Building Eraser and Surrounding Area Generator (BESAGen)

He Yang (20844672, yhecb@connect.ust.hk)
Kang Zhaowei (20925347,zkang@connect.ust.hk)

1. Introduction

In the project, we propose a powerful tool that generates an image with buildings erased and generates the surrounding environment from the input satellite image. In buildings erasing, given a satellite image as input, our goal is to output an image that erases the building in the input and inpaint the building pixels with generated environment pixels. In surrounding area generation, using a satellite image as input, our goal is to generate the surrounding environment by referring to the enviroments in the input images.

Fig. 1. An example of building eraser outcome

Fig. 2. An example of surrounding generator outcome

We achieve these goals by building a pipeline with two parts. The first part is the satellite imagery segmentation using UNet introduced in the original proposal. To improve our

project's novelty, we further introduce an image inpainting model, deep fill v2. These two parts together perform as the Building Eraser and Surrounding Area Generator (BESAGen).

2. Dataset and Preprocessing

Our 1Gb dataset contains totally 5936 pieces of 256*256 RGB satellite image in png files as input and their corresponding segmentation ground truth as target. They cover a large area of Staten Island, New York. Most of them are capturing urban area where there are different kinds of buildings, cars, roads, trees and grassland.

Fig. 3. A sample from the dataset

Figure 3 shows a sample from the dataset. The left one is a satellite image and the right one is the segmentation ground truth where buildings are highlighted in white.
When loading the data, we first convert each image into a (256, 256, 3) array. Then we apply normalization to them by dividing every entry in the arrays by 255.

3. Methodology

3.1. Machine Learning Task

We use machine learning approaches in the project and transfer the satellite imagery segmentation and image inpainting into two machine learning tasks. The first task is a binary semantic segmentation which is a supervised learning task. The input is a satellite image and the output is the segmentation prediction. The second task is generation which is also a supervised learning task. The input is the segmentation prediction from the previous task and the output is the final image with buildings erased or the surrounding environment generated.

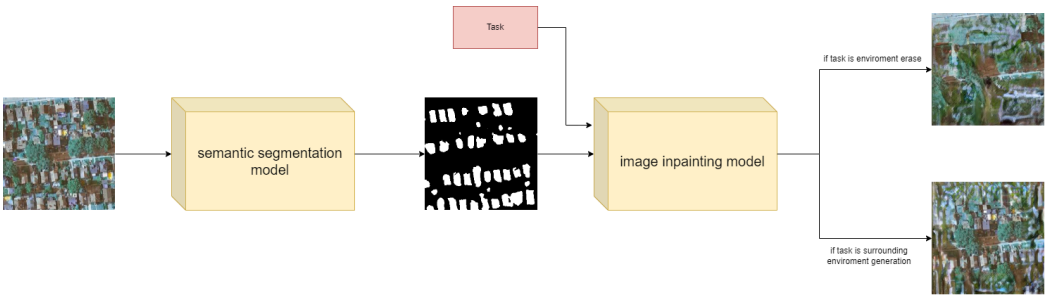


Fig. 4. Tasks and procedure

3.2. Computing Environment

The whole program runs on Google Colab platform using Nvidia GPU for computing.

3.3. Data Spiltting

We use 90% of the total samples to train and validate the UNet model, and the remaining ones are left for generation. During the training process, the ratio of data for training and validation is 4:1.

3.4. Model

We use UNet as our semantic segmentation model and deepfill v2 as inpainting model.

3.4.1 semantic segmentation model

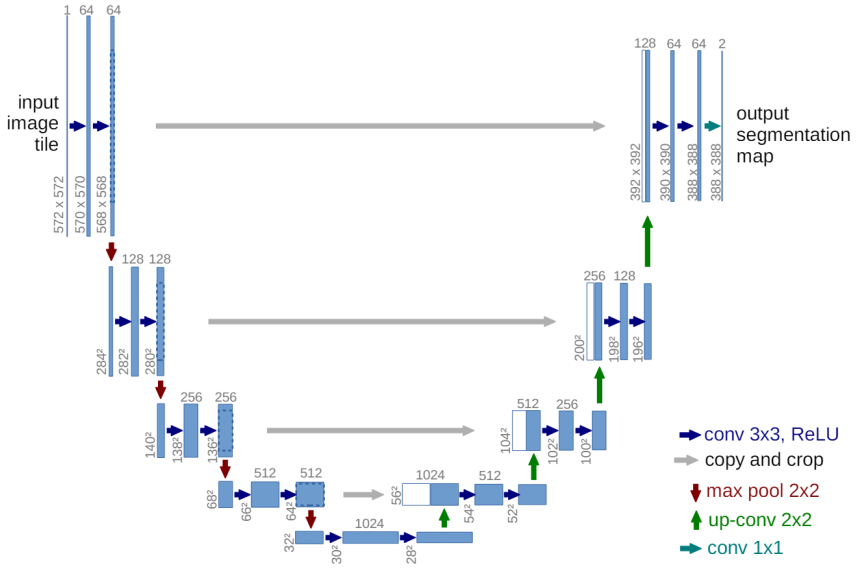


Fig. 5 UNet structure in [1]

We choose UNet [1] as our semantic segmentation model. Figure 5 shows a typical UNet structure introduced in [1], which is slightly different from the model we use. UNet was originally proposed for biomedical image segmentation. Previous study [2] showed that it also performs well in satellite imagery segmentation UNet has a convolutional encoder performing downsampling and decoder performing upsampling. There exist some direct

connections between encoder and decoder. In our model, the filter numbers are 64, 128, 256 and 512. There are two convolutional layers for each filter number in encoder and one in decoder.

3.4.2 image inpainting model

We choose the Free-Form Image Inpainting with Gated Convolution(deepfill v2 [3]) as the image inpaining model. The key reason for using this model is that it performs well when dealing with different shapes of mask in one dataset. Note that the deep fill v2 we used is from the github repositories(https://github.com/nipponjo/deepfillv2-pytorch)

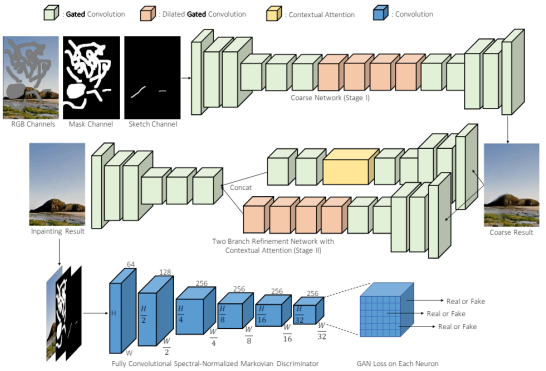


Figure 3: Overview of our framework with gated convolution and SN-PatchGAN for free-form image inpainting.

Fig. 6. deepfill v2 structure in [3]

The deep fill v2 consists of three networks. The first network(on the top of the above image) is an encoder-decoder architecture autoencoder that generates the coarse inpainting image from the input masked image. The second network is a two-branch refinement network with contextual attention. This network generates the final output image from the coarse inpainted image. The first and second network of deep fill v2 can be seen as a generator of GAN, and, the third network works as a discriminator of the first and second network. Specially, the output of this discriminator is a 3D tensor instead of a scaler. This special design improves the discrimination ability of the discriminator on spatial features of the generator output.
In practice, to improve the output image quality, we do some basic transformation to the input mask of the deep fill v2 depending on the task.
For the building erase task, we use a convolutional layer with a fixed weight kernel to expand the area of the mask output by the semantic segmentation model(In our practice, Unet mentioned above). By doing this we can increase the probability of the mask input to the deep fill v2 fully covering the building pixels in the image.
For the surrounding area generation task, we first reshape the image and mask to a smaller size while maintaining their original image size. And we mask the new come pixels after squeezing as mask to make sure the deep fill v2 can generate surrounding environments. To help generate more environment content, we use a BFS to detect the building pixels that are on the edge of the original image and also annotate them as mask

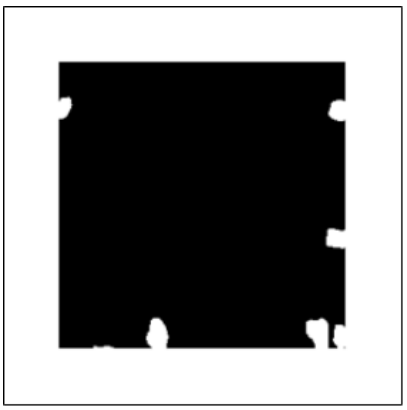


Fig. 7. A mask sample of the surrounding environment generation task

With the mask and original image, the deep fill v2 can generate the surrounding area.

3.5. Parameter Setting

For the UNet model,we set batch size to 10. We choose Adam as the optimizer with a learning rate of 0.0001 and weight decay being 0.00005. We use GELU as activation function in hidden layers, Sigmoid as activation function in the output layer and binary crossentropy as loss function. During training, we set the number of epochs to 100 and early patience to 5 to prevent overfitting. We use the jaccard coefficient to monitor the model performance during training, which measures the intersection between prediction and target. A winning solution of SpaceNet2 competition [3] showed that the jaccard coefficient performs well as a monitor function for satellite imagery segmentation tasks.

For the deep fill v2 model, we set batch size to 16. We choose Adam as the optimizer with a learning rate of 0.0001. The optimizer hyperparameter is the same for generator and disciminater. Beta1 is 0.5 and Beta2 is 0.999.

4. Result

4.1 Segmentation

epoch	accuracy	jaccard_coef	loss	val_accuracy	val_jaccard_coef	val_loss
0	0.857158422	9.94E-13	0.391575247	0.820343852	9.99E-13	0.403643727
1	0.869486153	8.55E-13	0.314017862	0.824715018	9.61E-13	0.443018407
2	0.898127079	6.32E-13	0.254931986	0.849401176	7.97E-13	0.366963744
3	0.911215425	5.49E-13	0.226751909	0.862653732	6.65E-13	0.343612671
4	0.918293774	5.04E-13	0.207954854	0.868077278	6.12E-13	0.314774841
5	0.924108028	4.69E-13	0.195082948	0.875171721	5.50E-13	0.302265972
6	0.926745892	4.51E-13	0.1877013	0.878763855	5.64E-13	0.300256073
7	0.930286527	4.32E-13	0.179980382	0.883434772	4.99E-13	0.285362661
8	0.933327436	4.14E-13	0.171341926	0.887413859	5.36E-13	0.280215085
9	0.935423732	4.02E-13	0.166277185	0.89039892	5.10E-13	0.275601625
10	0.937520564	3.90E-13	0.161357939	0.888438821	5.53E-13	0.280233383
11	0.939693749	3.78E-13	0.155732974	0.894029856	4.89E-13	0.262949646
12	0.940880299	3.70E-13	0.15283604	0.894443631	5.10E-13	0.2597875
13	0.943995118	3.53E-13	0.144960031	0.899583578	4.69E-13	0.256171763
14	0.944513977	3.51E-13	0.143748537	0.900658965	4.64E-13	0.251964748
15	0.946914673	3.36E-13	0.137186915	0.901594341	4.70E-13	0.251491874
16	0.948350966	3.28E-13	0.133099899	0.900095999	4.56E-13	0.247424543
17	0.950012982	3.18E-13	0.129000768	0.900951564	4.32E-13	0.245169014
18	0.951437771	3.10E-13	0.125551581	0.901349008	4.80E-13	0.244516417
19	0.952085257	3.07E-13	0.123950027	0.905275345	4.34E-13	0.237214014
20	0.953579247	2.98E-13	0.119539112	0.905843318	4.48E-13	0.239133313
21	0.95626682	2.83E-13	0.113153994	0.906829655	4.52E-13	0.235235244
22	0.95720464	2.76E-13	0.110493079	0.907092631	4.28E-13	0.23698695
23	0.959488869	2.63E-13	0.104739048	0.90927285	4.41E-13	0.250345975
24	0.961293697	2.53E-13	0.099732675	0.905704856	4.29E-13	0.261480272
25	0.962168515	2.48E-13	0.097348504	0.909178078	4.34E-13	0.256879002
26	0.963034809	2.42E-13	0.0954087	0.90730685	4.34E-13	0.254486501
27	0.965413094	2.28E-13	0.089114137	0.908860803	4.25E-13	0.273256361
28	0.96753031	2.16E-13	0.083316423	0.90750277	4.26E-13	0.271740466
29	0.969260216	2.04E-13	0.078792624	0.907985806	4.21E-13	0.276435524
30	0.969278991	2.04E-13	0.078434981	0.905064166	4.30E-13	0.293398976
31	0.970738709	1.96E-13	0.074508056	0.907897353	4.34E-13	0.29634437
32	0.972484469	1.85E-13	0.069770478	0.90813458	4.38E-13	0.327378541
33	0.974005818	1.76E-13	0.065849319	0.903383136	4.35E-13	0.334191084
34	0.974025488	1.75E-13	0.065454513	0.907143056	4.53E-13	0.337692291

Fig. 8. UNet training history

We use satellite images as input and the corresponding segmentation ground truth as target from the dataset to train the UNet model. As shown in Figure 8, the validation jaccard coefficient has not decreased for 5 epochs since the 30th epoch. The training process thus stopped at the 35th epoch due to early stopping and the model in the 30th epoch is saved as the best model whose validation accuracy reaches 0.908.

Fig. 9. An example of segmentation prediction by UNet

Figure 9 shows an example of segmentation prediction made by the UNet model. The model successfully detects most of the buildings in the input and precisely extract them from the background. Overall, the UNet model has good performance on the segmentation task.

4.2 Inpainting

As our dataset does not have the corresponding target image for our two generation tasks. We have to train our deep fill v2 on other dataset. We train our deep fill v2 on the place2 dataset[4]. And as this is a generation task. We will evaluate the model based on some output samples.

4.2.1. Building erasing task

Before After

Fig. 10. A building erasing sample

From Figure 10, we can find that the building on the left bottom is erased and is replaced by a generated forest. This example shows that our model can fetch content from the unmasked area and use the content information to replace the building.

4.2.2. Surrounding environment generation task

Before After

Fig. 11. A surrounding environment generation sample

Figure 11 shows one of the surrounding environment generation samples. From the sample we can see that Our model fetches the color and content from the input image and generates similar content such as forests(below) and roads(left bottom).
In the end of the report we will post more samples of the building erasing task and the surrounding environment generation task.

5. Conclusion and Limitation

5.1. Model Advantage

From the UNet training history and the task output sample, we can find that the Unet can annotate most of the building pixels in the image. And our deep fill v2 can basically inpaint the masked area with content fetched from the unmask area. We can conclude that our model can finish our task.

5.2.Model Disavantage

The details of the image are not refined enough. And the accuracy of the semantic segmentation model can be improved.

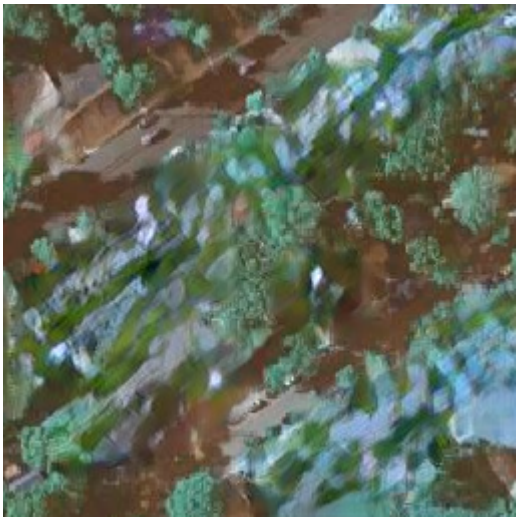
5.3. Possible Improve Approach

By using more complex semantic segmentation models (such as TransUnet [5]), the accuracy of building pixel annotation can be improved. And if a dataset specific for this task can be found, the performance of the image inpainting model is likely to improve.

More samples

Building erasing samples

Before



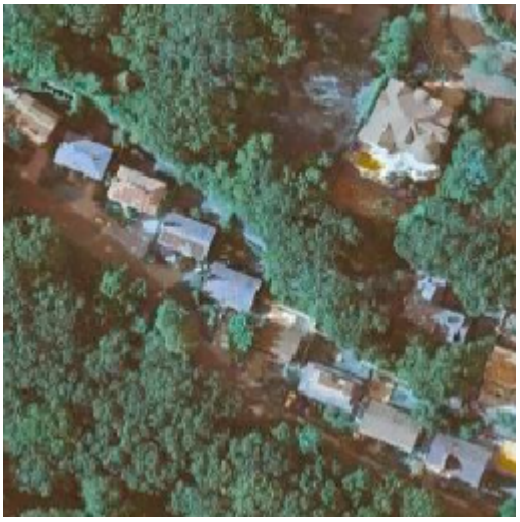
After

Before

After


Before After

Surrounding environment generation samples




Before After

Before After

Reference

[1] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.

[2] V. Khryashchev, L. Ivanovsky, V. Pavlov, A. Ostrovskaya and A. Rubtsov, "Comparison of Different Convolutional Neural Network Architectures for Satellite Image Segmentation," 2018 23rd Conference of Open Innovations Association (FRUCT), Bologna, Italy, 2018, pp. 172-179, doi: 10.23919/FRUCT.2018.8588071.

[3] https://github.com/SpaceNetChallenge/BuildingDetectors_Round2/tree/master/1-XD_XD

[4] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., & Huang, T. S. (2018). Generative image inpainting with contextual attention. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5505-5514).

[5]:Chen, J., Lu, Y., Yu, Q., Luo, X., Adeli, E., Wang, Y., ... & Zhou, Y. (2021). Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*.