Map style transfer using pixel-to-pixel model

Weisheng Jin1,†, Shihui Zhou2,† and Luhao Zheng3,† *

1College of Engineering, Auburn University Alabama, City, 36832, United States, wzj0028@auburn.edu
2Faculty of Science, Wilfrid Laurier University, Ontario, 430000, Canada, zhou3200@mylaurier.ca
3Huazhong University of Science and Technology, Wuhan, Hubei, 430000, China, u201816088@hust.edu.cn
†These authors contributed equally.

Abstract. The image style transfer is a well-known task and it is applied commonly in the field of artificial intelligence, which converts an image to an image with a specific style without modifying the image content and can be applied to many fields such as changing styled automatically for photos in the cellphone and map wrapping. Recent works focus on a general image style transfer. In this paper, we aim to solve the problem in a navigation application, which is to transfer the satellite maps into map images automatically by deep learning techniques. The methodology we choose for this project is a special GAN named pix2pix model. We decide to use the satellite images as our inputs for the model and let the corresponding Google map images be the output sources. Similar to a traditional GAN, we train the discriminator and generator simultaneously and we attempt to run the training process for 50 epochs in total. PNSR and SSIM are two features we are going to use to test the performance of our results. Finally, the accuracy needs to be improved.

1. Introduction
GAN (Generative Adversarial Networks) [1] was first proposed by Ian J. Goodfellow and his team in 2014. The main idea of this model is to get the balance between generative model G and discriminative model D. Through the adversarial process, both models are well trained. The Generative model can generate data which is real enough to cheat a discriminative model and discriminative model’s output will finally equal to 1/2. Over time GAN has been exploited for many fields and derived a lot of different models, such as CGAN [2] (Conditional GANs) and DCGAN [3]. In this paper we attempt to explore an application using the Pix2pix [4] model, along with a special model called PatchGAN [5]. It is a kind of special discriminator of GAN. Different from an ordinary discriminator, PatchGAN will output matrix which means that it can divide a picture into small pieces to process separately. And because of this feature, PatchGAN can make the model have better performance on image detail processing.

In this paper, we aim to transfer our satellite images into corresponding Google map images in high-quality by using a Pix2Pix model, and we test the performance by SRGAN features PSNR and SSIM [6].
2. Method

2.1. Pix2pix
The traditional method applied for images is using convolutional neural networks (CNN) [7] to construct a map between source images and target images. However, this might result in images with relatively low resolutions. In order to improve the resolution, conditional Generative Adversarial Network (CGAN) [2] is recommended. As shown in Figure 1, the result from L1 loss is remarkably vague compared to Ground Truth and CGAN.

![Input, Ground truth, L1, cGAN, L1 + cGAN comparison](image)

Figure 1. Image-to-image results comparison with diverse qualities

Pix2pix model is a special type of Generative Adversarial Network (GAN), which indicates that it consists of a generator model and a discriminator model. Unlike other GANs, pix2pix belongs to a special type of GAN named CGAN, which is commonly used for image-to-image transferring purposes by training a deep convolutional neural network. The architectures of pix2pix’s generator and discriminator are quite different from that of other traditional GANs. For the generator, a U-Net [8] architecture which represents an encode-decode model is applied. Both the encoder and the decoder are comprised of numerous standardized blocks involving convolutional, batch normalization, dropout, and activation layers. Pairs of mirrored layers, such as the first layer of the encoder and the last layer of decoder, are linked by skip-connections shown as dash arrays in Figure 2 in order to acquire results with much higher quality, as presented in Figure 1 previously [4].

![U-Net architecture for the generator](image)

Figure 2. The U-Net architecture for the generator [8]
On the other hand, the discriminator of the pix2pix model has a far more complex structure compared to the generator. A convolutional PatchGAN [5], also known as a Markovian discriminator, is used in our discriminator for detecting the generated pictures from the generator model. A PatchGAN can map the output prediction into a square with a specific size or patch of an input image, and it is definitely beneficial for cases when using the same model on input images with diverse patch sizes [4]. The output prediction of the model might be one value or a square activation map of values, and each value stands for the if a patch in the input image is real.

2.1.1. Training

In this section, more detailed information on training our model is provided. When it comes to the discriminator model, we send a sample whose size is 70*70 to the PatchGAN to generate the fake image, and then the network is optimized using both real and generated images. The model is trained on 256-by-256 resolution [4]. In the beginning, the model concatenates two input images and the combined result are delivered to a series of convolutional layers with diverse filters (a 64, a 128, a 256, two 512s), a 4-by-4 kernel, 2-by-2 strides, a kernel initialization set to be initialized, and the same padding. A leaky ReLU activation with a learning rate of 0.2 is utilized by all of these layers, and the last three layers also are followed by a batch normalization to accelerate training. Moreover, there is a patch output linked to the above layers. A sigmoid activation is used due to the fact that there is only one output neuron. The optimization applied is binary cross entropy, and a weighting is also used to update the model by half of the usual effect.

Turning to the generator model, the situation is more complicated. The generator is trained by adversarial loss, which develops the generator to generate plausible images in the specific domain. And in the situation of generated image and the expected output image can update a generator through L1 loss. Weight initialization should be defined initially in both encoder and decoder blocks. However, there is a downsampling layer, a leaky ReLU and a conditional batch normalization in an encoder block, in contrast to the upsampling layer, a ReLU activation and an ordinary batch normalization for the decoder block. In addition, there is a conditional dropout to avoid overfitting issue in the decoder block, followed by a merged skip-connection as well. For the whole generator, a number of encoder blocks are required for the encoder model. Similarly, some decoder blocks are also needed for the decoder model as well. A bottleneck with no batch norm is used to link the encoder model and the decoder model. Lastly, an output deconvolutional layer with 3 outputs and a tanh activation is added.

Training can be processed when the two models are prepared. The generator model is trained according to the discriminator model and is updated to minimize the loss, which is our final goal represented by equation (1) [4].

As revealed in equation (1), G stands for the generator and aims to minimize our objective against an adversarial D which aims to maximize it. The reason for choosing L1 distance instead of L2 is that L1 returns a clearer result [4].

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

(1)

2.1.2. Test

In the test process, we utilize the validation set in the map dataset¹ and for further analysis, we also obtain some screenshots of satellite images from Google Earth to validate the effectiveness of our method. In our paper, we attempted to use TensorFlow and Keras as our framework for coding and we run our code through GPU in order to accelerate the process. We run the model for 50 epochs. The images are going to be generated and the model can be saved after the training process is done.

3. Results and Discussion

3.1. Dataset Description

The map dataset¹ constitutes 2196 figures of satellite images, taken from New York, and their matching
Google maps images. All of these can be acquired from the pix2pix website and downloaded as a zip file with 255 megabytes. After unzipping it, a directory named “maps” with a training folder “train” and a validation folder “val” can be gained. There are 1097 images contained in “train” and 1099 in “val”, each of which is in JPEG format and has a length of 1200 pixels and 600 pixels in height. Figure 3 provides a corresponding Google map result and its relative satellite photo from “train”.

1. The dataset is available at [http://efrogsans.eecs.berkeley.edu/pix2pix/datasets/maps.tar.gz](http://efrogsans.eecs.berkeley.edu/pix2pix/datasets/maps.tar.gz)

![Google map image and Satellite image](image)

Figure 3. A sample for a pair of Google map image and its satellite version

In our paper, we train the pix2pix model by selecting the satellite images as our source images and Google map images as our target images for 50 epochs. For every 10 epochs, the model and produced images in png form are saved.

3.2. Evaluation Standards

In our project, we decide to use peak signal-to-noise ratio (PSNR) [9] and structural similarity (SSIM) [10] to evaluate the quality of our generated Google map images. Both of these two standards are commonly when dealing with images or videos by super-resolution generative adversarial network (SRGAN) [6]. To calculate PSNR and SSIM, one way is to convert our images into an array whose data type is unit8, so that the maximum pixel reaches 255 px. Both of which need to be in the same shape and takes the generated images and the target images in grayscale version instead of RGB version to calculate. For our project, we calculate the PSNR and SSIM for each pair of input images and target images and display the average as our final values of PSNR and SSIM.

3.3. Results

The result we obtained is displayed in Table 1.

| Method                  | PSNR | SSIM |
|-------------------------|------|------|
| Pix2pix – Map Transfer   | 3.934| 0.060|

Table 1 reveals that the value for PSNR and SSIM are relatively low, indicates that the result for 50 epochs we got is not quite of high quality. The PSNR value shows that the evaluation outcome is different from the corresponding pixels, while the SSIM value is normally ranged from 0 to 1. Our result shows that the structural similarity between the generated map image and the real map image is not obvious because the SSIM value we obtained is a bit large.

Next, we move on to the visualization of our generalized results after we have run 50 epochs in total. The result is displayed in Figure 4.
From the training, we have realized that generated Google Map images are more realistic as the number of epochs increases. With regard to the limitation of our current equipment and network, it could be argued that more epochs (e.g., 100 epochs) are needed if we expect more plausible results though more time-consuming. Furthermore, expanding the size of the training set such as image augmentation could also gain a higher accuracy.

4. Conclusion
In this paper, we propose a pix2pix method of transferring the satellite maps to map images. After constructing the appropriate models for a generator and a discriminator, we attempt to achieve outcomes with high resolution. After deciding the input image (satellite photos) and the output image (Google map image), we have gained results with 3.934 for PSNR value and 0.060 for SSIM value. Both outcomes illustrate that the high accuracy is not achieved due to the fact that the number of epochs is low, however the distortion for the image is good. Compared to the traditional GAN, pix2pix could be more suitable for tasks such as map style conversions with high resolution. Consequently, our method can effectively handle this task with satisfying performance and provide the automatic map generation method for navigation applications.

Acknowledgements
First of all, we would like to express our special appreciate to our instructor Dr. Pengyu Zhang for his continuous assistance and support. Moreover, we would also thank all the members from the PBL program (Machine Learning II) hold by Dr. Mark Volgersberger from 2020 to the beginning of 2021. This program imparts academic knowledge about deep learning and provides us a number of opportunities to practice our programming skills and learn from other group members. Our thesis benefits a lot from it.

References
[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. Generative Adversarial Nets., University of Montreal, Montreal, Jun 10 2014.
[2] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).
[3] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).
[4] P. Isola, J. Zhu, T Zhou, and A. A. Efros, Image-to-Image Translation with Conditional Adversarial Networks., Cornell University, New York, 2018.
[5] Ugur Demir, Gozde Unal. Patch-Based Image Inpainting with Generative Adversarial Networks. 20 Mar 2018.
[6] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, ‘A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. S. Twitter. Photo-Realistic Single Image Super-Resolution Using a
Generative Adversarial Network., May 25 2017.

[7] I. J. Goodfellow, Y. Bengio, A. Courville., Deep learning (Vol. 1). Cambridge: MIT press, 2016: 326-366.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In MICCAI, 2015.

[9] Q. Huynh-Thu and M. Ghanbari. Scope of validity of PSNR in image/video quality assessment. The Institution of Engineering and Technology, 9 June 2008.

[10] A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, pp. 2366-2369, doi: 10.1109/ICPR.2010.579.