

An Overview of Map Synthesis with Cartographic Design Using Aerial Images

Meiliu Wu (mwu233@wisc.edu), Yilei Hu(yhu366@wisc.edu)

Contents

1. Summary of current progress in methodology	2
1.1. Dataset preparation.....	2
1.2. Implementation of pix2pix for translating aerial images into Google Maps tiles	2
1.2.1. The basics of pix2pix model.....	2
1.2.2. Loss functions	3
1.2.3. The U-Net generator	3
1.2.4. The PatchGAN discriminator.....	4
1.3. Implementation of CycleGAN for two-way image translation	4
1.3.1. The basics of CycleGAN model.....	4
1.3.2. Loss functions	6
1.3.3. The difference between CycleGAN generator and pix2pix generator	7
1.3.4. The difference between CycleGAN discriminator and pix2pix discriminator.....	8
2. Current results.....	8
2.1. Comparison of ground truth and pix2pix maps with different epoch numbers	8
2.2. Two-way results of CycleGAN with different iterations.....	11
2.3. Project webpage (link)	12
3. Difficulties during the implementation.....	12
3.1. Evaluating the results: qualitative vs. quantitative?.....	12
3.2. Model training run on CPU hardware.....	12
4. Major changes compared with our proposal.....	13
5. Further efforts.....	13
5.1. Adding other GAN-related models in the comparative analysis	13
5.2. Comparing the advantages and disadvantages among different models.....	13
References	14

1. Summary of current progress in methodology

1.1. Dataset preparation

The training dataset contains 1,096 images with the same size 600x600, and testing dataset contains 1,098 images with the same size 600x600, which are available to the public contributed by Isola *et al.*, 2017 and Zhu *et al.*, 2017.

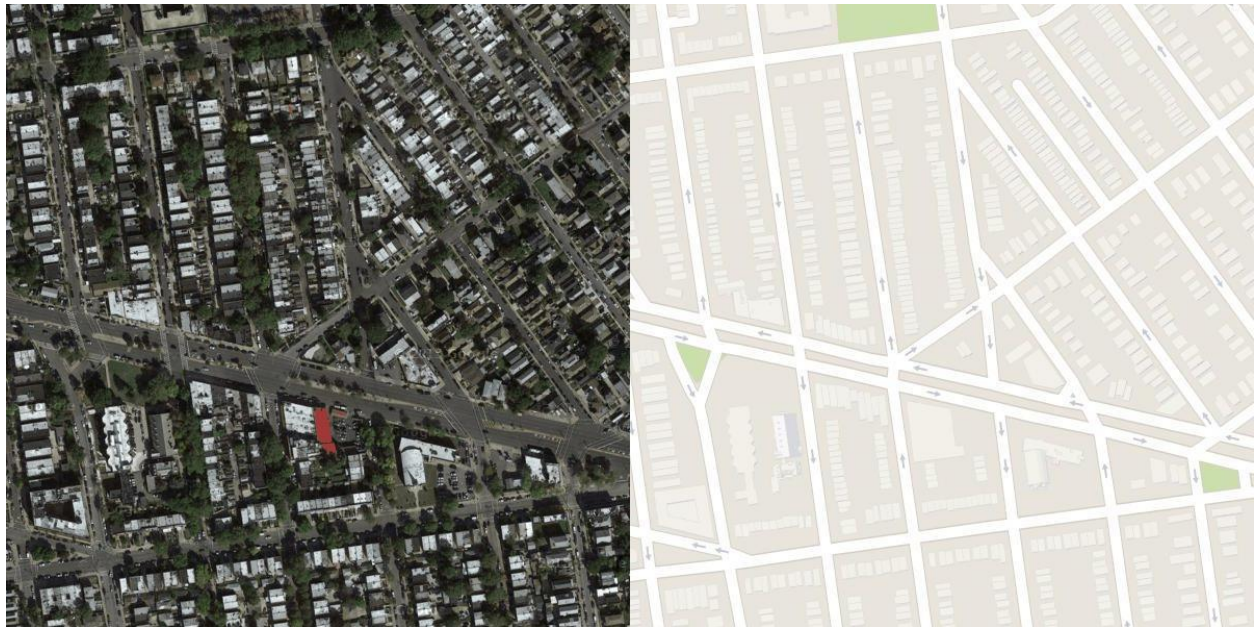


Figure 1. An aerial image from Google Earth Engine (left) along with the corresponding Google Maps tile (right). Images were sampled in New York City, and split into training and testing by the center latitude of the region, with a buffer region added to ensure that there is no overlap between training dataset and testing dataset.

1.2. Implementation of pix2pix for translating aerial images into Google Maps tiles

1.2.1. The basics of pix2pix model

The pix2pix model is a Generative Adversarial Network (GAN) model for image-to-image translation that could be applied to various scenarios, e.g., semantic labels \leftrightarrow photo, map \leftrightarrow aerial image, sketch \rightarrow photo, day \rightarrow night, and so on (Isola *et al.*, 2017). As described in our proposal, the architecture of the original GAN model consists of a generator for generating “fake” images that looked real and a discriminator that is fed with real images or generated images, and then determines whether the input images are fake or not. The generator is able to be updated based on the output of the discriminator, while the discriminator could be updated in a standalone manner. These two components of GAN are trained simultaneously and adversarially, with the fact that the generator tries to fool the discriminator during the updating process while the discriminator also improves its ability to distinguish the real from the fake.

Specifically, the pix2pix model is a conditional GAN (cGAN), since the process of generating an output image is based on a source image. Correspondingly, both the source image and the target image

(either real image or generated image) will be provided as input to the discriminator, which needs to identify whether the target image is a plausible synthetic image based on the source image.

1.2.2. Loss functions

First, the objective of a conditional GAN can be expressed as:

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

In the pix2pix model, the discriminator D takes both a source image (i.e., aerial photo) and a target image (i.e., Google maps image) as input and predicts the likelihood of whether target image is real or a fake translation of the source image, therefore, D seeks the better way to maximize the log likelihood of identifying real and fake images. Furthermore, since the training of the discriminator is faster compared to the generator, the discriminator loss is halved to slow down the training process.

As for the generator G , it tries to minimize the objective above against the adversarial D , but the goal of G is not only to fool the discriminator D but also to produce an output nearest the ground truth in an L2 sense. In practice, we implemented using L1 distance rather than L2 as L1 encourages less blurring (Isola *et al.*, 2017):

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

Therefore, the adversarial loss and the L1 loss are combined into a composite loss function to update the generator model. Specifically, the adversarial loss will have an impact on whether the generator is able to synthesize images that are plausible in the target domain, whereas the L1 loss regularizes the generator to synthesize images that are a plausible translation of the source image. In our implementation, the combination of the L1 loss to the adversarial loss is controlled by a hyperparameter lambda, which is set to [1, 100], indicating 1 to 100 times the importance of the L1 loss than the adversarial loss. Our final objective becomes:

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

1.2.3. The U-Net generator

The generator for the pix2pix model is implemented as a U-Net. The U-Net model is an encoder-decoder model for image translation where **skip connections** are used to connect layers in the encoder with corresponding layers in the decoder that have feature maps with the same sizes. This is because a large amount of low-level information needs to be shared between the input and output, and it would be desirable to deliver this information directly across the net.

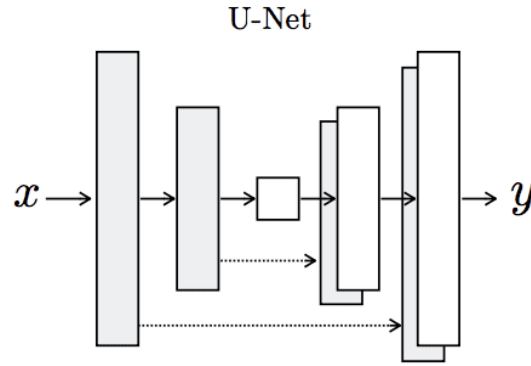


Figure 2. Architecture of the U-Net generator (Isola et al., 2017).

In our implementation, we added skip connections between each layer i and layer $n - i$, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer $n - i$, which is the same setting as the original pix2pix model. Furthermore, **batch normalization** with statistics calculated for each batch and **dropout layers** for randomness are used during both training and inference. The batch normalization is considered as instance normalization, specifically when the batch size is set to 1 in the pix2pix model. The encoder uses blocks of Convolution-BatchNorm-LeakyReLU like the discriminator, while the decoder uses blocks of Convolution-BatchNorm-Dropout-ReLU with a dropout rate of 50%. All convolutional layers use a filter size of 4×4 and a stride of 2×2 .

1.2.4. The PatchGAN discriminator

The discriminator is implemented as a PatchGAN in this project. The PatchGAN is constructed with a certain sized window of the receptive field, which is the relationship between one output activation of the discriminator to an area on the input image. We used a PatchGAN with the size 70×70 , which means that each output maps to a 70×70 square of the input image. The discriminator will classify the 70×70 patches of the input image as real or fake. The advantage of PatchGAN is that the same discriminator can be applied to input images of different sizes.

The output of the discriminator depends on the size of the input image but may be one value or a square activation map of values. Each value is a probability for the likelihood that a patch in the input image is real. These values can be averaged to give an overall likelihood or classification score.

As the original setting (Isola et al., 2017), we built the 70×70 discriminator architecture as C64-C128-C256-C512. After the last layer, a convolution was applied to map to a 1-dimensional output, followed by a Sigmoid function. BatchNorm was not applied to the first C64 layer as an exception to the above setting. All ReLUs were leaky with a slope α as 0.2. Finally, the discriminator was optimized using binary cross entropy, and a loss weight (0.5) mentioned in section 1.2.2. was applied to slow down the training process of the discriminator.

1.3. Implementation of CycleGAN for two-way image translation

1.3.1. The basics of CycleGAN model

The CycleGAN model was first described by Zhu *et al.*, 2017. The advantage of the CycleGAN model is that it can be trained without paired examples, which is necessary in pair-wised models such as the pix2pix model. As such, it does not demand examples of corresponding images before and after the translation in order to train the model. Instead, the CycleGAN model is able to utilize a collection of images from each domain and extract and harness the underlying style of images in the collection in order to perform the translation (Brownlee, 2019). The CycleGAN model is more practical for image translation since paired datasets are not always available.

The architecture of the CycleGAN model is comprised of two generators: one generator (Generator-A) for generating images for the first domain (Domain-A) and the second generator (Generator-B) for generating images for the second domain (Domain-B).

- Generator-A -> Domain-A
- Generator-B -> Domain-B

The two generators perform image translation, with the image generation process conditional on an input image from the other domain. For example, Generator-A takes an input image from Domain-B and Generator-B takes an image from Domain-A.

- Input image from Domain-B -> Generator-A -> Domain-A
- Input image from Domain-A -> Generator-B -> Domain-B

Next, each generator has its corresponding discriminator. The first discriminator (Discriminator-A) takes real images from Domain-A or generated images from Generator-A as input, and determines whether they are real or fake. Similarly, the second discriminator (Discriminator-B) takes real images from Domain-B and generated images from Generator-B as input, and determines whether they are real or fake.

- Real image from Domain-A -> Discriminator-A -> [Real/Fake]
- Input image from Domain-B -> Generator-A -> Discriminator-A -> [Real/Fake]
- Real image from Domain-B -> Discriminator-B -> [Real/Fake]
- Input image from Domain-A -> Generator-B -> Discriminator-B -> [Real/Fake]

The discriminators and generators are trained in an adversarial zero-sum process. The generators learn to better fool the discriminators and the discriminators seek to better detect fake images. Eventually, the CycleGAN model finds an equilibrium during the training process.

Additionally, the generators are regularized to not only synthesize new images in the target domain, but also translate more reconstructed versions of the input images from the source domain. This is achieved by using generated images as intermediate input to the corresponding generator model and comparing the output image to the original images. **The process of an image being passed through both generators is called a cycle.** The two generators are trained together to better synthesize the original source image, referred as **cycle consistency**.

- Input image from Domain-B -> Generator-A -> Generated image from Domain-A -> Generator-B -> Generated image from Domain-B

- Input image from Domain-A -> Generator-B -> Generated image from Domain-B -> Generator-A -> Generated image from Domain-A

An extra quality of the architecture is **identity mapping**, meaning that a generator is provided with input images from the target domain and is expected to generate the same image without change. This quality of the architecture results in a better matching of the color profile of the input image.

- Input image from Domain-A -> Generator-A -> Generated image from Domain-A
- Input image from Domain-B -> Generator-B -> Generated image from Domain-B

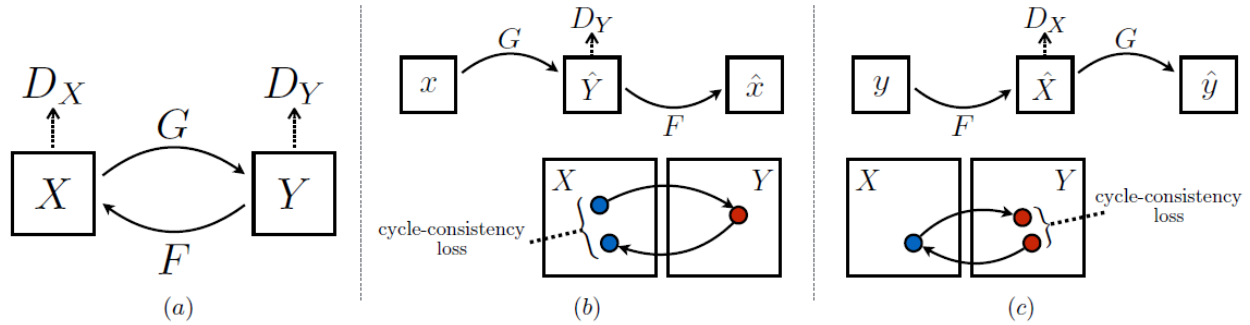


Figure 3. CycleGAN model contains two mapping functions $G: X \rightarrow Y$ and $F: Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X . D_Y encourages G to translate X into outputs indistinguishable from domain Y , and vice versa for D_X , F , and X . To further regularize the mappings, two “cycle consistency losses” are introduced so that if we translate from one domain to the other and back again we should arrive where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$ (Zhu et al., 2017).

1.3.2. Loss functions

The loss functions in the CycleGAN model are constructed based on the core logic of the pix2pix model. First, the Adversarial Loss for the mapping function $G: X \rightarrow Y$ and its discriminator D_Y is defined as (Zhu et al., 2017):

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(Y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(X)} [\log (1 - D_Y(G(x)))].$$

Next, the Cycle Consistency Loss is expressed as:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(X)} [\|x - F(G(x))\|_1] + \mathbb{E}_{y \sim p_{data}(Y)} [\|y - G(F(y))\|_1].$$

Finally, the full objective is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F).$$

where λ controls the relative importance of the two objectives. The ultimate goal is to achieve:

$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

1.3.3. The difference between CycleGAN generator and pix2pix generator

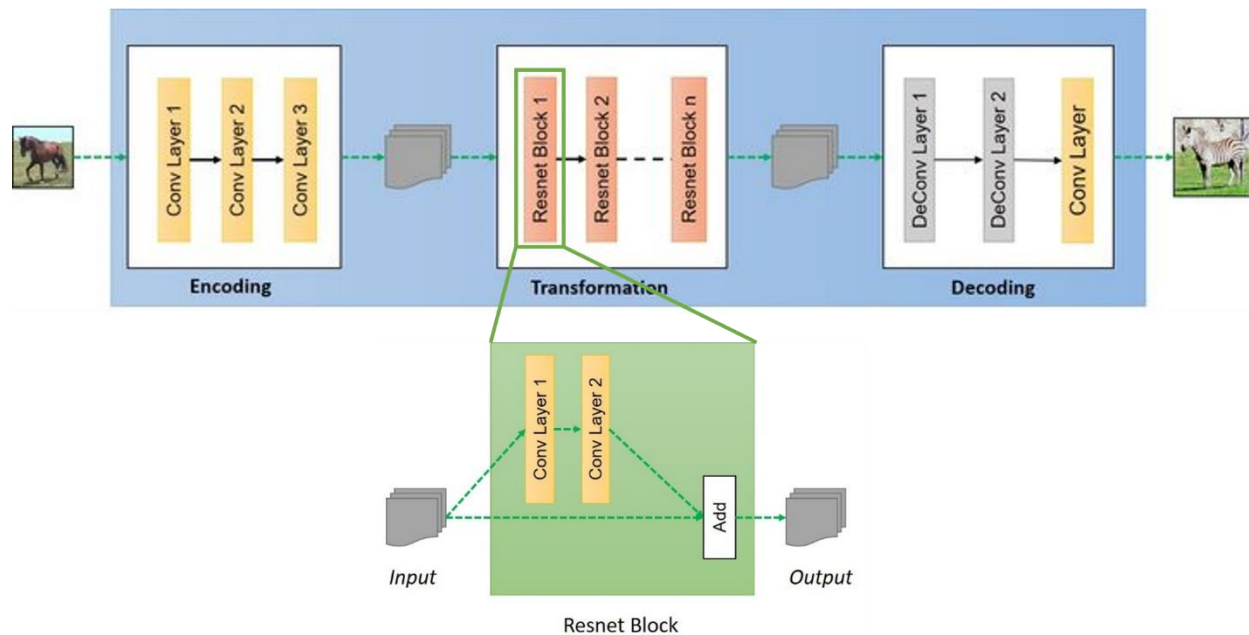


Figure 4. The architecture of CycleGAN generator and the structure of Resnet Block

The CycleGAN generator is also an encoder-decoder U-Net architecture. The generator first encodes the input image down to a bottleneck layer, then interprets/transforms the encoding with a number of **ResNet layers** that use **skip connections**, followed by a series of layers that decode the representation to the size of the output image. To implement this architecture, we defined a function called **ResNet blocks**. Each block consists of two 3×3 CNN layers where the input to the block is concatenated to the output of the block, channel-wise (Brownlee, 2019). Some input is directly added to the output, making sure that the differences between input and output could be narrowed down in order to maintain the features from the input image, such as location, size and shape. The benefit of using ResNet layers is that it is able to convert the feature vectors from domain A into the ones from domain B by combining the dissimilar features of the input image. In our case, 6 ResNet blocks were applied to the generator. Each ResNet block creates two Convolution-InstanceNorm layers with 3×3 filters and 1×1 stride and without a ReLU activation for the second layer.

The generators are trained via their associated discriminators to minimize the loss predicted by the discriminator for generated images marked as “real”, which is the adversarial loss. As such, they are encouraged to generate images that better fit into the target domain. The generators are also updated based on how effective they are at the regeneration of a source image, which is the cycle loss. Finally, a generator is expected to output an image without translation when it is provided an image from the target domain, which is identity loss. Putting all these losses together, each generator model is optimized via the combination of four loss functions:

- Adversarial loss (L2 or mean squared error).
- Identity loss (L1 or mean absolute error).
- Forward cycle loss (L1 or mean absolute error).
- Backward cycle loss (L1 or mean absolute error).

1.3.4. The difference between CycleGAN discriminator and pix2pix discriminator

A commonly-used pattern of deep convolutional discriminator models is Convolutional-BatchNorm-LeakyReLU layers. However, the CycleGAN discriminator uses **Instance Normalization** instead of Batch Normalization. It involves standardizing (i.e., scaling to a standard Gaussian) the values on each output feature map, rather than across features in a batch. In our implementation, the CycleGAN discriminator was optimized using least squares loss (L2) as mean squared error, and a loss weight (0.5) was still applied to slow down the training process of the discriminator (Brownlee, 2019).

On the other hand, the discriminators are updated based on real and generated images, while a **pool of generated images is maintained** during the training process for managing how quickly the discriminators learn. The original paper defines a pool of 50 generated images for each discriminator that is first populated and then probabilistically either adding new images to the pool by replacing existing images (50%) or using a generated image directly without adding it to the pool (50%).

2. Current results

2.1. Comparison of ground truth and pix2pix maps with different epoch numbers



Figure 5. A 10-site comparison among source, ground true, and generated images by pix2pix with different epoch numbers.

The generated images become more plausible along with increasing epoch numbers from 10 to 100 as shown by Figure 1. When epoch number is less than 20, the texture of the landscapes cannot be preserved, and most roads and greens are missing or disconnected, although the locations and colors of the identified features are generally correct. When epoch number is more than 60, their differences are insignificant and the results look “real” compared with ground true, indicating that the epoch number should be at least set as 60, which can be also proved by loss curves (Figure 6).

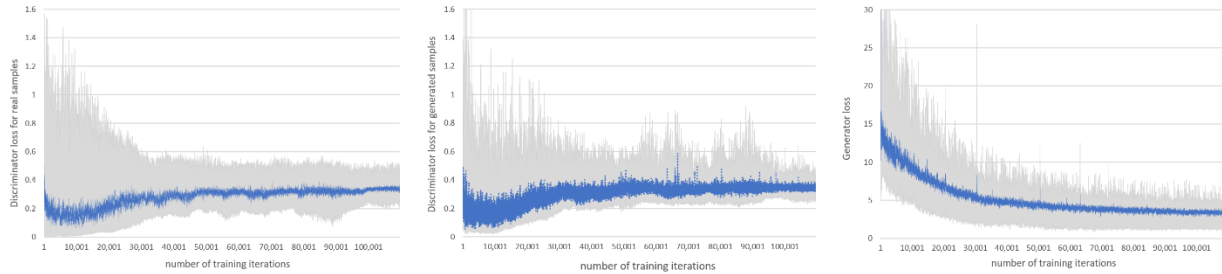


Figure 6. Loss curves over 100 epochs (i.e., 109,600 iterations) of pix2pix. Discriminator loss for real samples, discriminator loss for generated samples, and generator loss are represented from left to right respectively. Blue curves are the moving averages based on a 25-period of the original data (gray). The curves indicate that the image quality improves after 60 epochs when they start to plateau.

To explore the pix2pix performance in detail, we plotted 20 different sites from source, ground true, and generated images by pix2pix with 100 epochs (Figure 7). Generally speaking, the results from the pix2pix model are satisfying for the translation task of aerial images->Google Maps. The generated maps well preserve the locations, colors, textures, and shapes of different landscape features, including main roads (white lines) and highways (oranges curves), greens (e.g., parks, forests, gardens), residential neighborhoods and communities, building blocks, commercial places, water areas, etc.

However, some obvious “flaws” still can be detected based on our observation:

1. Some shortcuts in greens (see (c) and (t)), railways (e.g., a railway station in (b), a railway lane in (j)), and non-directional ramps/ring roads (see (g)) failed to be translated and went missing;
2. The connectedness of different landscapes was strengthened too much in our pix2pix model, resulting in the loss of detailed, small-sized features surrounded by a larger homogeneous region. For example, the generated image in (g) failed to delineate the small group of buildings among the greens in the bottom-left corner; the generated image in (d) did not produce the small lake/pond in the bottom-left corner;
3. The boundaries of the landscape features are blurred and coarse compared with the ground truth, especially for the buildings.

There is one possible reason behind these “flaws” – the frequencies of some specific features inferred from the training dataset are insufficient, therefore, the pix2pix model could not gather enough information to identify those features. For example, non-directional ramps/ring roads barely show up in the training dataset, so it is beyond the prediction capability of the pix2pix model to accurately delineate them in the output image.

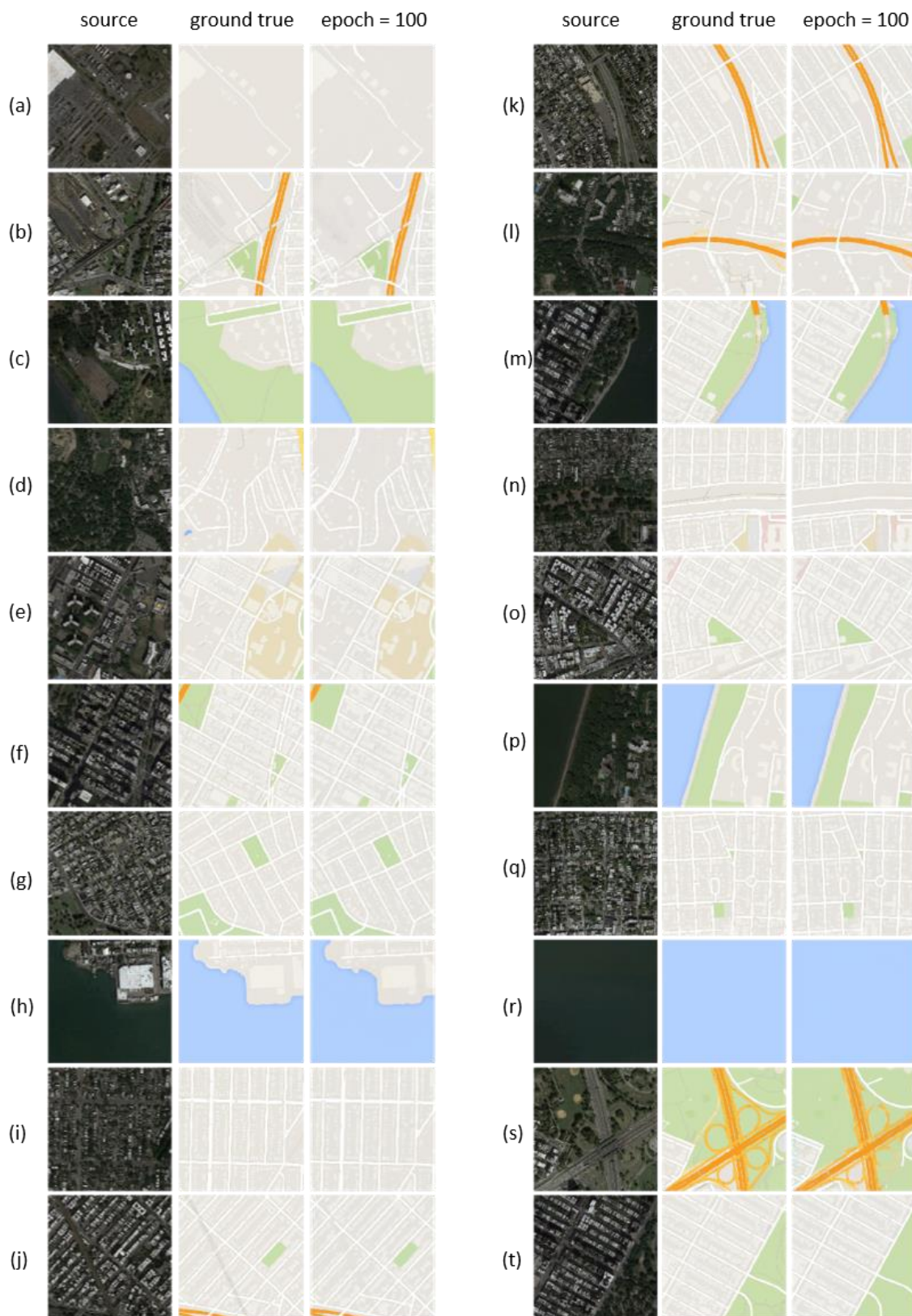


Figure 7. A 20-site comparison among source, ground true, and generated images by pix2pix with 100 epochs.

2.2. Two-way results of CycleGAN with different iterations













































































































































iteration times	Aerial photo->Google Map					Google Map->Aerial photo				
100										
										
200										
										
300										
										
400										
										
500										
										
600										
										
700										
										

Figure 8. Two-way results of CycleGAN models with 100-700 iterations.

Due to more demanding computations of the CycleGAN model and the lack of GPU hardware, we could only complete a few hundred iterations for CycleGAN. Therefore, we were not able to obtain satisfying transformations based on CycleGAN at this point. However, an interesting phenomenon is found that the backward translation Google Maps->aerial images evidently performs better than the forward one, and even based on limited iterations, the locations, shapes, textures of most landscape features are desirably generated. We could continue to dig into this difference more deeply during the training process in future.

2.3. Project webpage ([link](#))

At this stage, we've built our project webpage based on github.io. We drafted the main sections of this project and added them to the webpage, including motivation, literature review, methodology, results, encountered problems and discussion, etc. Additionally, source code and the links to the public datasets were also provided on the webpage.

3. Difficulties during the implementation

3.1. Evaluating the results: qualitative vs. quantitative?

To more holistically evaluate the visual quality of the results, both pix2pix model and CycleGAN model (Isola *et al.*, 2017; Zhu *et al.*, 2017) was based on Amazon Mechanical Turk (AMT) workers to assess perceptual realism. The percentage of Turkers labeled *real* will be utilized as a metric – a higher percentage indicates a better performance in image translation. This is because plausibility to a human observer is usually the ultimate goal for image generation. However, due to the time limit, we did not perform AMT as an evaluation yet. Instead, the current results from both pix2pix model and CycleGAN model were mainly evaluated by the project teammates as a qualitative process. We examined in detail the following aspects to evaluate the translation quality of the generated images:

- Locations;
- Colorization;
- Textures;
- Shape preservation;
- Connectedness.

3.2. Model training run on CPU hardware

We run the examination based on CPU hardware, although GPU hardware would be highly recommended. As a result, the speed was pretty slow due to the large training dataset and the high computational demands of the model. For example, the processor run for the **pix2pix** model is Intel® Core™ i7-10750H Processor, and it took 30 hours to finish running the model when the epoch number was 100. For the **CycleGAN** model, we used AMD Ryzen™ 7 2700X processor and each epoch took approximately 12 hours. Therefore, only a few epochs could be completed and applied to train **CycleGAN** model.

4. Major changes compared with our proposal

Given the limited time for the rest of the semester, it will be more feasible to complete a **comparative analysis** of the existing studies about map synthesis with cartographic design based on aerial images. Currently we are on the right track with the progress. We could continue to devise a novel, better GAN-related model for this application in the next step when time permits.

Accordingly, based on the full observance of its historical development, this project can potentially contribute to building a general framework in this field, providing a comprehensive comparison of the feasibility of different models, and discovering the unrevealed considerations of factors for further efforts in future.

5. Further efforts

5.1. Adding other GAN-related models in the comparative analysis

- Mode seeking generative adversarial network (MSGAN)
- BicycleGAN
- StarGAN
- MapGAN (Li *et al.*, 2020)

5.2. Comparing the advantages and disadvantages among different models

Table 1. An example of expected results

Model	Advantages	Disadvantages
pix2pix	1. Pix2pix is easy to implement and requires less computations. 2. U-Net applies skip-connection so that the down-sampled layers share more information with the up-sampled ones. 3. PatchGAN discriminator improves the quality of generated images and reduces the amount of computations. 4. L1 loss function is added to ensure the consistency of input and output.	1. Paired-wise training dataset is required. 2. The diversity of the generated images is constrained on the availability of the training dataset, and therefore the application scenarios of pix2pix might be highly-limited due to the one-to-one mapping relationship between input and output.
CycleGAN	1. Paired-wise training dataset is not required, making this model more practical for image translation.	1. CycleGAN fails to perform geometric transformation satisfactorily. 2. It lacks diversity for output. One specific feature might only have one style in all output images - the averaged of the training dataset. 3. It is not useful when the generated image overfits to the real image;
...

References

Brownlee, J. (2019) *Generative Adversarial Networks with Python, Deep Learning Generative Models for Image Synthesis and Image Translation, Machine Learning Mastery*. Machine Learning Mastery.

Available at:

https://books.google.com.sa/books/about/Generative_Adversarial_Networks_with_Pyt.html?id=YBimDwAAQBAJ&redir_esc=y%0Ahttps://machinelearningmastery.com/generative_adversarial_networks/.

Isola, P. *et al.* (2017) 'Image-to-image translation with conditional adversarial networks', *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua, pp. 5967–5976. doi: 10.1109/CVPR.2017.632.

Li, J. *et al.* (2020) 'MAPGAN: An intelligent generation model for network tile maps', *Sensors (Switzerland)*, 20(11). doi: 10.3390/s20113119.

Zhu, J. Y. *et al.* (2017) 'Unpaired image-to-image translation using cycle-consistent adversarial networks', *arXiv*, pp. 2223–2232.