CIS 680 Final Project
An Implementation of pix2pix

Grace Boatman [1] **Trevor Wexner** [2]

Video Link: https://www.youtube.com/watch?v=_VbRQ-i5IGk

Abstract

We implement the pix2pix algorithm originally developed by Berkley AI Research (Isola et al., 2017) with some variations in order to transform images of one format into another. Specifically, we build a conditional GAN model to transform semantic labels of a scene to to photographic representations of that scene. Our conditional GAN uses a UNet generator and a PatchGAN discriminator. We experiment with various patch sizes in the discriminator as well as various distance penalties in the loss function, ultimately achieving visually convincing results.

1. Introduction

GANs have been a prevalent topic in recent computer vision research (MIT). These networks have been used for various tasks, from generating fake images from random noise (Goodfellow et al., 2014) to transforming the gender or facial hair composition of a face in a photograph (He et al., 2019). All of these applications have the same general architecture - a generator, which learns how to transform random noise or a given input to look like a desired output, and a discriminator, whose job it is to distinguish the fake generated output from the ground truth. These two components are trained alternately until the discriminator is no longer able to distinguish between the ground truth and the output of the generator.

While more traditional GANs have taken no input, instead simply sampling random noise to generate a fake image of a desired type, more recent works have allowed for inputs to solve image transformation tasks (Isola et al., 2017). These networks, called conditional GANs (cGANs), have the generator take in a desired input, x, and have the output of the discriminator conditioned on that input (Mirza et al.,2014). For example, rather than simply generating an instance of a face with a mustache, a cGAN would take in a face with no mustache as input, generate an image of that face with a mustache, and then have the discriminator decide whether that generated image is a true face with a mustache given what the original face looked like.

Such a cGAN architecture is the focus of Berkley AI Lab's pix2pix network, which we implemented in this paper. Specifically, the pix2pix model is developed with the goal of being a generic framework that has the ability perform any image translation problem. As such, we implement the proposed cGAN architecture and evaluate the performance of our pix2pix model on translating segmentation labels to corresponding photographs.

2. Methods

2.1. Network Architecture

As described in the original paper, we use a UNet generator followed by a PatchGAN discriminator.

2.1.1. Generator

Because the generator takes in an image as input that needs to be transformed to output of the same dimensions, the authors select (and we replicate) an encoder-decoder based architecture. Specifically, we use the proposed UNet architecture. Rather than a traditional VAE-type framework, which simply uses the output of each previous layer as input to the subsequent layer, the UNet uses skip connections, wherein the output of each layer $i < n/2$ of the encoder passed directly to layer $n - i$ of the decoder. These skip connections allow for patterns detected earlier on to be passed directly to output decoder layers, thereby preserving the observed pattern and not diluting in through further convolutions. Moreover, these skip connections can speed up convergence.

We adhere largely to the architecture proposed in Isola's paper, using 7-layer encoder and decoders. Specifics are shown in the tables 1 and 2. All convolutions are 4x4 filters with stride 2 and padding = 1. A slope of .2 is used for all leaky ReLU's. We use a dropout probability of .5 for the decoder.

Layer	Encoder Architecture
Layer 1	Conv2d, 64 filters, Leaky-ReLU
Layer 2	Conv2d, 128 filters, BatchNorm, Leaky-ReLU
Layer 3	Conv2d, 256 filters, BatchNorm, Leaky-ReLU
Layer 4	Conv2d, 512 filters, BatchNorm, Leaky-ReLU
Layer 5	Conv2d, 512 filters, BatchNorm, Leaky-ReLU
Layer 6	Conv2d, 512 filters, BatchNorm, Leaky-ReLU
Layer 7	Conv2d, 512 filters, BatchNorm, Leaky-ReLU
Layer 8	Conv2d, 512 filters, BatchNorm, Leaky-ReLU

Table 1. Encoder Architecture

Layer	Decoder Architecture
Layer 1	Conv2dTranspose(Encoder Output), 512 filters, BatchNorm, Dropout, ReLU
Layer 2	Conv2dTranspose(Layer 1 + Encoder Layer 7), 512 filters, BatchNorm, Dropout, ReLU
Layer 3	Conv2dTranspose(Layer 2 + Encoder Layer 6), 512 filters, BatchNorm, Dropout, ReLU
Layer 4	Conv2dTranspose(Layer 3 + Encoder Layer 5), 512 filters, BatchNorm, ReLU
Layer 5	Conv2dTranspose(Layer 4 + Encoder Layer 4), 256 filters, BatchNorm, ReLU
Layer 6	Conv2dTranspose(Layer 5 + Encoder Layer 3), 128 filters, BatchNorm, ReLU
Layer 7	Conv2dTranspose(Layer 6 + Encoder Layer 2), 64 filters, BatchNorm, ReLU
Layer 8	Conv2dTranspose(Layer 7 + Encoder Layer 1), 3 filters, Tanh

Table 2. Encoder Architecture

2.1.2. Discriminator

We also largely adhere to the discriminator function, coined *PatchGAN* by the pix2pix creators. This framework only penalizes predictions on the scale of image patches, as opposed to individual pixel-wise error attribution otherwise commonly used. This patch aggregation allows for better capture of neighborhood features. Each ”patch” corresponds to the original receptive field of each output pixel. Thus, different patch sizes will correspond to different network architectures. While the authors of the original paper experiment with patches of size 1x1, 16x16, 70x70, and 286x286, we restrict our focus to patches of size 1x1, 70x70, 142x142, and 286x286. We choose to experiment with 142x142 patches rather than the 16x16 described in the paper, because we believe this larger patch size will give more insight into capturing actual neighborhood-level traits for larger objects. The specific architectures our networks are in tables 3,4,5, and 6. For each architecture, the discriminator is trained on sets of two photos that are concatanated: (X, Y_{fake}) and (X, Y_{true}). Here X is the input image, Y_{fake} is the corresponding generator output, and Y_{true} is the corresponding ground truth. With the exception of the 1x1 patch network, all convolutions in the 70x70 network have a kernel of size 4, stride of 2 (unless otherwise specified), and padding of size one. The 1x1 network simply uses 1x1 convolutions for all layers.

2.2. Objective Function

The loss function for a conditional GAN is traditionally specified as:

$$L_{cGAN}(G, D) = E_{x,y}[log(D(x, y))] + \\ E_{x,z}[log(1 - D(x, G(x, z))]$$

The authors of the original pix2pix paper use this loss to define the network objective:

$$G* = \arg\min_G \max_D L_{cGAN}(G, D) + \lambda L_{dist}(G)$$

Here L_{dist} is defined so that, not only must output of the generator fool the discriminator, but it must also be close in a distance sense to the ground truth. The original pix2pix algorithm uses an L1 distance, but we experiment with L1, L2, and no distance adjustments.

70x70 patches are used in experimenting with the non-L1 losses because they were found to be most optimal by Isola et al. We compare the results from each distance loss.

2.3. Data

Our algorithm was trained using the *cityscapes* dataset provided by the authors (Cordts et al., 2016). These contain pairs of photos and their corresponding semantic labels for various scenes in urban areas. A test/train split is provided. All images were size 256x256 and were normalized within the -1 to 1 range. Training was performed on a subset of

Layer	**70x70 PatchGAN Architecture**
Layer 1	Conv2d, 64 filters, leaky-ReLU (slope = .2)
Layer 2	Conv2d, 128 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 3	Conv2d, 256 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 4	Conv2d, 512 filters, stride = 1, BatchNorm, leaky-ReLU (slope = .2)
Layer 5	Conv2d, 1 filter, stride = 1, Sigmoid

Table 3. 70x70 PatchGAN Architecture

Layer	**142x142 PatchGAN Architecture**
Layer 1	Conv2d, 64 filters, leaky-ReLU (slope = .2)
Layer 2	Conv2d, 128 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 3	Conv2d, 256 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 4	Conv2d, 512 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 5	Conv2d, 512 filters, stride = 1, BatchNorm, leaky-ReLU (slope = .2)
Layer 6	Conv2d, 1 filter, stride = 1, Sigmoid

Table 4. 142x142 PatchGAN Architecture

Layer	**286x286 PatchGAN Architecture**
Layer 1	Conv2d, 64 filters, leaky-ReLU (slope = .2)
Layer 2	Conv2d, 128 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 3	Conv2d, 256 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 4	Conv2d, 512 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 5	Conv2d, 512 filters, BatchNorm, leaky-ReLU (slope = .2)
Layer 6	Conv2d, 512 filters, stride = 1, BatchNorm, leaky-ReLU (slope = .2)
Layer 7	Conv2d, 1 filter, stride = 1, Sigmoid

Table 5. 286x286 PatchGAN Architecture

Layer	**1x1 PatchGAN Architecture**
Layer 1	Conv2d, 64 filters, stride = 1, leaky-ReLU (slope = .2)
Layer 2	Conv2d, 128 filters, stride = 1, BatchNorm, leaky-ReLU (slope = .2)
Layer 3	Conv2d, 1 filter, stride = 1, Sigmoid

Table 6. 1x1 PatchGAN Architecture

1,500 images, batch size 64, learning rate 2e-4, Adam optimizer, beta1 = 0.5, for 50 epochs.

3. Experiments

N	Loss	pixel-wise acc
1 x 1	**cGAN + L1 Loss**	89.34
70 x 70		88.31
128 x 128		88.57
286 x 286		88.21
70 x 70	cGAN	84.02
	cGAN + L1	88.31
	cGAN + L2	88.52
70 x 70	cGAN + L1, Reversed	77.5

Figure 1. Pixel-wise accuracy for various N values and loss functions; given as percent similarity of generated image to ground truth test image. Values are quite similar for different N values which is surprising since images look qualitatively different to the human observer. Unsurprisingly, adding L1 or L2 loss encourages the model to produce images that are more similar to the ground truth image and cGAN loss alone.

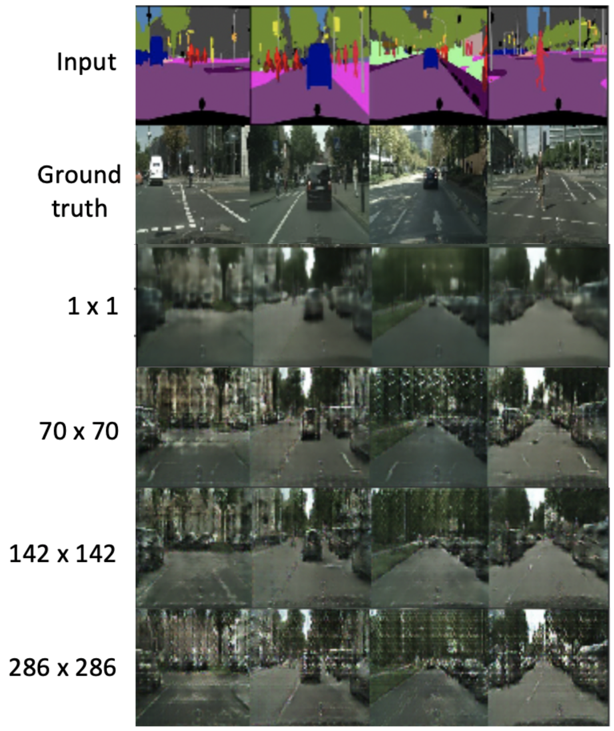

Figure 2. Qualitative results for various N x N discriminator receptive field sizes (r.f.s). When the receptive field is restricted to 1 pixel, images look blurry because each pixel is considered alone without surrounding context. As the receptive field size increases, the discriminator takes a larger patch of the image with greater context when assessing if it is real or fake, thus images tend to look more realistic. Looking specifically at images of 70 x 70 versus 142 x 142 r.f.s., we start to see that a smaller r.f.s. produces crisper fine-details, but overall images don't look as realistic. For example, tree leaves look more detailed in former, while overall images looks smoother and more realistic in the latter.

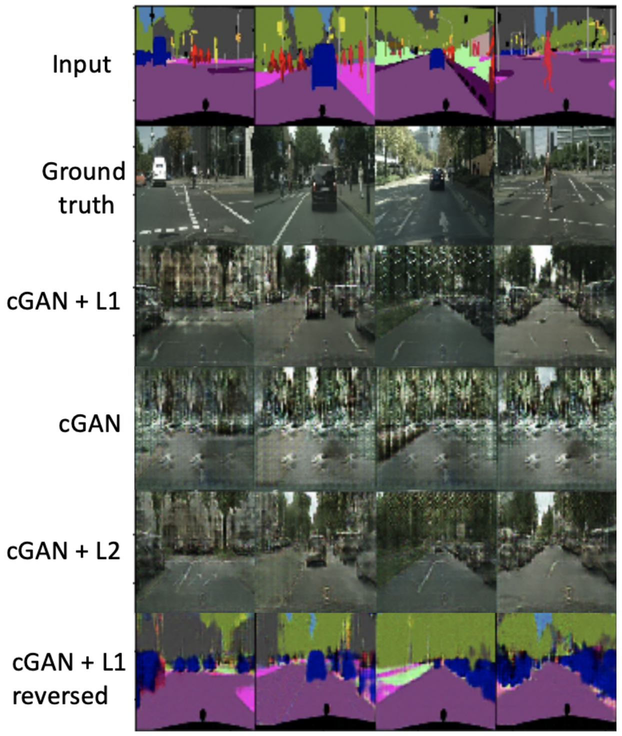

Figure 3. Qualitative results for various loss functions. By inspection, cGAN loss alone performs the worst, while cGAN + L1 or L2 performs relatively well. L1 loss produces images that are notably crisper than images created by models trained with L2 loss. This is because of the nature of L2 loss to encourage more mid-tones and blur. When we switch x and y, we are able to translate images the other way.

4. Conclusions and Future Directions

We saw a trade-off between receptive field size and fine-grained details versus overall image quality. In the future, it would be interesting to train two discriminators, one with a small r.f.s. and another with a large r.f.s. and sum their results in the loss function. This way, much like adding L1 or L2 loss to the overall loss equation, we would be able to encourage the network to produce images with nice details and an overall cohesive look. This has not been done by the authors and we suggest it as a great extension to this paper. Along these lines, we observed that the cGAN renders roads, trees, and cars quite well but does a very poor job at rendering people, and oftentimes completely ignores them. This is likely because people are much smaller objects and do not weigh as heavily in calculation of the loss. Hopefully inclusion of an additional smaller r.f.s. discriminator will help resolve this problem, and we suggest looking out for this as a possible finding of the above stated experiment.

5. References

References

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. doi: 10.1109/cvpr.2017.632. URL https://arxiv.org/pdf/1611.07004.pdf#page=15&zoom=100,0,530.

Call for papers: Ijcv special issue on generative adversarial networks for computer vision. URL http://people.csail.mit.edu/junyanz/ijcvgans.html.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11):5464–5478, 2019. doi: 10.1109/tip.2019.2916751.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/cvpr.2016.350.