# Photorealistic Image Generation of Architecture from Sketches using GANs

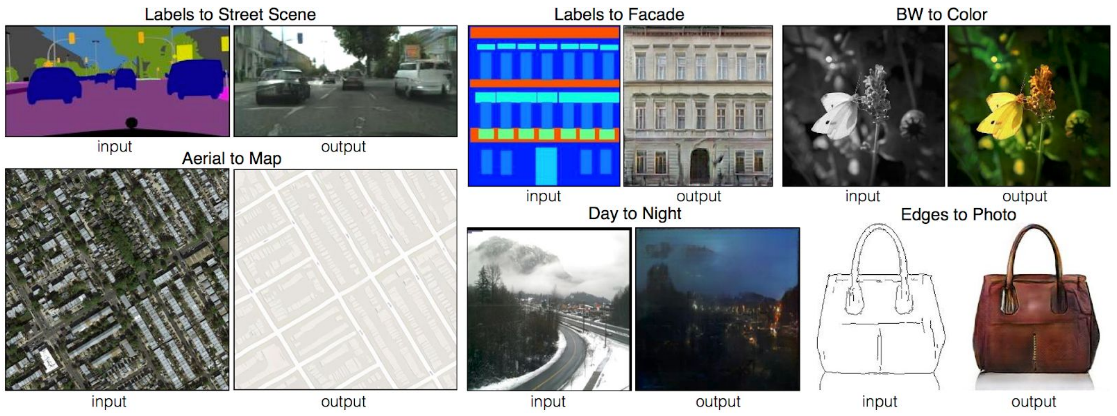
Daniel Wen (lanhaow), Hima Tammineedi (htammine)
Shalom Yiblet (syiblet), Irina Javed (irjaved)

## Abstract

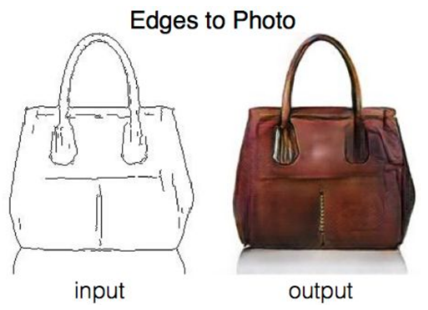
The idea of converting a given input image into another, transformed, output image is an interesting concept. The approach we take in this project is to use a Generative Adversarial Network (GAN) to learn a generative model which can perform this task. This approach is beneficial in that the loss function is learned from the data, so the same network can be applied to a variety of different image to image translation problems. We present our network architecture as well as results on using this approach to convert sketches of cats, shoes, and buildings into their photorealistic counterpart.

## Introduction

We wish to understand how to produce realistic looking images from simple sketches of an image. To do this, we critique and implement the algorithm described in “Image-to-Image Translation with Conditional Adversarial Networks” (Isola et al., 2016). The paper investigates how conditional adversarial networks can be used for image-to-image translation problems. Their results were very interesting:



They were able to use a single general model and apply it to multiple different image translation problems such as converting satellite images to map images and colorizing black and white images. For our project, we only focus on applying this model to the problem of turning edges into realistic photos. In particular, we will apply this model to images of buildings, cats, and shoes.



This research is significant because it can help designers and architects rapidly create and visualize products and models. We also pursue this research because GANs are a new field that shows a lot of potential.

**Literature Review**

In our project, we chose to recreate the model and results presented in P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros's paper, *Image-to-Image Translation with Conditional Adversarial Networks*. Before diving into that, however, we first gained a better understanding of the theory behind the aforementioned paper's model.

In trying to understand conditional generative adversarial networks (CGANs), we first consulted the paper which first introduced the world to generative adversarial networks (GANs), Ian Goodfellow and team's paper, *Generative Adversarial Nets*. The paper outlines the general format of the GAN problem, detailing the loss function the titular neural networks use and the data distributions GANs both sampled from and learned. The paper also describes the convergence of said data distributions, given that the discriminator and generator have enough capacity, to a global optimum where the real data distribution equals the generated distribution. The theory behind this conclusion was presented in a mathematically dense fashion, but its implications were present not only in the most theoretical sense but when we ran our own model as well. The downside of the model presented by Goodfellow is that the generator often produces noisy outputs, incapable of photorealism in image generation tasks, as compared to later models.

In knowing our problem used conditional generative adversarial nets, we additionally consulted M. Mirza and S. Osindero's paper, *Conditional Generative Adversarial Nets*. The paper briefly presents what a GAN is (drawing from the Goodfellow paper we ourselves used) as well as the conditional mode of a GAN, and immediately goes into various experimental results using the latter. The GAN and CGAN are almost identical, except for an additional input given to the CGAN's discriminator and generator in producing their output. The distinction is a small yet powerful one, allowing a significant increase in the capabilities of the GAN model--especially when it comes to image translation tasks.

Coming back to our original paper, *Image-to-Image Translation* expounds upon the CGAN like the other papers and goes into the various image-to-image translation tasks the CGAN can be used for. Such tasks include generating aerial photos from maps, cityscape/building labels to photorealistic images, day to night photos, and what we implemented in our project--sketches to filled in images. The paper also details a more complicated generator architecture we found to be better suited to the image-translation problem at hand. The increases in performance our model was able to achieve compared to more traditional models were due to specific model design choices made by the paper, such as the U-Net framework with skip-connections to better allow lower-level information to be passed through or the added L1 loss to the objective function to improve the generator's performance.

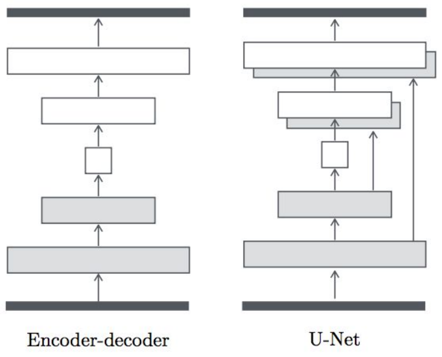
To better improve our understanding of the U-Net framework we consulted O. Ronneberger, P. Fischer, and T. Brox's paper *U-net: Convolutional networks for biomedical image segmentation*. While detailing a mapping problem different from our own (in one case, the segmentation of neuronal structures using CNNs), it nonetheless provided the necessary information to help us apply the U-Net architecture to one of our own CNNs--the generator. It describes the downsampling and upsampling in the expanding and contracting paths, respectively, of the U-Net. It also details how outputs can be made more precise by combining the the higher resolution features of said contracting path with the outputs of the expanding path, vital to the image translation we tackled in our own project.

## Architecture

Using a GAN is advantageous because we do not need to specify an explicit loss function that produces realistic images. Instead, the GAN learns a loss function based on the data given to the network. These loss functions are explained in the next section.

A GAN consists of two networks: a generator and a discriminator. The generator create a fake image given a sketch, and the discriminator decides whether the generated image is real or fake. The generator is an encoder-decoder, which repeatedly downsamples the input, and then upsamples it back into the size of the desired output. However, since downsampling reduces the amount of information, we cannot expect such a network to produce images with sharp details.

To address this issue, we add skip connections, which send information from a layer in the encoder to the corresponding layer in the decoder. This network structure is known as the "U-Net" architecture [4].



The U-Net architecture features skip connections between mirrored layers (layer i connects to layer (n-i) directly, where n is the total number of layers)

Our architecture is based on [2]. The layer notation is as follows.

- Ck: Convolution-BatchNorm-ReLU with k filters.
- CDk: Convolution-BatchNorm-Dropout-ReLU layer with k filters.

Layer hyperparameters:

- Convolution: stride 2, 4x4 kernel size
- Dropout: rate 0.5

Discriminator architecture: C64-C128-C256-C512-C512-C512-C1

- After the last layer above, a sigmoid function is applied (the resulting scalar is the probability of the input being a real image)
- Leaky ReLU is used instead of ReLU

Generator architecture (encoder-decoder)

- Encoder: C64-C128-C256-C512-C512-C512-C512
    - Leaky ReLU is used instead of ReLU
- Decoder: CD512-CD1024-CD1024-C1024-C512-C256-C128
    - The first layer does not have batch normalization
    - The last layer outputs 3 channels, followed by a tanh function

**Algorithm**

The objective of a GAN is to train (1) a generator which takes random noise as input and then generates some output data and (2) a discriminator tasked with determining if its input is real data or fake data generated by the generator. Let **G**(***z***) be the output, or the fake data, generated by passing noise, ***z***, into function **G**. Let **D**(***x***) be the probability that ***x*** is sampled from the data distribution, i.e. the real data. We then want to alternately train the discriminator **D** to maximize the objective function V(**D**,**G**) (see below) and train the generator **G** to minimize V(**D**,**G**). Intuitively this means training **D** to correctly classify both the real and fake data, and training **G** to “trick” **D** into classifying the generated data as real data. It can be shown that the optimal solution to this model when $p_{data} = p_g$ occurs when, given $\mathbf{x} \sim p_{data}$ and $\mathbf{z} \sim p_z$, $\mathbf{D}(\boldsymbol{x}) = \mathbf{D}(\mathbf{G}(\boldsymbol{z})) = 0.5$, or when the discriminator is unable to differentiate and is equally likely to classify the input as real or fake.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

When implementing a GAN, we generally follow the below model:
First sample *n* noise samples from fake distribution $p_z$, ($\{z^{(1)}, z^{(2)}, \ldots, z^{(n)}\}$), and *n* samples from the data distribution $p_{data}(\boldsymbol{x})$, ( $\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ ). Then update the discriminator by ascending its stochastic gradient (want to maximize the function) over the n samples (see below).

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

Do this first step *k* times. Follow that by again sampling *n* noise samples from distribution $p_z$, ($\{z^{(1)}, z^{(2)}, \ldots, z^{(n)}\}$), and updating the generator by descending its stochastic gradient (want to minimize the function over the n samples (see below).

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Do both steps for however many iterations.

In our model, we build further upon the basic idea of a GAN, by using a conditional GAN. A Conditional Generative Adversarial Network (CGAN) differs from a GAN only in that ***x*** and ***z*** are now passed through functions **D** and **G** conditioned on some extra information ***y***. The model changes as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{z}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

Conditional GANs are useful in order to have the generator output images based on certain conditions (like a class variable). For example, if you have a GAN trained on the MNIST data, you could condition on the digit 7, in order to have your generator produce only images of the number 7.

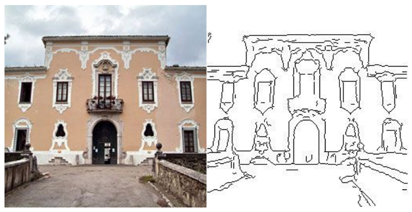
In our case, we are conditioning on an input image which is a sketch of a particular building. The generator should then produce a realistic looking building image given the input sketch so that it generates the same image as the input sketch, except that it will be colorized properly. If we didn't do this, then the generator would produce a generated image of any building it wants. /

Note that our generator only takes the sketch as input, without noise. This strategy was presented in [2] because it simplifies the architecture.

**Data sets**
Our data sets consist of images cropped and resized to 128x128.

| Data set | Number of training examples | Source(s) |
|---|---|---|
| Buildings | 12731 | ImageNet, Sheffield |
| Cats | 3324 | Microsoft Research Asia |
| Shoes | 6311 | UT Zappos50K |

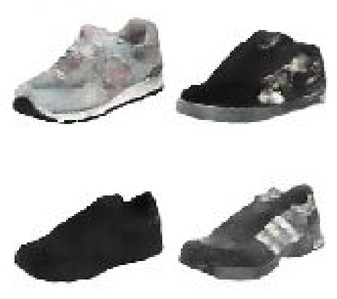
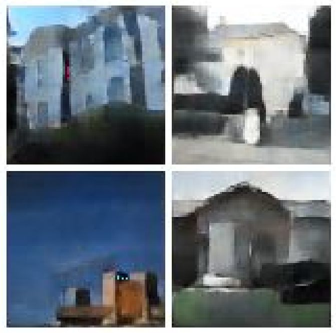
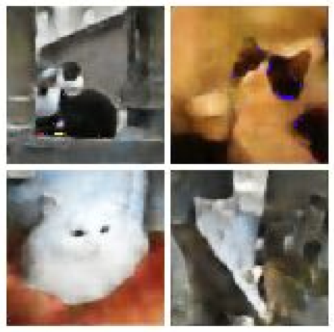
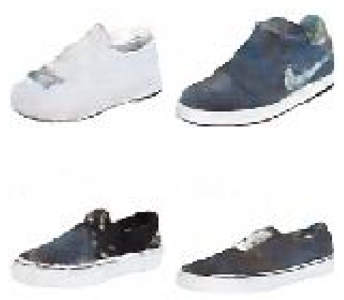
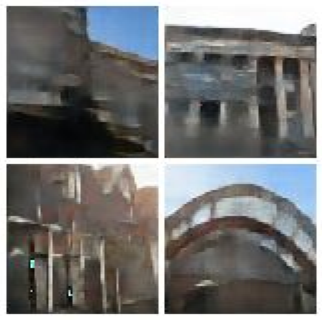
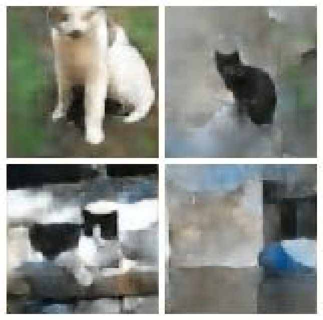
To generate the sketches, we performed Holistically Nested Edge Detection on the images followed by post-processing.



The images were of good quality and cleanliness. Unlike the other two data sets, the shoe images isolate the subject on a white background. It is likely that the reduced noise from the background improved the generated shoes.

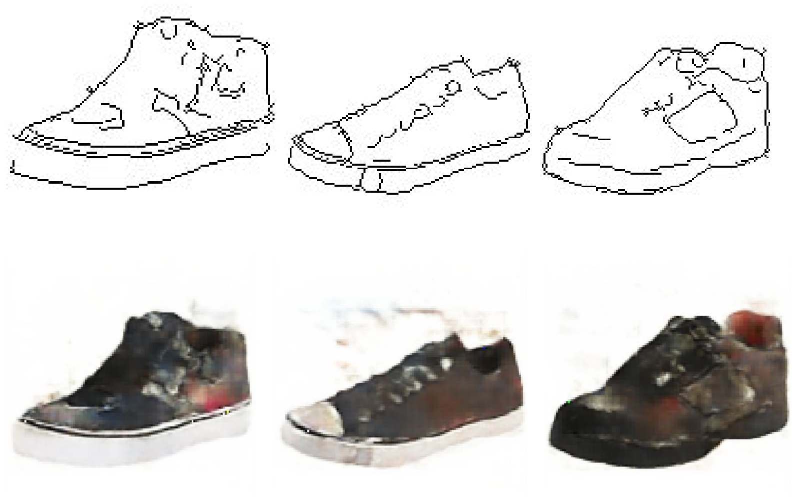
Some other datasets for buildings that we found were very noisy and had objects other than buildings, and so we decided against using those.

**Results**

As you can see in the figures above, for the three data sets that we trained our GAN on, our generator learned to generate realistic images on the training data set. We didn’t have a concrete way of determining how “good” a generated image is other than by using our own judgements. Simply from eyeballing the results, it's evident that the shoe data set generated the most realistic images. Interestingly, however, when comparing the losses of the GAN on each data set, we saw that the discriminator was at a 0.5 confidence for all images which meant that it wasn’t able to distinguish real from fake at a higher rate than random chance. This is exactly what we wanted as this means the generator network was producing images so real that the discriminator wasn’t able to tell the difference anymore.

In the Figures below the first line is the sketches given to the generator and the second line is the generated result image. As you can see the results are pretty good. The generated shoes look realistic. However, out of the three data sets, the building data set and the cat data set led to poor results on the images that the GAN didn’t train on. Only the shoe data set did well on the test data.



Compared to other data sets, the shoe data was the most organized, cleanest, and largest data set that we trained on. From this, we conclude that the quality of the generated images strongly depends on the quality of the training data. In the original paper by Isola et al, they got their results by training on data sets that were orders of magnitude larger than hours. We believe that had we had more data to train on, the quality of the images produced would been comparable to the original paper.

We do think the generated images could have been produced at a higher quality, as they are a bit blurry in some cases. This problem could be remedied with a different weighting on our L1 regularization term, or perhaps with more training.

Overall, we found that the architecture specified in the paper we implemented was a good choice, especially with its use of skip connections, and layer design. Increasing the number of layers may also produce even better results.

## Conclusion

The results in this paper show that it is actually very easy to learn mappings from input images to transformed output images. Our output images show us that we were actually able to generate realistic looking images. One problem, however, was that our produced images weren't of the best quality, and had a lot of noise in some cases. But in general, the model was able to very well approximate what we wanted. Improvements could possibly be made by changing the network architecture or reconfiguring the loss function in order to increase the sharpness of the produced images.

## References

1. Goodfellow, Ian, et al. “Generative adversarial nets.” Advances in Neural Information Processing Systems. 2014.

2. 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.

3. M. Mirza, S. Osindero. Conditional Generative Adversarial Nets. arXiv preprint arXiv:1411.1784, 2014.

4. O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, pages 234–241. Springer, 2015.