Using Generative Adversarial Networks to Produce Synthetic Overhead Imagery

Cyber Security Engineering

Bawer Alissa, Twinkle Gera, Amir Itayem, Adalid Helguero, and Mohammad Saad

Sponsors: Tim Parker and Jonathan Brant

Background

- **Generative Adversarial Networks (GAN):** are neural networks in machine learning which are composed of a discriminar and a generator.
  - These two model are train against one another with the rules of a zero sum game, where a loser and winner is always involved
  - Progressive GANs and Deep Convolutional GANs are two different types of architecture in producing synthetic images
  - These two different architecture involve the use of convolutional neural networks.
  - In order to produce the best synthetic images, both were tested.
- **Convolutional Neural Network (CNN):** are neural networks specifically designed to take images as input to recognize edge detection and any other image details
- **Deep Convolutional GAN:** DCGAN is a design architecture for GANs which also uses transpose convolutional layering and strides in order to upsample and downsample the input
- **Progressive GAN:** One of the latest design of a GAN architecture and builds of a DCGAN. A progressive GAN downsample an image to a small size and trains until sufficient learning takes places. It then increase the image dimensions and repeats the same steps again until the desire image shape is produce.
  - Progressive GANs are used to help produce images of higher resolution because it increases the stability of the models.

Objectives and Materials

The objective of this project was to produce a realistic synthetic image using a DCGAN and/or a Progrossive GAN

Materials

- **Tensorflow:** is a deep learning frankwork for machine learning.that was used.
- **Keras:** is a high API that uses Tensorflow as a backend. Keras was used because of the simplicity and the intuitive in training models
- **What is xView?:** xView is one of the largest public datasets of overhead images. These images that are taken from around the world contains complex scenes. xView believes on progressing learning efficiency, improve detection of fine-grained classes, and reduce minimum resolution.
- **How large are the overhead images?:** The overhead images usually are 2900 x 3100. However, for our project those images have very large dimensions. This could be beneficial for the accuracy of machine learning, but having a dataset will cut down the training rate drastically. Our solution was adding a chipping function that is utilized to downsample the images into smaller image of the size of 256 by 256 pixels.

Generator & Discriminator

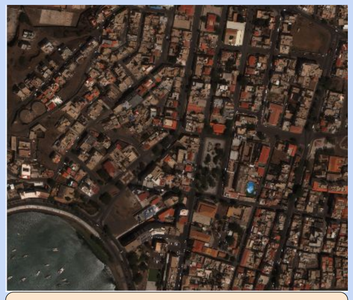
Generator: The generator is integrated into the GAN. First the generator will build images using the architecture. It will start from a 100 dimensional noise vector. The first layer adds a fully connected layer called “dense” into the generator architecture. The desired image size is 256x256x3 pixels. The dimensions consist of the length and width of the image. The last dimension shows if it’s in color or grayed scale

Discriminator: The discriminator inspects the images produced by the generator and determines whether the images are authentic or synthetic. These results are provided back to the generator, which uses this data to make synthetic imagery more realistic until it can deceive the discriminator. The discriminator penalizes the generator if it produces implausible results. The discriminator then takes an image and applies convolutional layers until a one dimension noise vector is left. Finally, the discriminator uses the vector to predict whether the image is synthetic or real

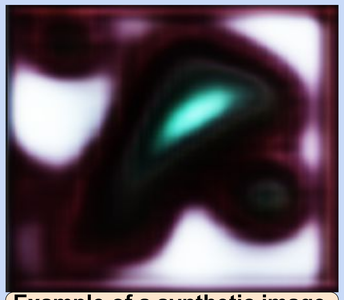
Training Method

DCGAN_- When the DCGAN start training, it first sends a random noise vector of 100 randomly generated image samples to the generator. Afterwards, a variable is created which contains the image data of fake and real images. While training, we used a batch size of 100. This batch contains 100 randomly selected images from our entire chipped dataset. These images are what help both of our models to learn how to generate synthetic images. We sample 100 random images 22 times and this would be one epoch. We train the DCGAN usually for 50 epochs.

Progressive GAN - The progressive GAN starts by convoluting an 8x8 image for the first few epochs. Both, the generator and discriminator, will upsample up to 16x16 and so on during training based on the epoch. A layer is added and the image size is upsampled repeatedly until 256x256 size images are being generated. At this point, we continue training until we reach around 50 to 70 epochs.

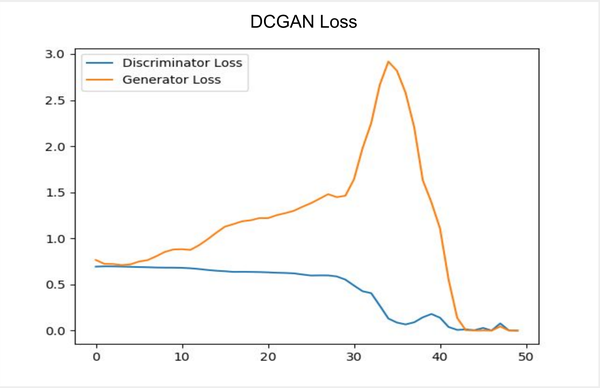


Example of an xView image

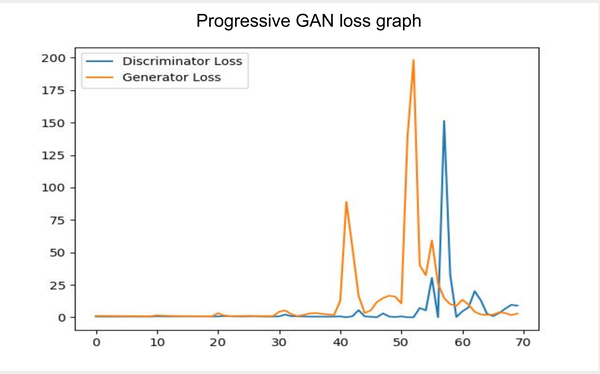


Example of a synthetic image during training

Loss Graphs



Graph above depicts the binary loss of our DCGAN. This measured the loss for each epoch using a binary cross-entropy function for 50 epochs.



Graph above depicts the binary loss of our Progressive GAN. This measured the loss for each epoch using a binary cross-entropy function for 60 epochs.

References

msaad4@gmu.edu ahelguer@gmu.edu aitayem@gmu.edu tgera@gmu.edu

balissa@gmu.edu jonathan.c.brant@lmco.com tim.parker@lmco.com