CS 535 Pattern Recognition
cGAN for Image-to-Image Translation
(Satellite --> Google maps)

Yashshree Patil
Guided by Vladimir Pavlovic

Introduction

Image-to-image translation aims to learn the mapping between two visual domains here satellite to google map image.

In this project cGAN (Conditional adversarial networks) acts as a solution to the image-to-image translation problem of satellite to map translation.

Approach used is Pix2Pix GAN for image translation.

RELATED WORK

Most of the work on Image to image translation is done with unconditional GANs, UNIT or multi - hop GAN , relying on L1, L2 or regression terms.

I am not the first to apply GANs in the conditional setting. Prior and concurrent works have conditioned GANs on discrete labels, text, and images. Their work differs in architectural choices for the discriminator and generator. I have used "U-Net" Architecture for generator and convolutional “PatchGAN” classifier for discriminator, a similar architecture appeared in https://arxiv.org/pdf/1611.07004.pdf . But unlike them, my focus is more on the image-to-image translation of Satellite to Google map and vice versa, we can say that it is a recreation of cGAN image-to-image translation https://arxiv.org/pdf/1611.07004.pdf focusing more on satellite and google map images.

Pix2Pix GAN

Why this approach?

- Image-to-image translation requires specialized models and hand-crafted loss functions.
- Pix2Pix GAN provides a general purpose model and loss function for image-to-image translation.
- The Pix2Pix GAN is demonstrated on a wide variety of image generation tasks, including translating photographs from day to night and products sketches to photographs from summer to winter.

Dataset

Maps DATASET:
http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/maps.tar.gz
This is a dataset comprised of satellite images of New York and their corresponding Google maps.

- There are a total of 2195 satellite-google map image pairs. Such that 1096 image pairs are in the Train folder and 1099 image pairs are in the validation folder.
- Each image contains both a google maps image and its corresponding satellite image.
- (Satellite image on the left and google map image on the right)
- Dimensions of the images are 1,200 pixels in width and 600 pixels in height.

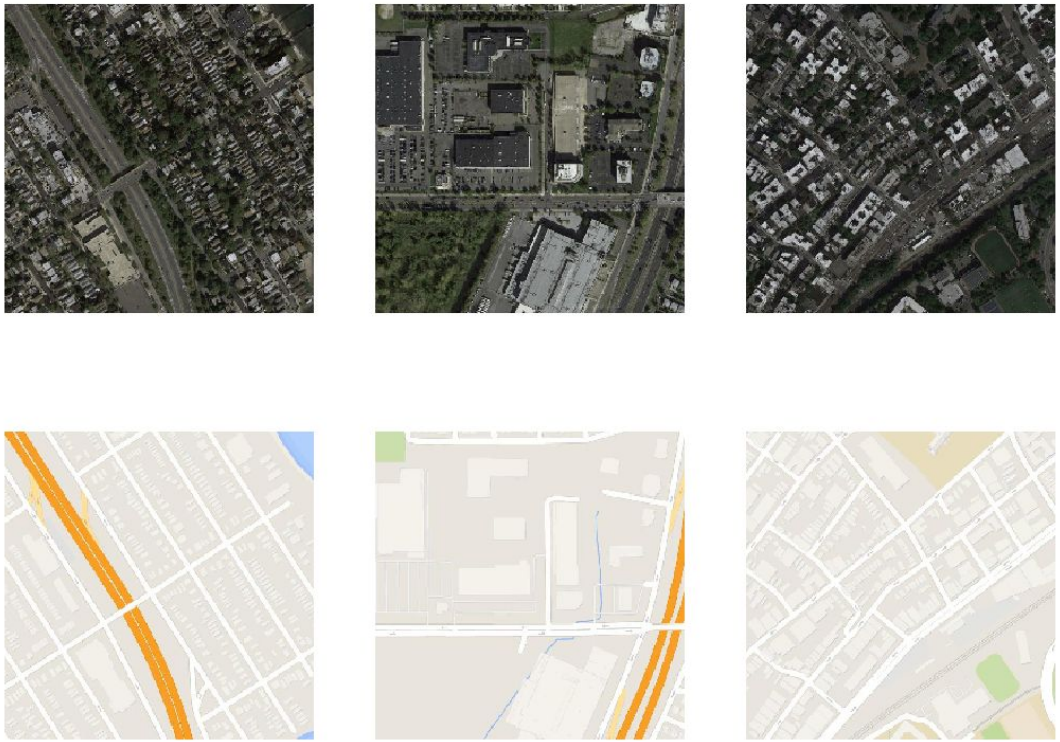
DATASET IMAGE

DATA Pre-Processing

Each image consists of both google maps and satellite images hence we need to prepare the dataset for proper training (by separating the satellite and google maps images).

This can be done by using Pix2Pix GAN model in Keras. The images in the training dataset will be loaded, rescaled, and split into the satellite and Google map elements and the resulting image will be 1096 separate image pairs with the dimension of 256x256 pixels.

AFTER PRE-PROCESSING 3 image pairs look like this



Model Architecture

The GAN architecture is consists of a generator model for creating new plausible synthetic images, and a discriminator model that classifies images as real (from the dataset) or fake (generated).

DISCRIMINATOR

The discriminator defines the relationship between one output of the model to the number of pixels in the input image.

PatchGAN model, it is based on an effective receptive field of the model, which gives a correlation between an output patch to the number of pixel in the input picture, it is designed in such a way that the output prediction model is patched to 70x70 square. This approach can be widely used because this model can take up images of different sizes.

The output of the discriminator gives the probability or likelihood that the patch of the input image is real.

GENERATOR

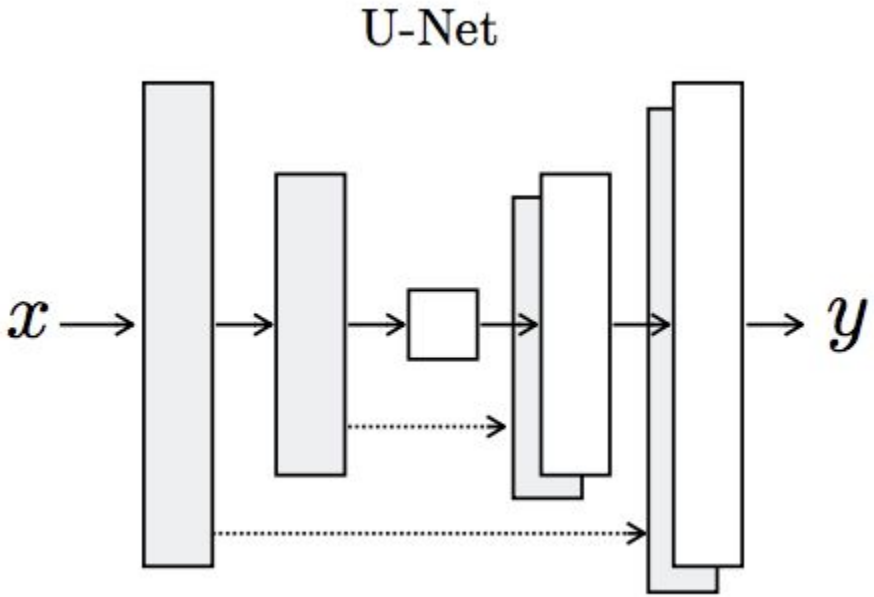
The generator is a complex encoder-decoder model with a U-Net architecture. The input of the generator model is the source image (satellite image). The output of the generator model is the target image (google maps image)
This is achieved by the following process:

1. The image is first down-sampled through the U-Net Architecture to the bottleneck layer this process is also known as encoding.

2. Then it is upsampled through the bottleneck representation to the output image size this process is also known as decoding.

The overall shape of this architecture looks like an alphabet "U" hence, known as U-net architecture.

How is U-net different from the simple encoder-decoder architecture of the generator? The U-net architecture has skip connections between the encoding and the decoding layers.



COMPOSITE MODEL

This logical or composite model involves stacking the generator on top of the discriminator. A source image is provided as input to the generator and to the discriminator, although the output of the generator is connected to the discriminator as the corresponding "*target*" image. The discriminator then predicts the likelihood that the generator was a real translation of the source image.

TRAINING

The discriminator model is trained directly on real and fake images, whereas the generator model is trained via the discriminator model.

It is updated to minimize the loss predicted by the discriminator for generated images marked as "*real*." As such, it is encouraged to generate more real images.
The generator is also updated to minimize the L1 loss or mean absolute error between the generated image and the target image.

TRAINING

Taking the defined generator, discriminator, composite model, and loaded dataset as input. The number of epochs is set at 100, A batch size of 1 is used.

we are going to use a fixed iteration of 100 epochs. The totals images available in the train data are 1096. 1 epoch means running over all the images once (i.e. 1 iteration through all the images in train dataset) batch size of 1 means 1096 training steps. The generator model is saved and evaluated at every 10 epochs i.e. 10 epochs covers 10960 training steps. therefore, to complete 100 epochs there will be a total of 109600 training steps to be completed.

TRAINING

Training steps:

1. In each training step a batch of real examples are selected at random, then a generator is used to generate a batch of matching generated (Fake) samples using the real image source from the dataset.
2. The discriminator is then updated with a batch of fake and real images.
3. the generator model is updated providing the real source images as input and providing class labels of 1 (real) and the real target images as the expected outputs of the model required for calculating loss. The generator has two loss scores as well as the weighted sum score.
4. The weighted sum score is more important to us as it is used to update the model weights.
5. The loss of each iteration is stored and the performance of the model is evaluated after 10 training epochs i.e 10960 training steps.

We have 3 types of Loss (d1, d2, and g)

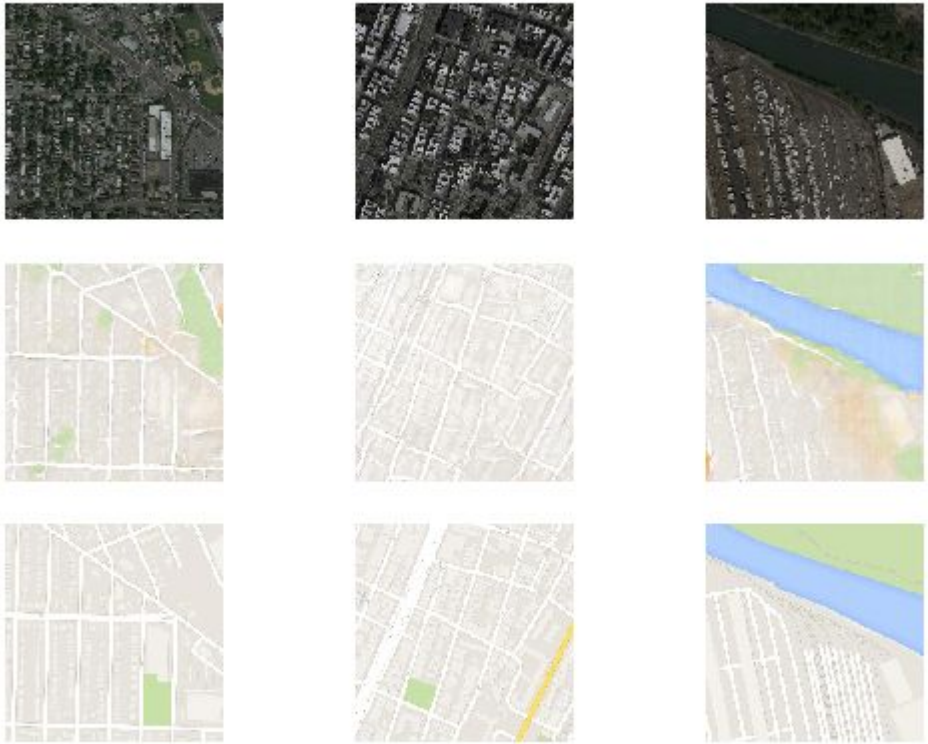
d1 - discriminator loss (real image example)

d2 - discriminator loss (Generated/fake image example)

g - weighted average of cGAN and L1 loss

```
>1, d1[0.566] d2[0.520] g[82.266]
>2, d1[0.469] d2[0.484] g[66.813]
>3, d1[0.428] d2[0.477] g[79.520]
>4, d1[0.362] d2[0.405] g[78.143]
>5, d1[0.416] d2[0.406] g[72.452]
...
>109596, d1[0.303] d2[0.006] g[5.792]
>109597, d1[0.001] d2[1.127] g[14.343]
>109598, d1[0.000] d2[0.381] g[11.851]
>109599, d1[1.289] d2[0.547] g[6.901]
>109600, d1[0.437] d2[0.005] g[10.460]
>Saved: plot_109600.png and model_109600.h5
```

AFTER 10 TRAINING EPOCHS



AFTER 100 TRAINING

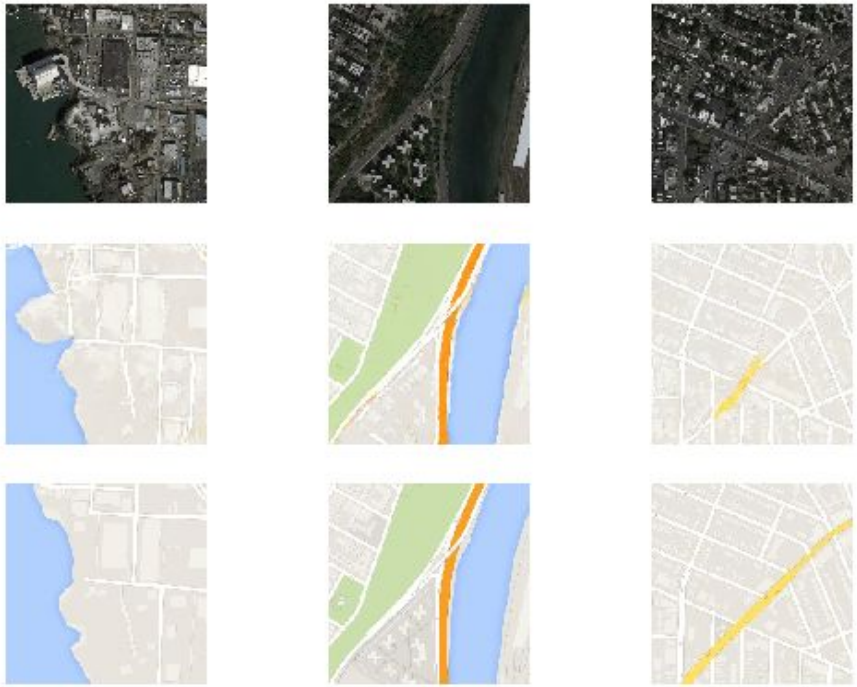


IMAGE to IMAGE translation

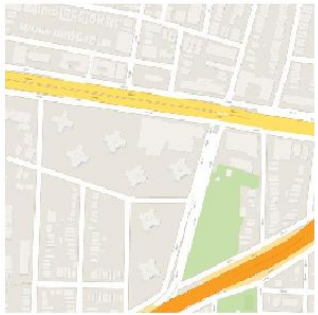
To use these models in image-to-image translation we need to load the most accurate model, training dataset, and choose a random image pair to test on the model. For now, the image chosen will be from the training dataset.

OUTPUT

Source

Generated

Expected



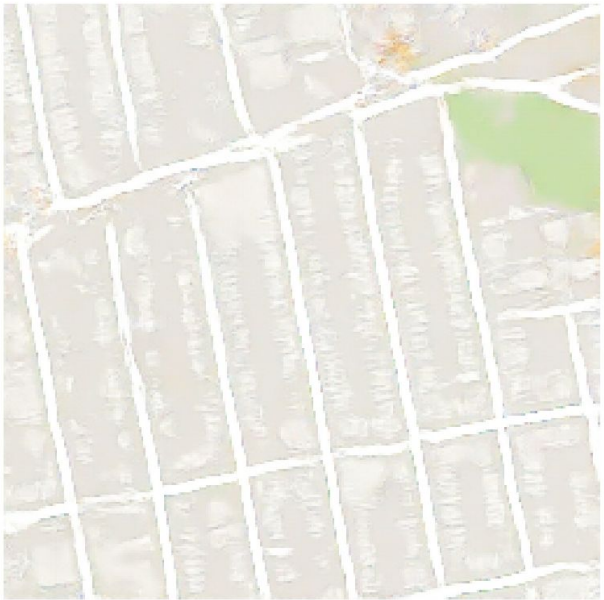
Stand alone image

For standalone an image was selected from the validation dataset and the satellite element of the image was cropped and pasted/saved in the working directory and used as input to the model.

We must load the image as a NumPy array of pixels with the size of 256×256, rescale the pixel values to the range [-1,1], and then expand the single image dimensions to represent one input sample.

Running the example will load the image from the file, creates a translation google maps image for it, and display the output on the screen.

The generated image appears close to the accurate translation of the input image. The streets are still not in straight lines and not all buildings are visible clearly. With more training (increasing epochs) or choice of a different model (for generator/discriminator), higher-quality images can hopefully be generated.

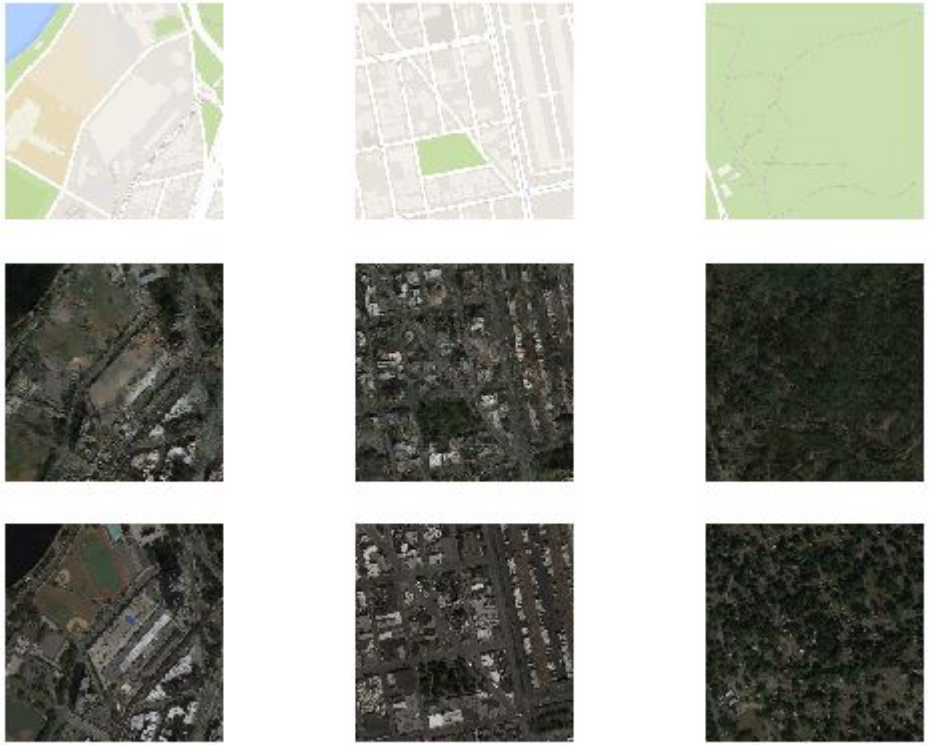


Google maps to Satellite

After developing a good translation model for Satellite to google maps translation, it is quite easy to build a model that transforms Google maps to the satellite.

To achieve this I had used the same code to train the model with only one difference while loading the dataset/the real samples switch the satellite images with the google maps images and the model will be trained such that when given a google map image it would develop its corresponding satellite image.

Google maps to satellite image



THANK YOU