We use the architecture described in Section 3.1 for shape completion. In this case, each dataset only contains a single class, so we can use an off-the-shelf network, such as pix2pixHD [56] for rendering.

**Results** As seen in Fig. 9, our 2 step technique allows us to complete the simplified edge maps from the partial strokes and also generate realistic images from the autocompleted simplified edges. Table 1 also demonstrates, across two datasets (faces and shoes), that using a 2 step procedure produces stronger results than mapping directly from the partial sketch to the completed image.

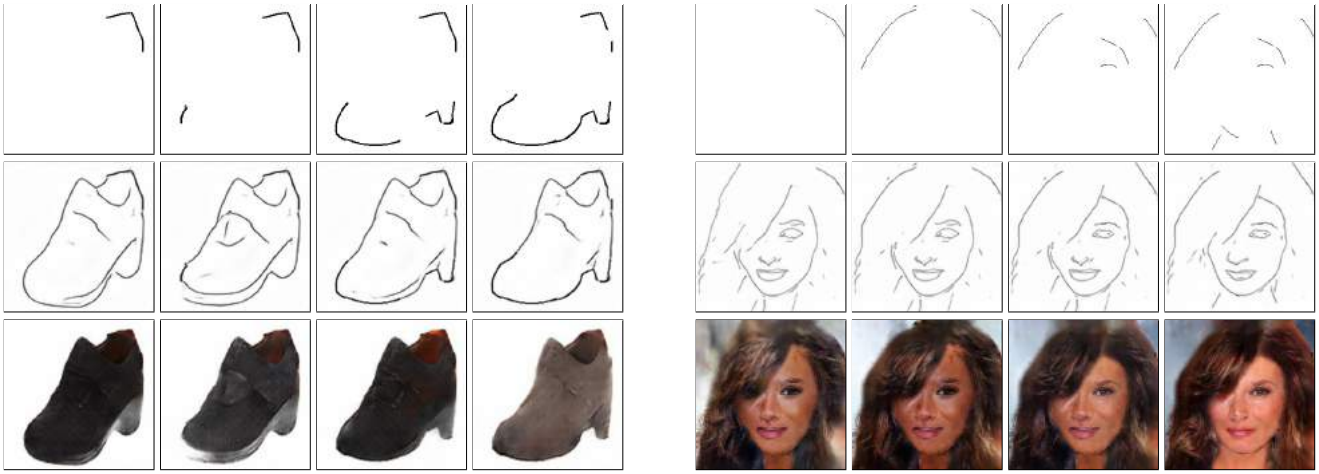


Figure 9: **Example Sketch & Fill Progression.** The **first row** represents the progressive addition of new strokes on the canvas, the **second row** shows the autocompleted sketch, and the **third row** is the final generated image. As the sparse strokes are changed by the user, the completed shape and generated image evolve as well. Note that changing a stroke locally produces coherent changes in other parts of the image.

| Method | SkinnyResNet | | EncDec | |
|---|---|---|---|---|
| | Class. Acc [%] | AMT Fool. Rate [%] | Class. Acc [%] | AMT Fool. Rate [%] |
| Ground truth | 100.0 | 50.0 | 100.0 | 50.0 |
| 1 gen/class | ***97.0*** | 17.7±1.46 | – | – |
| Concat (In) | 62.6 | 15.0±1.4 | 39.2 | 7.5±1.06 |
| Concat (All) | 64.5 | 15.3±1.41 | 51.4 | 5.4±0.88 |
| Cat(In)+Aux-Class | 65.6 | 14.5±1.5 | – | – |
| Cat(All)+Aux-Class | 67.0 | 19.7±1.42 | – | – |
| BlockGate(+bias) | 89.6 | 19.6±1.34 | – | – |
| BlockGate | **99.6** | 17.3±1.61 | – | – |
| AdaIn | 94.5 | 14.9±1.47 | – | – |
| ChanGate(+bias) | 94.1 | 14.8±1.43 | – | – |
| ChanGate | ***97.0*** | **23.4±1.99** | 92.7 | 14.1±1.48 |

Table 3: **Accuracy vs Realism on Multiclass Outline→Image task.** We measure generation accuracy with a pretrained network. We measure realism using the real vs. fake judges from AMT. Higher is better for both. Our SkinnyResNet architecture outperforms the Encoder-Decoder network, inspired by MUNIT [24]. We perform a thorough ablation on our architecture and find that channel-wise gating achieves high accuracy and higher realism.

## 4.2. Multi-Class Generation

**Datasets** To explore the efficacy of our full pipeline, we introduce a new outline dataset consisting of 200 images (150 train, 50 test) for each of 10 classes – basketball, chicken, cookie, cupcake, moon, orange, soccer, strawberry, watermelon and pineapple. All the images have a white background and were collected using search keywords on popular search engines. In each image, we obtain rough outlines for the image. We find the largest blob in the image after thresholding it into a black and white image. We fill the interior holes of the largest blob and obtain a smooth outline using the SavitzkyGolay filter [44].

**Architecture** For the shape completion, we use the architecture in Section 3.1. For class-conditioned image generation, test the gated architectures in Section 3.2.

**Results** In order to test the fidelity of the automatically completed shapes, we evaluate the accuracy of a trained classifier on being able to correctly label a particular generation. We first test in Table 2 that our 2 stage technique is better than 1 step generation. We evaluate the results on the multi-class outline to image generations on two axes: adherence to conditioning and realism. We first test the conditioning adherence – whether the network generates an image of the correct class. Off-the-shelf networks have been previously used to evaluate colorizations [59], street scenes [25, 56], and ImageNet generations [42]. We take a similar approach and fine-tune a pretrained InceptionV3 network [49] for our 10 classes. The generations are then tested with this network for classification accuracy. Results are presented in Table 3.

To judge the generation quality, we also perform a "Visual Turing test" using Amazon Mechanical Turk (AMT). Turkers are shown a real image, followed by a generated image, or vice versa, and asked to identify the fake. An algorithm which generates a realistic image will "fool" Turkers into choosing the incorrect image. We use the implementation from [59]. Results are presented in Table 3, and qualitative examples are shown in Fig. 10.

**Gating Architectures** We compare our proposed model to the residual **Encoder-Decoder** model [24]. In addition, we compare our proposed gating strategy and **SkinnyResNet** architecture to the following methods for conditional image generation:


Figure 10: **Conditioning injection comparison.** We show results across methods on the outline→image task using the **SkinnyResNet** architecture. Naive Concatenation **Concat** often confuses classes, such as oranges and basketballs, while gating mechanisms such as the **ChannelGate** method succeed. The gating method also improves results for the **EncoderDecoder** architecture.

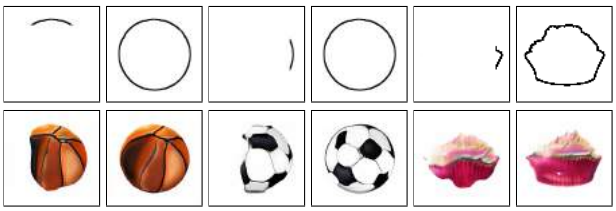


Figure 11: **Directly mapping from partial outline to image** Our proposed system uses a 2-stage approach, using a completed edge map as an intermediate. Here, we show results when directly mapping from the partial outline to the image. When the outline is well-defined, the network can generate realistic images. However, when the outline is sparse, the network struggles with the geometry.

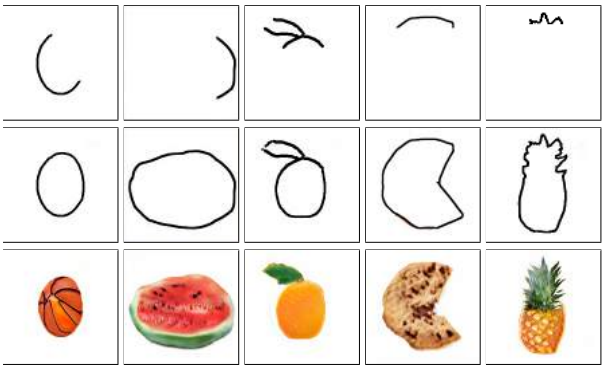


Figure 12: **Multiclass Sketch & Fill results** A few input strokes (first row) are enough to automatically complete the class specific outlines (second) and appearance (last).

- **Per-class**: a single generator for each category; this is the only test setting with *multiple* networks, all others train a single network
- **Concat (In)**: naive concatenation, input layer only
- **Concat (All)**: naive concatenation, all layers
- **Concat (In)+Aux-Class**: we add an auxiliary classifier, both for input-only and all layers settings
- **BlockGate(+Bias), BlockGate**: block-wise soft-gating, with and without a bias parameter
- **AdaIn**: Adaptive instance normalization
- **ChannelGate(+Bias), ChannelGate**: channel-wise soft-gating, with and without a bias parameter

**Does naive concatenation effectively inject conditioning?** In Fig. 10, we show a selected example from each of the 10 classes. The per-class baseline trivially adheres to the conditioning, as each class gets to have its own network. However, when a single network is trained to generate all classes, naive concatenation is unable to successfully inject class information, for either network and for either type of concatenation. For the **EncoderDecoder** network, basketballs, oranges, cupcakes, pineapples, and fried chicken are all confused with each other. For the **SkinnyResNet** network, oranges are generated instead of basketballs, and pineapples and fried chicken drumsticks are confused. As seen in Table 3, classification accuracy is slightly higher when concatenating all layers ($64.5\%$) versus only the input layer ($62.6\%$), but is low for both.

**Does gating effectively inject conditioning?** Using the proposed soft-gating, on the other hand, leads to successful generations. We test variants of soft-gating on the **SkinnyResNet**, and accuracy is dramatically improved, between $89.6\%$ to $99.6\%$, comparable to using a single generator per class ($97.0\%$). Among the gating mechanisms, we find that channel-wise multiplication generates the most realistic images, achieving an AMT fooling rate of $23.4\%$. Interestingly, the fooling rate is higher than the per-class generator of $17.7\%$. Qualitatively, we notice that per-class generators sometimes exhibits artifacts in the background, as seen in the generation of "moon". We hypothesize with the correct conditioning mechanism, the single generator across multiple classes has the benefit of seeing more training data and finding common elements across classes, such as clean, white backgrounds.

**Is gating effective across architectures?** As seen in Table 3, using channelwise gating instead of naive concatenation improves performance both accuracy and realism *across* architectures. For example, for the **EncoderDecoder** architecture, gating enables successful generation of the pineapple. Both quantitatively and qualitatively, results are better for our proposed **SkinnyResNet** architecture.

**Do the generations generalize to unusual outlines?** The training images consist of the outlines corresponding to the geometry of each class. However, an interesting test scenario is whether the technique generalizes to unseen shape and class combinations. In Fig. 1, we show that an input circle not only produces circular objects, such as a basketball, watermelon, and cookie, but also noncircular objects such as strawberry, pineapple, and cupcake. Note that both the pineapple crown and bottom are generated, even without any structural indication of these parts in the outline.

## 5. Discussion

We present a two-stage approach for interactive object generation, centered around the idea of a shape completion intermediary. This step both makes training more stable and also allows us to give coarse geometric feedback to the user, which they can choose to integrate as they desire.

## Acknowledgements

AG, PKD, and PHST are supported by the ERC grant ERC-2012-AdG, EPSRC grant Seebibyte EP/M013774/1, EPSRC/MURI grant EP/N019474/1 and would also like to acknowledge the Royal Academy of Engineering and FiveAI. Part of the work was done while AG was an intern at Adobe.

## References


[1] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 2

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *ICML*, 2017. 2

[3] Wengling Chen and James Hays. Sketchygan: towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018. 2

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *NIPS*, 2016. 2

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014. 3

[6] Dale J Cohen and Susan Bennett. Why can't most people draw what they see? *Journal of Experimental Psychology: Human Perception and Performance*, 23(3):609, 1997. 1

[7] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics (TOG)*, 27(3):88, 2008. 2

[8] Tali Dekel, Chuang Gan, Dilip Krishnan, Ce Liu, and William T Freeman. Sparse, smart contours to represent and edit images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3511–3520, 2018. 1

[9] Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[10] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *ICLR*, 2017. 2

[11] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *ICLR*, 2017. 2

[12] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 2

[13] Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*, 31(6):56–66, 2011. 2

[14] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *ICML*, 2018. 2

[15] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip H. S. Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *CVPR*, 2018. 2

[16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2

[17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, pages 5767–5777, 2017. 2

[18] David Ha and Douglas Eck. A neural representation of sketch drawings. *Conference on Neural Information Processing Systems*, 2017. 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. 2

[21] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[23] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1510–1519, 2017. 3, 5

[24] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018. 2, 3, 4, 6

[25] P. Isola, J-Y. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2, 3, 4, 5, 6

[26] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018. 5

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019. 3

[28] D. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014. 2

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 3

[30] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 2

[31] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. Shadowdraw: real-time user guidance for freehand drawing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 27. ACM, 2011. 2

[32] Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1525–1534, 2019. 4, 5

[33] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 2

[34] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *ICML*, 2018. 3, 4

[35] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018. 2

[36] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *ICML*, 2017. 2, 5

[37] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. *CVPR*, 2019. 3

[38] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2017. 2

[39] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *AAAI*, 2018. 3

[40] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *ACM Transactions on Graphics (TOG)*, 37(4):99, 2018. 1

[41] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016. 2

[42] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *NIPS*, 2016. 6

[43] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, volume 2, 2017. 1

[44] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964. 6

[45] Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. On expert performance in 3d curve-drawing tasks. In *Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling*, pages 133–140. ACM, 2009. 1

[46] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):121, 2016. 4, 5

[47] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986. 2

[48] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE Design Automation Workshop*, DAC '64, pages 6.329–6.346, New York, NY, USA, 1964. ACM. 2

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 6

[50] D Ulyanov, A Vedaldi, and VS Lempitsky. Instance normalization: the missing ingredient for fast stylization. corr abs/1607.0 (2016). 4

[51] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016. 2

[52] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. *ECCV*, 2018. 3

[53] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558, 2016. 3, 4

[54] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 2

[55] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 5

[56] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CVPR*, 2018. 5, 6

[57] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014. 5

[58] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *ICML*, 2019. 2

[59] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pages 649–666. Springer, 2016. 6

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

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CVPR*, 2017. 1

[62] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. pages 465–476, 2017. 2, 3