Image-to-Image Translation with Conditional Adversarial Networks

Pavan Kumar Reddy Pinnapureddy (180050073) Dorna Vineeth (180050030)
Mavuri Siva Krishna Manohar (180050057)

December 6, 2020

Abstract

We investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing maps from aerial images. This work suggests we can achieve reasonable results without hand-engineering our loss functions either.

1 Introduction

1.1 Problem

Many problems in image processing, computer graphics, and computer vision can be posed as "translating" an input image into a corresponding output image.Each of these tasks has been tackled with separate, special-purpose machinery, despite the fact that the setting is always the same: predict pixels from pixels. The use of conditional generative adversarial networks(cGANs) aims to develop a common framework for all these problems.Here, we focus on its application for translation of aerial images to maps.

1.2 Background

Image-to-image translation problems are often formulated as per-pixel classification or regression. These formulations treat the output space as "unstructured" in the sense that each output pixel is considered conditionally independent from all others given the input image. Conditional GANs instead learn a **structured loss**. Structured losses penalize the joint configuration of the output. The conditional GAN is different in that the loss is learned, and can, in theory, penalize any possible structure that differs between output and target.

For our generator we use a "U-Net"-based architecture , and for our discriminator we use a convolutional "PatchGAN" classifier, which only penalizes structure

at the scale of image patches. We will see an elaborate view of them in further sections. This approach is effective on a wider range of problems.

2 Algorithm

2.1 Architectural details

2.1.1 U-net

U-net architecture can be broadly thought of as an encoder network followed by a decoder network. Unlike in classification where the end result of the the deep network is the only important thing, it provides a mechanism to project the discriminative features learnt at different stages of the encoder onto the pixel space. The decoder consists of upsampling of previous decoder layer and concatenation of corresponding encoder layers followed by regular convolution operations.

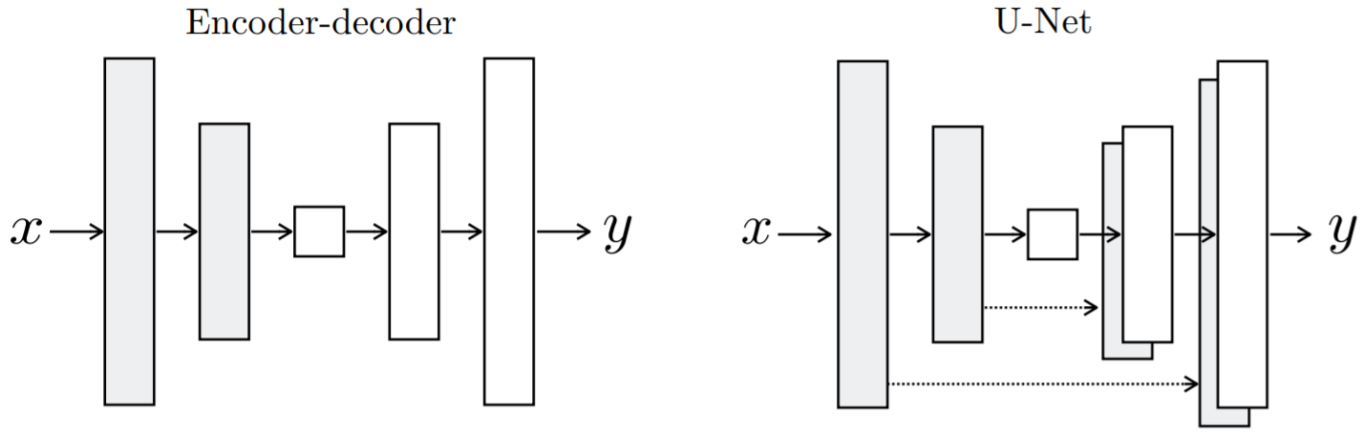


Figure 1: Two choices for the architecture of the generator. The "U-Net" is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks

2.1.2 PatchGAN

PatchGAN is a type of discriminator for generative adversarial networks which only penalizes structure at the scale of local image patches. The PatchGAN discriminator tries to classify if each patch in an image is real or fake. This discriminator is run convolutionally across the image, averaging all responses to provide the ultimate output of . Such a discriminator effectively models the image as a Markov random field, assuming independence between pixels separated by more than a patch diameter. It can be understood as a type of texture/style loss.

2.2 Objective

GANs are generative models that learn a mapping from random noise vector z to output image y, G : z → y. In contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to y, G : {x,z} → y. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes".

The objective of a conditional GAN can be expressed as:

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

where G tries to minimize this objective against an adversarial D that tries to maximize it, i.e.

$$G^* = argmin_G max_D \mathcal{L}_{cGAN}(G, D) \tag{2}$$

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance. The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense. We also explore this option, using L1 distance rather than L2 as L1 encourages less blurring:

$$\mathcal{L}_{L1}(G) = E_{x,y,z}[||y - G(x, z)||_1] \tag{3}$$

Our final objective is

$$G^* = argmin_G max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{4}$$

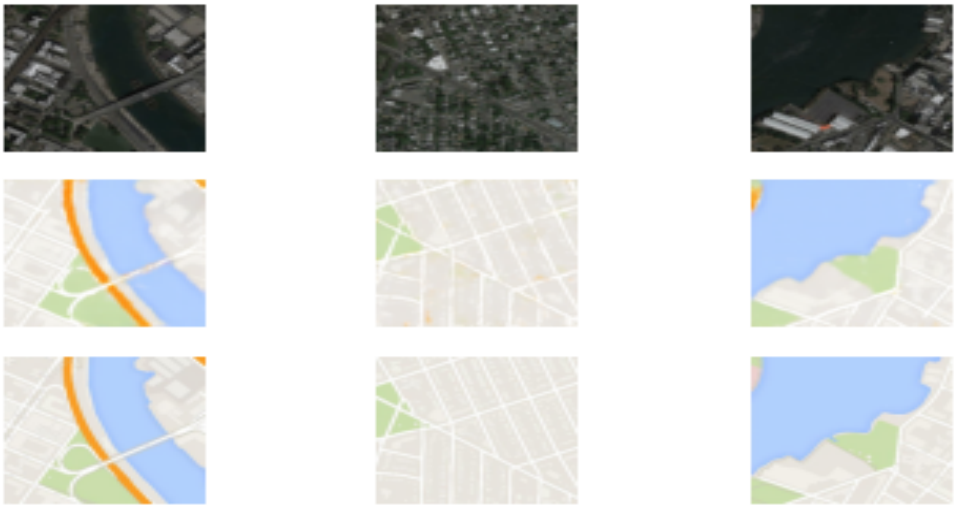
3 Experiment

We have implemented this network and used this for image translation from aerial images to maps and vice versa. We have also studied the impact of using U-net instead of simple encoder-decoder.

3.1 Aerial images to maps with different values of epochs

(a) epochs=10

(b) epochs=20



(a) epochs=50

(b) epochs=100

Figure 3: Results obtained for Aerial images to Maps for different number of epochs
Legend: Row 1 - input images, Row 2 - output images, Row 3 - Target images

3.2 Maps to Aerial images with different values of epochs

(a) epochs=20

(b) epochs=40

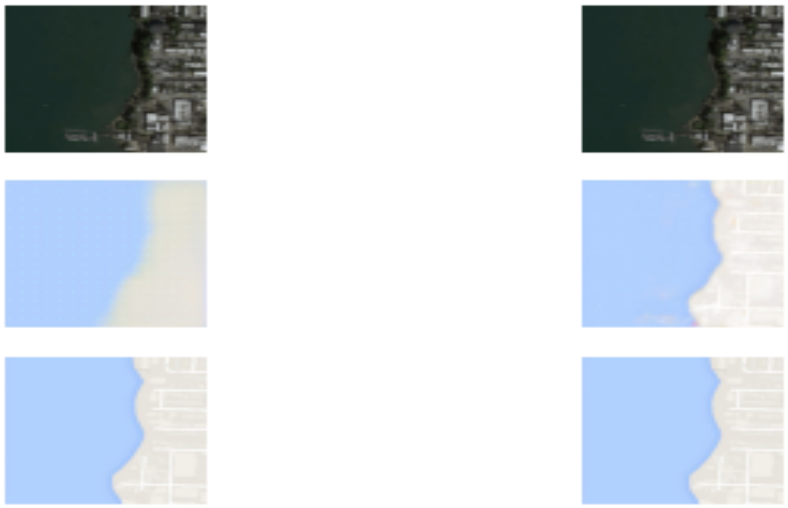
(a) epochs=70

(b) epochs=100

Figure 5: Results obtained for Maps to Aerial images for different number of epochs
Legend: Row 1 - Target images, Row 2 - Output images, Row 3 - Input images

3.3 Comparision of results with Unet (L1+cGAN) and with pure encoder-decoder (L1+cGAN)

(a) epochs=20



(b) epochs=50

Figure 6: Results obtained with U-net vs with pure encoder-decoder
Legend: Left - with pure Encoder-Decoder, Right - with U-net

3.4 Plot of loss with respects to epochs

4 References

1. Image-to-Image Translation with Conditional Adversarial Networks
   https://arxiv.org/pdf/1611.07004.pdf
2. Understanding Generative Adversarial Networks (GANs)
   https://towardsdatascience.com/understanding-generative-adversarial-n
3. U-net architecture
   https://developers.arcgis.com/python/guide/how-unet-works/
4. PatchGAN
   https://arxiv.org/pdf/1611.07004v3.pdf
