Pix2Pix GAN based Image translation

A Project Report

Presented to

Prof. Nick Brown

Multidisciplinary Graduate Engineering Programs
Northeastern University

As a portfolio project requirement for the Class
INFO6105

By

Aniruddha Tambe (002101113)
Kinjal Thakkar (001568960)

Dec, 2021

Abstract

To advance the domain knowledge of artificial neural network computing, understanding & implementing Generative Adversarial Networks (GANs) is crucial. This portfolio project involves implementing a Pix2Pix model–a conditional GANs model - by programming the key concepts of encoders, decoders, and generator & discriminator models using Keras. The composite model is then run for predefined calculated training steps. The results of this implementation are saved over 100 epochs as model.h5, after every 10 epochs. This model can be used on unseen Satellite image data to generate faithful translation as Google Map images.

Introduction

We explore conditional GANs as an approach to the image-to-image translation problem. By using 3 different types of loss function, GANs are able to effectively model the relation between source and target image. We used the Berkeley maps dataset to train a variation of the above model. This dataset consists of 1097 training examples and 1099 validation examples, each with a resolution of 1200x600.

Training process

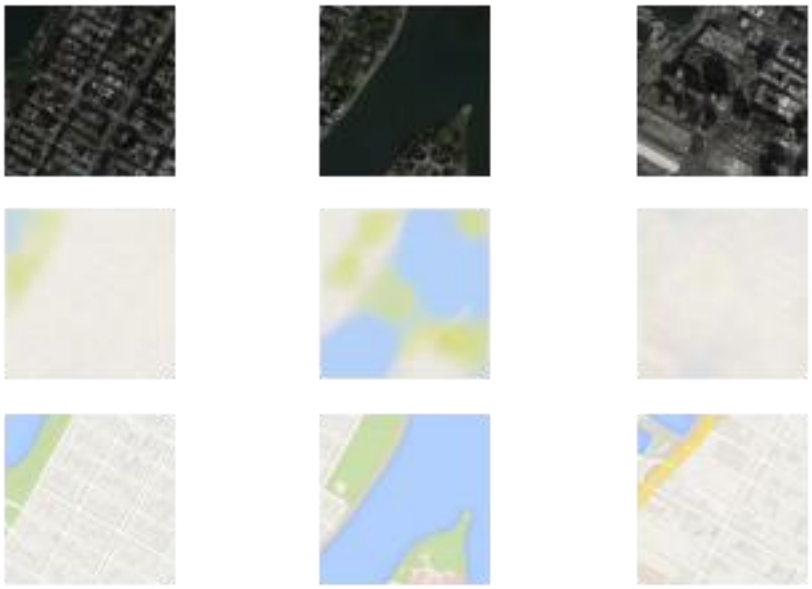

Figure 1: Scaled image generated at the 98th epoch

Image preprocessing

The Berkeley image dataset needed to be decoupled, to retrieve the Satellite source image and its corresponding target maps image, in the form of numpy arrays. These images are then used to compute the loss, to improve the model's efficacy.

Cost function and Objective

To effectively “fool” the discriminator into classifying the fake generator outputs as real image, we train generator and discriminator differently. The discriminator model is trained using the real images, whereas the generator is not. The generator is trained to reduce the adversarial loss and the L1 loss (difference between generated image and expected output). The generator can also be updated using the weighted sum of adversarial loss and the L1 loss, to yield the better results.

The cost or objective of a conditional GAN can be expressed as,

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

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

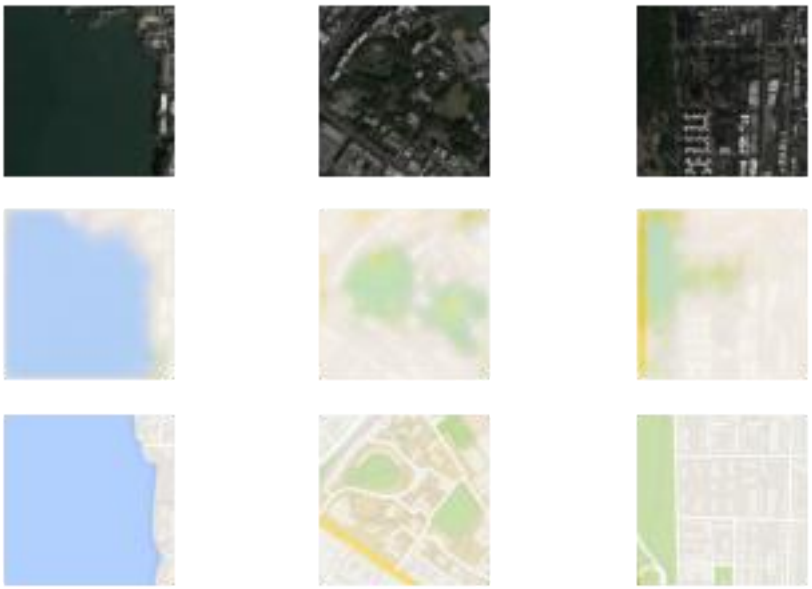

Figure 2: Scaled image generated at the 109th epoch – some semblance of water bodies and land mass is more visible

Architecture

The architecture is comprised of 2 models, discriminator & generator. The discriminator is a DCNN, whose design is based on the 70×70 PatchGAN model, such that it is scalable to images of any sizes. The generator model follows the U-net architecture such that it consists of an encoder & decoder, to effectively downsample and upsample the input image through a bottleneck layer.

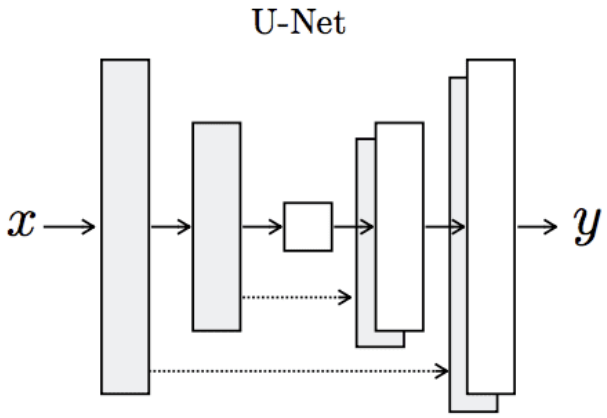

Figure 3: U-net model used the generator model in CGANs

The encoder and decoder of the generator model are blocks of standard layers convolutional, batch normalization, and dropout and activation layers.

Execution flow

1. Importing the python libraries
2. Load dataset
    a. Call load_images(path,size=(256,512))
       Function returns numpy array of decoupled images
3. Display the decoupled images
4. Create the GAN
    a. Define the encoder block
       Add Conv2D, batch normalization and activation layers
    b. Define the decoder block
       Add Conv2D, batch normalization, dropout and activation layers
    c. Define generator
       Add encoder, bottleneck & decoder model
    d. Define discriminator

Add multiple conv2D + relu layers , batch normalization layers, sigmoid activation layer

5. Utility functions
    a. Define summarize performance
       Function summarized performance on each epoch
    b. Define generate fake samples
       Function returns a batch of fake images
    c. Define generate real samples
       Function returns a batch of real images
6. Training the model
    a. Set the discriminator & generator
    b. Set the composite GAN model
    c. Call the train model by passing above models
    d. Run model for specified epochs – Save each 10^{th} iteration as a model.h5 & output image files

Challenges

The conditional GANs architecture lacks the capacity to generate "truly" HD images with resolutions comparable to 2K or 4K.

Conclusion

After using the techniques mentioned in the "Image-to-Image Translation with Conditional Adversarial Network" paper, we can justify the efficacy of the pix2pix model (generator-discriminator) in effectively generating a 256x256 target image of relatively close likeliness.

References

1. Image-to-Image Translation with Conditional Adversarial Networks
   https://arxiv.org/pdf/1611.07004.pdf
2. Sat-to-map github repository
   https://github.com/shagunuppal/sat_to_map
3. Generating Fake Book Pages Using a GAN
   http://anderff.com/blog/2019/11/01/deepzine/
4. How to Develop a Pix2Pix GAN for Image-to-Image Translation
   https://machinelearningmastery.com/how-to-develop-a-pix2pix-gan-for-image-to-image-translation/