School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Deep Generative Streets

Jialin Zhu

Submitted in accordance with the requirements for the degree of
MSc Advanced Computer Science (Intelligent System)

2018/2019

The candidate confirms that the following have been submitted*:*

<As an example>

Items	**Format**	**Recipient(s) and Date**
Deliverable 1	*Report*	*SSO (September/03/2019)*
Deliverable 2	*Software URL*	*Supervisor, assessor (September/03/2019)*
Deliverable 3	*Representative samples of results*	*Supervisor, assessor (September/03/2019)*

Type of Project: Empirical Investigation

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) _____

Summary

The development of Deep Learning has brought many help to other areas of the computer science. As for computer graphics, image-to-image translation systems have always relied on extremely complex algorithms and they have significant limitations. Deep learning techniques improve the development convenience of such systems. And as long as different training data is used in suitable neural networks, different image-to-image systems can be made.

Image-to-image translation systems, which are based on neural network that can convert map images into real streets image, are very promising. For example, city planners can preview the real situation from sketch design which is very helpful for their work. This project is to do some investigation and experimentations based on this. The goal is to create high-resolution real images with limited hardware resources. The main idea is to generate seamless low-resolution images through neural networks that can be directly used for combining them into a high-resolution one. Furthermore, this project implement Deep Learning techniques as the foundation rather than using some traditional methods.

Acknowledgements

First, I would like to express my deepest gratitude to my supervisor, Doctor Tom Kelly. Many ideas were inspired by him and many difficult problems were solved under his suggestion. Without his help, I could not go this far on this project.

Next, I want to thank Doctor John Stell for his advice about finding some ways to evaluate results in the progress meeting.

Finally, I want to thank for resources provided by the University of Leeds which include HPC (High Performance Computer) and student account for downloading necessary data.

Table of Contents

Summary ..... **iii**
Acknowledgements ..... **iv**
Table of Contents ..... **v**
List of Figures ..... **viii**
List of Tables ..... **x**
Chapter 1 Introduction **1**
1.1 Context 1
1.2 Project Description 1
1.3 Aim 1
1.4 Objective 2
1.5 Deliverables 2
1.6 Project planning 3
1.7 Report structure 5
Chapter 2 Background Research **7**
2.1 AI (Artificial Intelligence) 7
2.2 ML (Machine Learning) 8
2.2.1 Task 8
2.2.2 Performance 9
2.2.3 Experience 10
2.2.4 ML development history 10
2.3 DL (Deep Learning) 11
2.4 Deep Neural Network 11
2.4.1 Forward Propagation 12
2.4.2 Loss Function 12
2.4.2 Gradient Descent 12
2.4.3 Back Propagation 13
2.5 CNN 13
2.6 GAN (Generative Adversarial Network) 14
2.6.1 Generator 15
2.7 image-to-image System based on GAN 16
2.7.1 Pix2Pix GAN 16
2.7.2 Bicycle GAN 17

Chapter 3 Preparation 19
3.1 Dataset 19
3.1.1 Data analysis 19
3.1.2 Digimap 19
3.1.3 Ordnance Survey National Grid 20
3.1.4 Products Choice 20
3.2 Program 22
3.2.1 Programming language 22
3.3 Open source mature code 22
3.4 Environment for developing, training and testing 22
Chapter 4 Experiment and Analysis 24
4.1 Data Pre-processing 24
4.2 Basic training with Pix2Pix GAN and Bicycle GAN 26
4.2.1 Official Implementation 26
4.2.2 Basic Training on Pix2Pix GAN 27
4.2.2 Basic Training on Bicycle GAN 29
4.3 Generated Images Analysis 30
4.3.1 Low-resolution Level Analysis 30
4.3.2 Generate Map Without Label 32
4.3.3 High-resolution Level Analysis 34
4.4 Seams Bicycle GAN 36
4.4.1 Basic Seams Bicycle GAN 36
4.4.2 6-channel Seams Bicycle GAN 38
4.4.2 A Trick Successful Case 40
4.5 Mask and New Loss Function 42
4.5.1 Add Mask 42
4.5.1 Add New Loss Function (Mask Loss) 45
4.6 Spatial Varying Z 48
Chapter 5 Evaluation 51
5.1 Basic Generated Images Evaluation 51
5.2 Combined Image Evaluation 52
5.2.1 Pixels Distance Energy 52
5.2.2 Seams Energy 53
Chapter 6 Conclusion 57
6.1 Conclusion 57

6.2 Future Work 57
List of References 59
Appendix A External Materials 61
Appendix B Ethical Issues Addressed 62
Appendix C Code 63
Appendix D Representative Samples 64

List of Figures

Figure 1.1 From map image to real streets image 2

Figure 1.2 Original Plan Gantt Chart 3

Figure 1.3 Real Process Gantt Chart 4

Figure 2.1 Different AI research area 7

Figure 2.2 Examples of underfitting, appropriate fitting and overfitting (Goodfellow et al., 2016, p.111) 10

Figure 2.3 Performance on large amount data (Brownlee, 2016) 11

Figure 2.4 Deep Neural network 12

Figure 2.5 Gradient Descent in three-dimensional data space 13

Figure 2.6 CNN 14

Figure 2.7 Distribution sample of Generator 16

Figure 2.8 Bicycle GAN network (Zhu et al., 2017) 18

Figure 3.1 Image-to-image translation system example (Isola et al., 2017) 19

Figure 3.2 Examples of downloaded data from Digimap. The left is map image and right is satellite image. 21

Figure 3.3 Flaws in data 21

Figure 4.1 An example of final dataset 25

Figure 4.2 The naming relationship between the small images and large images 25

Figure 4.3 Results views in final html file. The input is the map image used to generate output. The ground truth is the real satellite image for comparison. The encoded is the output of Encoder which is used to identify correct z value. Random sample 01 to random sample 05 are the generated streets images. 27

Figure 4.4 Pix2Pix GAN result example 1. The real_A is the input map image. The fake_B is generated streets image. The real_B is real satellite image. .. 28

Figure 4.5 Pix2Pix GAN result example 2. Houses and Roads are shaper than before compared to figure 4.4. 28

Figure 4.6 Bicycle GAN result example 1 29

Figure 4.7 Strange pattern in Bicycle GAN result 31

Figure 4.8 Mis-generation in Bicycle GAN result 31

Figure 4.9 Other flaws in Bicycle GAN result 32

Figure 4.10 Map images exported from GML without line layer (left) and with line layer (right) 33

Figure 4.11 Map images exported from GML (left) and map images downloaded from Digimap (right) 34

Figure 4.12 New map images remove the influence of text 34

Figure 4.13 Examples of combined high-resolution images. Seams can be clearly seen in these two pictures. Image on the left is result not using "sync". The right image is result using "sync". 35

Figure 4.14 Real satellite image 36

Figure 4.15 Creating new seams dataset 37

Figure 4.16 Road distortion in results 38

Figure 4.17 Add extra 3 channel in Seams Bicycle GAN 39

Figure 4.18 Results of 6-channel Seams Bicycle GAN 40

Figure 4.19 New seams will be introduced if replace result directly 40

Figure 4.20 Using different resolution in test phase 41

Figure 4.21 Results of 3072 resolution images with seams being processed by Seams Bicycle GANs 41

Figure 4.22 Masks added into the neural network 43

Figure 4.23 Examples of Seams Bicycle GAN with different masks 44

Figure 4.24 Seamless results replaced back into high-resolution combined image 45

Figure 4.25 mask loss function working process 45

Figure 4.26 Examples of Seams Bicycle GAN with different masks adding new loss 47

Figure 4.27 Seamless results replaced back into high-resolution combined image 48

Figure 4.28 Two ways of Bicycle GAN adding z value (Zhu et al., 2017) 48

Figure 4.29 Spatial varying z value 49

Figure 4.30 ordered numbers of small images 49

Figure 4.31 Results of spatial varying z value 50

Figure 4.32 Combined high-resolution image by spatial varying z value 50

Figure 5.1 More details on the building roof in real satellite image 53

Figure 5.2 Image view during calculating SE value 54

Figure 5.3 Histogram of Seams Energy value 56

List of Tables

Table 1.1 Description for Project tasks 4

Table 5.1 Results of FID 52

Table 5.2 Results of Seams Energy 54

Chapter 1 Introduction

1.1 Context

Deep Learning became more and more popular nowadays and it attracted many researchers' to study it in depth (Goodfellow et al., 2016). It is a branch of Machine Learning and Artificial Intelligence. Machine Learning is a multi-disciplinary subject involving many disciplines, such as probability theory, statistics, approximation theory, convex analysis and algorithm complexity theory (Bishop, 2006). Specializing in how computers simulate or implement human learning behaviours to acquire new knowledge or skills and reorganize existing knowledge structures to continuously improve their performance.

The original intention of Artificial Intelligence was to make machines capable of creating and learning like humans. Some investigators are already trying to give machine the ability to generate 2D images just like human being. There is an interdisciplinary relationship with computer graphics. Some of the techniques have been well tested and can generate results which are really close to the real ones. One of them is called GAN (Generative Adversarial Network) first introduced by Goodfellow et al. (2014). This project mainly focuses on investigation of GAN and try to solve some known issues.

1.2 Project Description

Image-to-image translation systems have become very successful over the past few years, combined with the boom in data availability it has become increasingly plausible to translate between arbitrary image domains.

The title of this project is "Deep Generative Streets". It is mainly dealing with map images and real streets images. In this case, it is possible to build an image to image translation systems based on GAN that can translate map images to real streets images. It was proved to be feasible by Isola et al. (2017) with their special GAN model.

This project is to make further investigations and research on this basis.

1.3 Aim

The main aim of this project is to generate high-resolution real streets images through GAN using map images. This goal can be divided into two parts.

First, use existing technology to implement the image-to-image system which can translate map images to real streets images in a low-resolution level (shown in figure 1.1). This is a

relatively simple task because the framework for coding in Deep Learning is very mature now and there is a lot of open source resource available.

Figure 1.1 From map image to real streets image

Next, it is not possible to generate high-resolution or infinite results at once with limited resources. The core challenge of this project is also here. Investigation and experiments will be done to find a reasonable solution for generating high-resolution images. Trying to combine low-resolution images into a high-resolution image without seams is the main method. It also means that infinite image can be generated in this way.

1.4 Objective

The objectives and milestones to be achieved during the project are as follows:

a) Get suitable data and pre-process data for further traning.

b) A model of GAN should be provided. It should have the ability to translate some low-resolution map images to relatively real streets images. This also includes tuning and optimizing the model to get enough accurate results. This process is to ensure that low-resolution images have good quality which means they are ready for high-resolution image combination.

c) Combine small images together to generate a high-resolution image. In this part, investigeting, researching, experimenting and modifying neural network architecture will be done for solving images combination problems.

d) Demonstrate, evaluate, analysis the final results of GAN model.

1.5 Deliverables

a) The final research report should be provided. This report should include an introduction of the research, research background information, problems of this project, methods

used, what model does the research finally come up with, what architecture used in the final model, and the conclusion of whole project.

b) This project should provide a GAN model that meets the requirements of the project. Also the code (include scripts) for this model should also be provided.

c) Some representative examples should be provided. Representative results should be kept for future comparison during the procedure.

1.6 Project planning

The initial plan is illustrated with the Gantt chart in Fig 1.2.

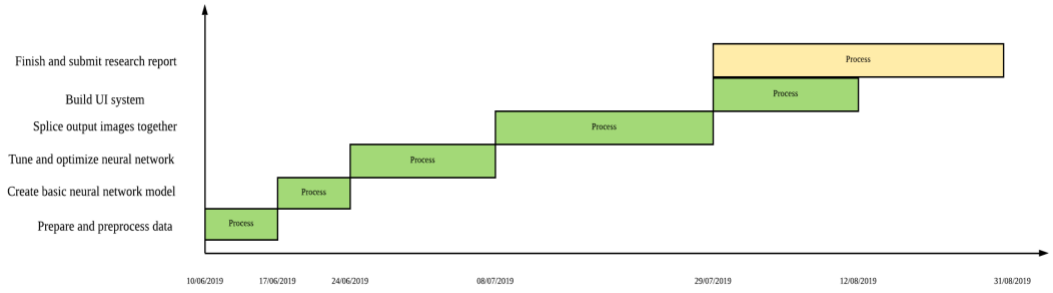

Figure 1.2 Original Plan Gantt Chart

The implementation of the initial plan was affected for the following reasons:

a) Too much noise in the dataset. For example, there is text on the map images which leads to unanticipated results. It takes a lot of time to remove the text in the map.

b) Training time is too long for this project although it is much shorter compared to real industrial projects. It taks almost 13 hours for a 3000 images dataset, 40 hours for 10000 images, and more time for some special methods tried

c) Images combination problem is much harder than expected. This is because in this project only neural networks related techonologies are used rather than introducing some external means like the graphcut texture method (Kwatra et al., 2003) for image synthesis and tileGAN (Frühstück et al., 2019) which adds a special energy function to process the synthesis

d) The questions like what are reasonable evaluation criteria and how to detect the tile seams of the final high-resolution images which are consisted of small images are relatively hard. This means although some results look better with human eyes view, it cannot just say that this is a better result. There must be some criteria. It also took some time to find a reasonable evaluation method.

The biggest impact of these factors on the progress of the project is reflected in the abandonment of a neural network implementing from scratch which is intended to do in the

original plan. Instead of this, some mature open source GAN frameworks were used in the time period of the entire project. A lot of code was added and modified to solve the problem of image synthesis, and a lot of scripts were written for data pre-processing, evaluation and other tasks. Almost 70 neural network models (include many failed cases) has been trained in last 3 months. Also because there was not enough time, the original planned UI System was abandoned. But some representative samples were placed in the appendix.

The final real progress Gantt chart is illustrated in figure 1.3.

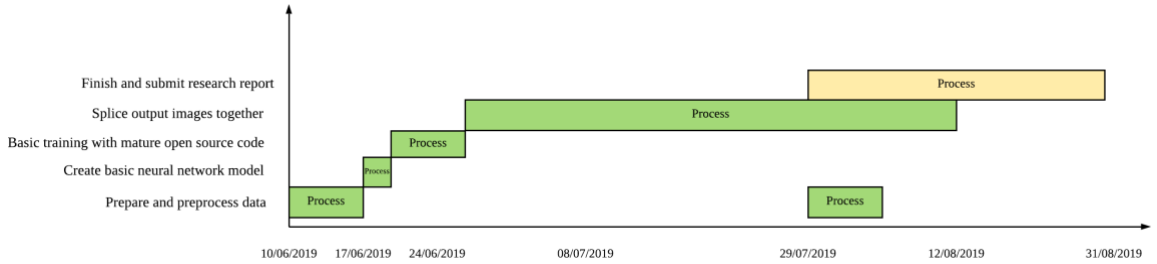

Figure 1.3 Real Process Gantt Chart

The detailed plan information of these two Gantt charts shown above is placed in table 1.1.

Table 1.1 Description for Project tasks

Task Name	**Detail**	**Duration**
Prepare and pre-process data	Data is relatively important in a Deep Learning project. This phase is mainly to find the right data and reasonably process data to prepare for the training.	14 days
Create Basic neural network model	Create a basic neural network to achieve small-resolution map images to real-scene image translation. This task was only tried for 3 days.	3 days
Tune and optimize neural network	This is to adjust the neural network wrote from scratch originally. But because this process takes a lot of time, it was suggested that more mature open source code should be used in order to concentrate on solving the core synthesis problem of the project.	/
Basic training with mature open source code (include tuning and optimizing parameters)	Use a sufficiently mature framework to implement low-resolution map images to real-scene images translation.	7 days

Splice output images together	Find a way to combine the output of the neural network together into high-resolution images. This also includes modifying the structure of the neural network (the structure of the data) and adding various functions (such as adding a new loss function) to create suitable low-resolution images can be used for high-resolution images synthesis.	47 days
Build UI system	In the original plan, a suitable UI system will be built to show results. But because of time constraints, this cannot be done.	/
Finish and submit research report	Finish report for this project.	28 days

1.7 Report structure

The report is carried out according to the following outline:

Chapter 1 gives introduction information about this project, include basic background information and description detail. This chapter also identifies aims, objectives and deliverables. At the end, the initial plan and the actual plan of the project are given.

Chapter 2 discusses all the relevant technologies about project. From the relatively broad Machine Learning, Depp Learning introduction to the detailed Generative Adversarial Network.

Chapter 3 talks about how to build the dataset (where to go for getting desired data and so on), what programming language (include open source code) and which training environment was used of the project.

Chapter 4 is the core chapter of whole report. This chapter mainly talks about what have been done, Include the pre-processing of the data, the underlying results (achieve low-resolution map images to real-scene image translation), what methods have been tried to synthesize output images together to a seamless high-resolution images and analyse whether they have worked or not. Also, a trick successful case is discussed in this chapter.

Chapter 5 discusses the evaluation criteria for neural network results. Some of the improvements in the results can be visually observed, and others may not. This requires a sufficiently reasonable evaluation method to assess the outcome of the project.

Chapter 6 is a conclusion for the whole things had been done in the project. This chapter also discusses the foreseeable vision for future work.

Chapter 2 Background Research

2.1 AI (Artificial Intelligence)

Artificial Intelligence is one of the most popular computing technologies from the day it was proposed. Artificial Intelligence focuses on the theory and practice of self-developing systems that demonstrate the characteristics associates with human intelligence behaviour in Science and Engineering domains (Tecuci, 2012). In short, AI is trying to make computers or any other type of hardware to work as people. They can make right decisions under complex situation just like what human does. AI can be divided into many different fields.

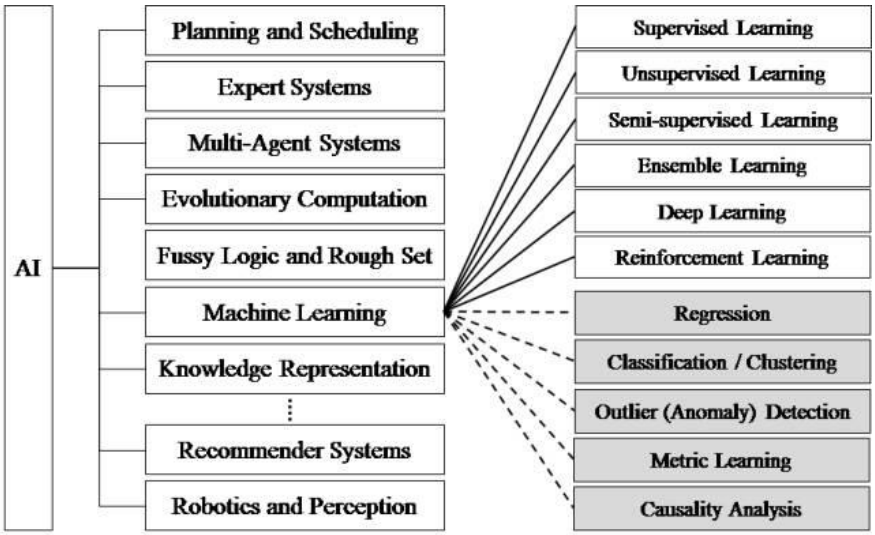

Figure 2.1 Different AI research area

In the early days of Artificial Intelligence development, some tasks which are very difficult for human intelligence but relatively simple for computers (Goodfellow et al., 2016). Designing a program through a series of mathematical rules can solve some complex problems. KRR (Knowledge Representation and Reasoning) is the typical sample. It tried to build an information form system database that can represent real life knowledge and special rules to infer new unknown information. But relatively speaking, a lot of knowledge or rules in real life cannot be defined in a formal way. Using a complex rule have a big chance to slow down the performance of the entire system and may introduce unpredictable bugs. From this point of view, it is very important to let AI have the ability to learn knowledge independently. This ability can be called Machine Learning.

Machine Learning, especially statistical machine learning and connectionism machine learning, has become mainstream since the 1990s (Zhou, 2016). It still has tremendous development potential until today.

2.2 ML (Machine Learning)

The content of Machine Learning is very close to the meaning of the word. It means that the machine can learn by itself. This usually uses calculation approaches which are based on experience to make machine has ability to improve its performance. A more academic definition (Mitchell, 1997) is :

"Definition: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

In the computer field, E (experience) is actually represented as data. The regular patterns of E can be seen as the data which is what machine trying to learn. The learning phase of ML is also called train or training. Therefore, ML can also be seen as a special algorithm for models generation from data. These models can be used to analyse or predict the same kinds of data.

2.2.1 Task

The T (task) in Machine Learning does not mean finishing the model learning procedure, T is to handle similar data by well-learned models (Goodfellow et al., 2016). Specifically, by inputting the same type of data, the corresponding output should be outputted. For example, a trained model with cat data should be able to identify whether the input data is right representation of cats or not.

There are three most common tasks in machine learning:

a) **Classification.** Classification predicts discrete values. In this case, the model needs to judge which category the input data belongs to. If using k different types of cats data for training, then the model should be able to determine which type of cat the new input is. If only using the data of cats and non-cats to train, the result is whether a cat or not. The classification problem actually is trying to find reasonable decision boundaries that can separate different types of data in data space. But in some cases the data may not be linear separable. It is still possible to find a line that can approximately separates different data. In this way, the number of misclassified data is the least.

b) **Regression.** Regression is a task for continuous value prediction. This is to make a numerical based prediction for the input data. The value of the different features of the input data can have a large effect on the value of the output. For example, as the size of the house increases, the price of the house may continue to increase. Or as the distance from the city to the city increases, the price of the house may decrease. Under this circumstance, it is not expected a discrete result whether a house is worth buying but the continuous result of the house price. The regression is to find the best line which

can fit the data distribution. A suitable line can always be found for reasonable data (a straight line, a curve line or a hyperplane). This line does not have to cover every node, but it can describe the direction of the data.

c) **Clustering.** Clustering is to automatically divide the data into several different clusters. It seems to be the same type of identify the type of data as classification. The difference is that clustering does not know the type information. It automatically divides the data into different categories according to data's different features.

The ideas of other tasks are close to these.

2.2.2 Performance

P (Performance) is the metric required to evaluate the capabilities of machine learning algorithms.

For classification and clustering tasks, the accuracy of a model is always used to measure its performance. Specifically, it is the proportion of correct samples to the total sample.

For regression, common method of verifying the quality of model is to calculate MSE (Mean-Square Error) of the fit line and real data points. The MSE equation is:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - f(x_i))^2$$

In order to finish performance evaluation, a part of the dataset is taken out as a test set. When the training is over, test set is used to test the performance of models.

In current ML projects, another part of the total data set is often extracted as a validation set besides test set. The validation set is generally used to implement Cross-validation (Kohavi, 1995). The purpose of Cross-validation is to add non-training data during training process to avoid over-fitting of the model.

This leads to two main challenges in ML area: underfitting and overfitting. Both of these two concepts are specific to the generalization capabilities of the model. Generalization means that a model can still maintain good output with unseen input. Good generalization ability is also the goal pursued by ML. If a model does not provide good generalization in training set, then it appears underfitting. When the model has excellent generalization performance on training set and bad generalization on test set, it appears overfitting. Examples of underfitting and overfitting can be directly illustrated by figure 2.2.

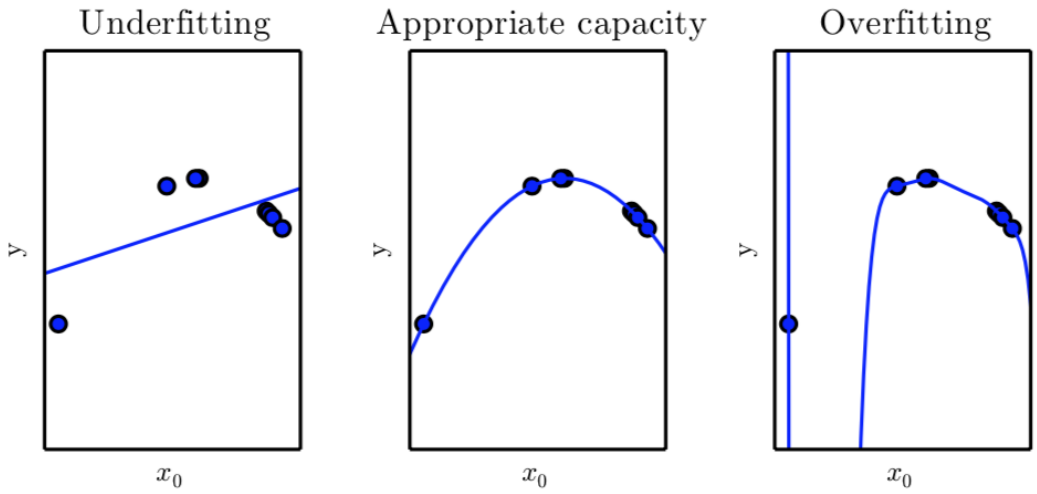

Figure 2.2 Examples of underfitting, appropriate fitting and overfitting (Goodfellow et al., 2016, p.111)

2.2.3 Experience

E (Experience), as talked before, is the presentation form of data. Different data has different characteristics, but all data can be roughly divided into two categories: data for supervised learning and data for unsupervised learning.

Supervised learning refers to that the algorithm knows every sample's corresponding result in the training set during training. Typical supervised learning algorithms are the classification and regression mentioned above.

Unsupervised learning refers to that the algorithm does not know any sample's corresponding result in the training set during training. It spontaneously brings together data from same category. Typical unsupervised learning algorithms is clustering in the context.

2.2.4 ML development history

Although the concept of ML was proposed very early, scientists still tend to give machine logic reasoning ability instead of learning ability in the early development of the AI field.

It was not until the 1980s that ML became the mainstream branch of the AI field. From then on until the 21st century, ML was dominated by symbolic ML (e.g. decision tree) and statistical learning (e.g. Support Vector Machine). The rapid development of hardware calculation power after the 21st century has made the multi-layered Artificial Neural Network (also called Deep Neural Network) a mainstream (Zhou, 2016).

2.3 DL (Deep Learning)

In theory, the more complex the model is, the more it can adapt to complex learning tasks (Zhou, 2016). And because of the complexity of the structure, its learning ability is also stronger. But complex models mean low training efficiency. This is the reason why complex models only have been widely used in recent years.

Deep neural networks, based on complex models, have experienced vigorous development in these years due to the improvements of computing resources. Deep Learning has become the most famous area in Machine Learning.

Most of the features of Deep learning are basically the same compared to Machine Learning, and the main improvement of it is that massive amounts of data can be handled without reaching the critical point of performance too early.

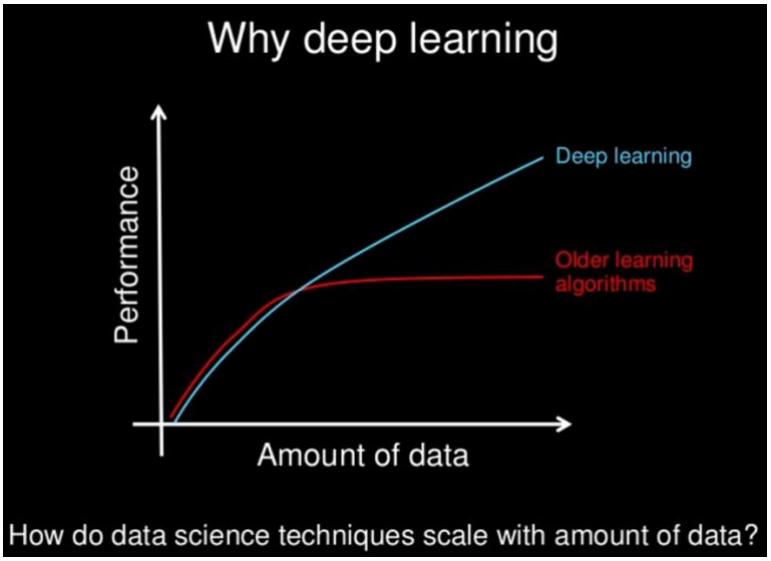

Figure 2.3 Performance on large amount data (Brownlee, 2016)

2.4 Deep Neural Network

Neural networks are very old models that were originally created to make machines works like human brain. Because of this, it can solve different machine learning problems well (Zhou, 2016).

The human brain receives the signals from the various sense organs to obtain outside information. Neurons in human body handle the various signals. The basic structure of neural networks is depended on a unit which just like neuron.

It accepts input from the previous neurons and uses these inputs to do calculation. When the result of this calculation is larger than its own threshold, it activates itself and passes its output to next neuron.

A complete neural network can be seen in figure 2.4.

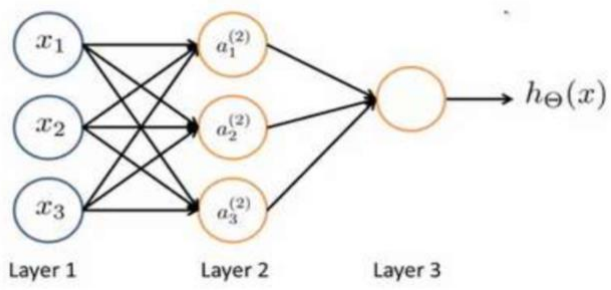

Figure 2.4 Deep Neural network

The above figure shows a three-layer neural network. The first layer is the input layer, the last layer is called the output layer, and the middle layer is the hidden layer. If a neural network needs to output multiple categories (multi-class classification), increasing the nodes of the output layer will fulfill this requirement.

2.4.1 Forward Propagation

The working principle of a neural network is to receive the output of the previous layer as the input of the current layer, and then calculate whether the current node needs to be activated or not. The calculation is according to the weight between the nodes in neighboring layer.

This left-to-right algorithm is called the forward propagation algorithm. For the model shown in figure 2.4, If represent x, θ and a by matrices, it can be transferred into :

$$\theta \cdot X = a.$$

2.4.2 Loss Function

Loss function is used to describe the difference between the prediction results of the neural network and the real results. Once the Forward Propagation is finished, the loss of the current model can be calculated from the output of the neural network and the actual results. One of the loss functions can be illustrated as :

$$\mathrm{E} = \frac{1}{2}\sum_{j=1}^{L}(y_j^k - a_j^l)^2.$$

The process of finding best weights for the neural network can be seen as the process of seeking the value of a small loss.

2.4.2 Gradient Descent

As mentioned previously, the learning procedure of neural network can be seen as to calculate weights in the neural network based on input and output data in the training set. This is actually solving the nonlinear equations. When there is enough data, reasonable weights can be found. But when the neural network is complex, such mathematical operations are very difficult to finish. Even replacing the equations with matrices is still not a simple task.

Gradient Descent is a commonly used method for machine learning models that greatly simplifies the computational process.

The weights that satisfy the demand of neural networks make the value of the loss function zero. In general, a good model only needs this value to be small enough. Gradient Descent is used to minimize the above loss function.

Using the knowledge of calculus to find the extreme point of loss function is the way to find minimum value. The derivative is enough for this problem. But in Deep Learning, the neural network parameters might have millions of parameters, so it is impossible to directly calculate the derivative.

Think of the minimize process as a ball scrolls down in data space in the figure 2.5. Along the direction of the "maximum derivative" direction in the model, it is most likely to find a minimum point with the fastest speed. This method is called Gradient Descent. It may also encounter problems such as local minimum and global minimum, but these situations are generally avoidable.

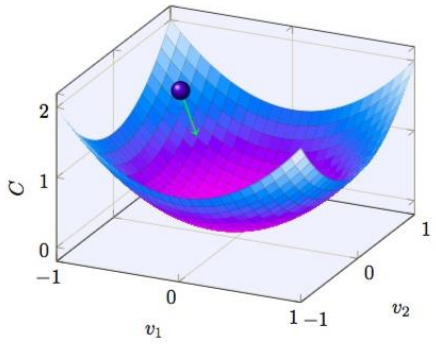

Figure 2.5 Gradient Descent in three-dimensional data space

2.4.3 Back Propagation

Most of the neural networks use backpropagation as the way to optimize models.

The idea of backpropagation is that calculate the previous layer's contribution to the network's loss (the loss function mentioned previously) and then do this process from the last layer to the previous layer until the algorithm reaches the initial input layer.

This backward process effectively measures the error gradient of all the connection weights in the network, and finally optimizes the weights of each layer by applying the Gradient Descent algorithm in each hidden layer.

2.5 CNN

There are many other types of neural network models: SOM (Self-Organizing Map), Hopfield Network, Boltzmann machine and so on.

One of the these that deserves a special mention is CNN (Convolutional Neural Network). The closest model to today's CNN model is LeNet-5 (LeCun et al., 1998).

Imagine using a regular neural network for image classification. For a 256 pixels by 256 pixels image, input layer of neural network requires 65536 input nodes. If using a multi-layer neural network, the required parameters can reach millions levels. Even with the support of well-developed hardware today, training such a neural network is still a very time consuming task. So, an approach called "weight sharing" has been proposed. It lets a group of neurons share the same weights. CNN is using this strategy.

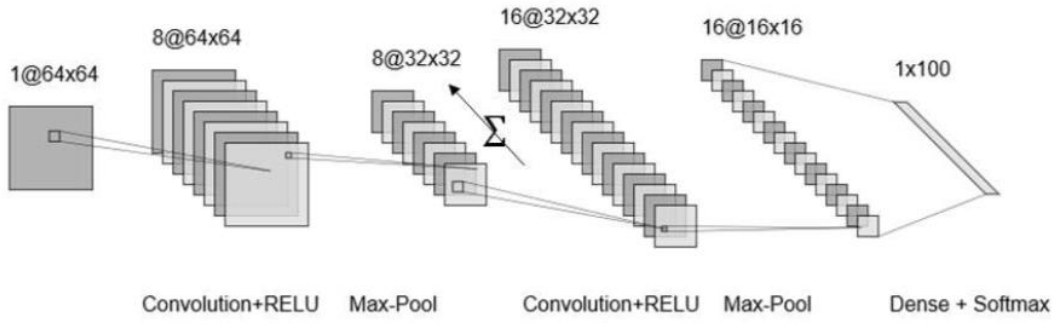

Figure 2.6 CNN

The CNN uses multiple groups of convolutional layers and pooling layers to process the input data and then map the output target at the fully connected layer. The convolutional layer uses the convolution method to extract the features of input data. This approach reduces the number of parameters and keep the effect of feature extraction at a maximum level. The pooling layer is used to sample data based on local correlation. For example, adjacent pixels are likely to contain common information in one image. This reduces the amount of data while retaining useful information.

There are many extended neural network models based on the convolution structure of CNN. Some well-known networks are AlexNet (Krizhevsky et al., 2012), VGG Net (Simonyan et al., 2014), U-Net (Ronneberger et al., 2015) and ResNet (He et al., 2016).

2.6 GAN (Generative Adversarial Network)

GAN is a new model in Machine Learning area introduced by Ian Goodfellow (2014). It has the ability to generate enough realistic fake data. This characteristic comes from the structure of GAN. Basic GAN has two main independent models, one is G (generator), another is D (discriminator) (Isola et al., 2017).

G is a network that can generate pictures. It receives a random noise z and then generates a picture through this noise. This can be described as G(z).

D is a discriminating network that discriminates whether a picture is real one or fake one. Its input is x (x represents a picture) and the output D(x) represents the probability that x is a real picture. If D(x) is 1, it means that the picture is 100% a real picture. If the output is 0, it means that it is impossible to be a real one.

The generator will learn to generate fake images which can fool the discriminator during the training phase. The discriminator will learn to distinguish fake images that are generated by Generator and real images at the same time.

The relationship between Generator and Discriminator is adversarial and both of them will constantly achieve their own optimization through confrontation in the training process.

In the most ideal state, G can generate a picture G(z) that is spoofed. For D, it is difficult to determine whether the picture generated by G is true or not.

For the entire GAN network, its rule can be expressed by the following equation :

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(x)}[\log(1 - D(G(z)))].$$

Parameter x represents a real picture, z represents noise input to the G network, and G(z) represents a picture generated by the G. D(x) indicates the probability that the D network determines whether the picture is real or not. D(G(z)) is the probability that D judges whether the picture generated by G is real or not.

The purpose of G: as mentioned above, D(G(z)) is the probability that D judges whether the picture generated by G is real or not. G should hope that the picture generated by itself is closer to the truth. In other words, G wants D(G(z)) to be as large as possible and V(D, G) to be as small as possible. So we see that the first token of the expression is smallest for G.

The purpose of D: The stronger the ability of D, the larger D(x) should be, and the smaller D(G(x)) should be. At this time, V(D, G) will become larger. So the formula is the largest for D.

The D and G training process can also use the Gradient Descent method.

2.6.1 Generator

Generator is a model that produces specified distribution data. In this case, it has the ability to generate images which are similar to the training data. Generative models generally have a special distribution for generating samples. For example, if input a random distributed data, the generator will model the output according to the probability density function of the generated distribution. So, the input sample will be transformed to the sample obtained with the specified distribution.

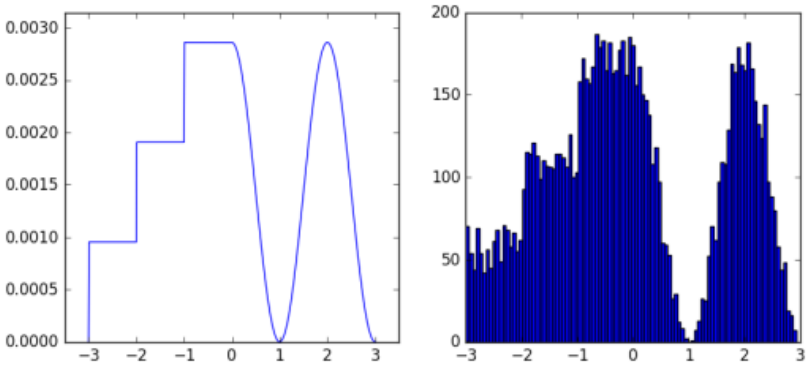

Figure 2.7 Distribution sample of Generator

The generator's training process optimizes its own parameters to maximize the likelihood function under the currently observed samples. This is also known as MLE (Maximum Likelihood Estimation).

MLE is a method for estimating the parameters of a probability model. Its process is illustrated by the following equation :

$$\hat{\theta} = arg\max_{\theta} P(x|\theta) = arg\max_{\theta} \prod_{i=1}^{n} P(x_i|\theta).$$

MLE is a basic idea. In practice, many projects use KL divergence (Kullback–Leibler divergence). Assuming that the real distribution is P and the sampling distribution is Q, the KL divergence is :

$$D_{KL}(P||Q) = \sum_{x\in\Omega} P(x) \log \frac{P(x)}{Q(x)}.$$

The KL divergence describes the difference between the two distributions. So, minimizing the KL divergence is equivalent to MLE.

2.7 image-to-image System based on GAN

GAN is clearly the most suitable neural network model for this project. According to the project's goal, an map image needs to be used as input and then a realistic satellite image is expected as output. This is obviously an image-to-image system. Image-to-image translation system based on GAN has also been proposed by many researchers.

2.7.1 Pix2Pix GAN

The work of Pix2Pix GAN (Isola et al., 2017) has attracted a lot of attention when it was first released. It uses GAN's framework to solve the problem of general Image-to-Image translation. Previous GANs networks can implement functions like: change images into different styles, turn the day in the image into a night and so on. Pix2Pix GAN network is able to achieve all the functions because all these tasks are mapping from pixel to pixel. The name of Pix2Pix is also referred to pixel to pixel.

Pix2Pix GAN is based on cGAN (Conditional GAN) which is proposed by Mirza et al. (2014). The improvement of cGAN compared to GAN is that it adds support for conditional constraints. It can be expressed using the following equation :

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(x)}[\log(1 - D(G(z|y)))].$$

As can be seen from the equation, it only adds the condition y. Both D and G must satisfy the prerequisite y.

For an image translation task, input of G should obviously be an image x and the output is an image y. Pix2Pix GAN changes the input of D. Because in addition to generating a real image, it is also necessary to ensure that the generated image matches the input image. The input of D should also include the original image x. This can illustrated by the equation :

$$L_{cGAN}(G,D) = E_{x,y}[\log D(x,y)] + E_{x,z}[\log(1 - D(x,G(x,z)))].$$

In addition, G shares a lot of information between the input and output (i.e. image colouring tasks sharing side information between input and output). Thus, in order to ensure the similarity between the input image and the output image. It is necessary to add a L1 loss which is :

$$L_{L1}(G) = E_{x,y,z}[||y - G(x,z)||_1].$$

The final equation is:

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

The Pix2Pix GAN also uses a U-Net as the structure in Generator to retain pixel-level details at different resolutions. D in Pix2Pix GAN is implemented as Patch-D. The so-called patch means that no matter how large the generated image is, input of D is divided into multiple fixed-size patch In this way, L1 loss can prevent global distortion and D can ensure that the patches of image are accurate.

2.7.2 Bicycle GAN

From the introduction of Pix2Pix in the previous section, it realizes the translation of one image to another. This relationship is one-to-one. But many times, what is needed is a one-to-many translation. Bicycle GAN (Zhu et al., 2017) achieves that one input image can be translated to multiple output images. Many contributors of Bicycle GAN are also contributors to the Pix2Pix GAN.

Bicycle GAN combines two GANs: cVAE-GAN (Conditional Variational Autoencoder GAN) which is introduced by Bao et al. (2017) and cLR-GAN (Conditional Latent Regressor GAN).

cVAE-GAN learns the implicit distribution of image through VAE (Variational Auto-Encoder) and multi-style output distribution method. cVAE-GAN starts with the ground truth target image B and encodes it into the latent space. This latent value are used to generate a

sample z with a random value. The generator then attempts to map the input image A with the sample z to the original image B. This sample z value is the key why Bicycle GAN can achieve multi-style images generation.

cLR-GAN starts with the implicit sampling of random samples. The condition generator should produce an output. When this output is input to the encoder, encoder should return the same implicit code. cLR-GAN randomly samples implicit coding from a known distribution, uses this encoding to map A to output B and then attempts to reconstruct implicit coding from the output.

The structure of Bicycle GAN Can be shown in figure 2.8.

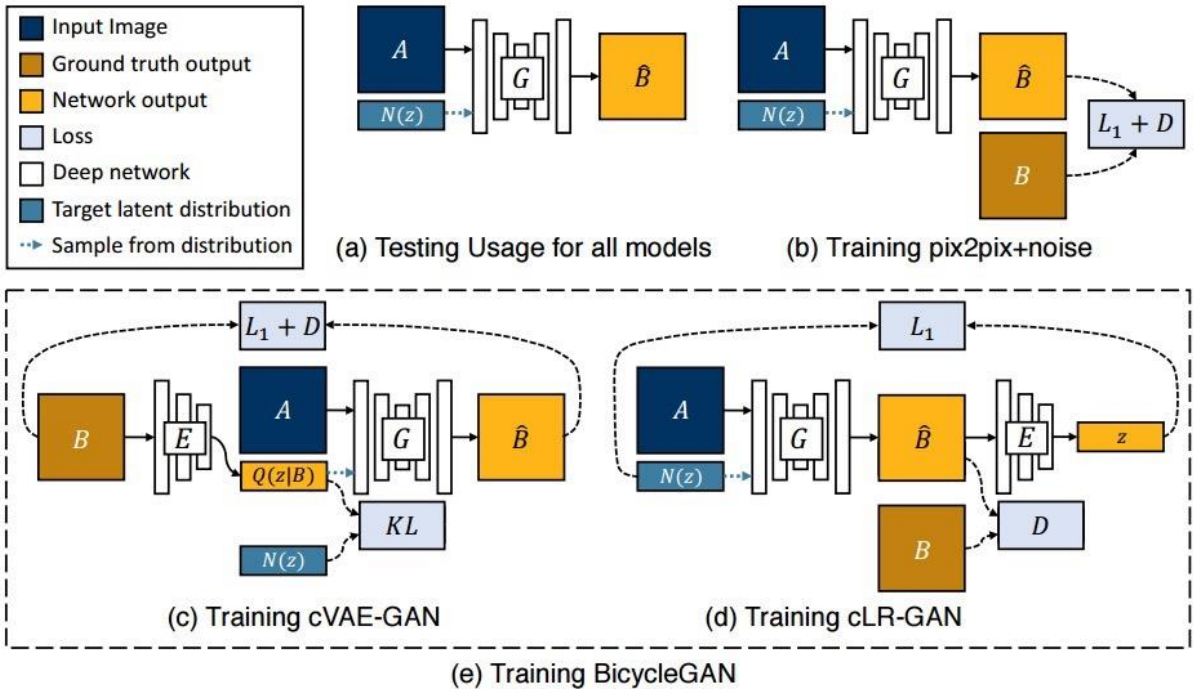

Figure 2.8 Bicycle GAN network (Zhu et al., 2017)

This can be illustrated by the equation:

$$G^*, E^* = arg \min_{G,E} \max_{D} (L_{GAN}^{VAE}(G, D, E) + \lambda L_1^{VAE}(G, E) + L_{GAN}(G, D) + \lambda_{latent} L_1^{latent}(G, E) + \lambda_{KL} L_{KL}(E)).$$

For generator G, U-Net is used. For discriminator D, two PatchGAN discriminators at different scales are used. For the encoder E, CNN and ResNet can be used.

Chapter 3 Preparation

3.1 Dataset

In machine learning, data is one of the most important parts of the whole system. Whether one algorithm or network model works well or not is hardly depend on the quality of data in dataset. Models trained with low quality or unreal data does not represent the real procedure. Valuable data is a good foundation for the project.

3.1.1 Data analysis

The core idea can be illustrated by an image-to-image demo available at: https://affinelayer.com/pixsrv/. The demo shows that one image which contains only edges of a cat can be translated into an enough realistic cat image. The difference between this demo and “Deep Generative Streets” project is that this demo is based on cat images and the project turns map images into real streets images.

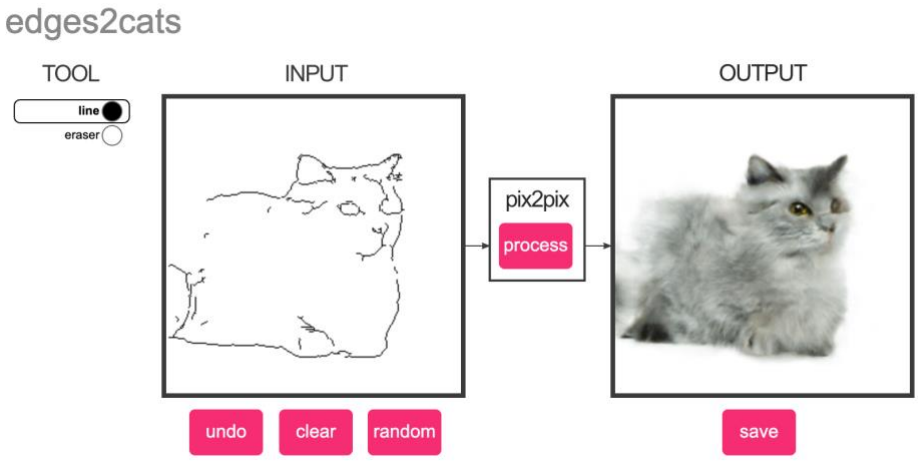

Figure 3.1 Image-to-image translation system example (Isola et al., 2017)

During the training phase, one suitable type of necessary data is a pair of images which one image is a map and the other is real streets. These two images should cover exactly the same area and they should have same resolution. Under this circumstance, the well-trained model can turn map images to real images.

3.1.2 Digimap

EDINA is a world-class centre for digital expertise, based at the University of Edinburgh. They are a division of Information Services and they provide many different kinds of data. For example, MANTRA, agcensus and Digimap. Digimap is a web mapping and online data delivery service developed by the EDINA national data centre for UK academia and it packs

up Ordnance Survey maps and data as pre-built customisable maps or spatial data for download. It is also the source of other map service providers (i.e. OpenStreetMap). Digimap is free for its members which mostly are education institutions in the UK.

There are nine varieties of map data accessible to users on Digimap:

Ordnance Survey, Historic, Geology, Marine, Environment, Aerial, Lidar, Global and Society.

The Ordnance Survey map data and Aerial map data are suitable for this project.

3.1.3 Ordnance Survey National Grid

World's most famous coordinate system is WGS (World Geodetic System) used by GPS (Global Positioning System). The newest version is WGS 84 (also known as WGS 1984, EPSG:4326). The accuracy of WGS is extremely high and it is said that the uncertainty is less than 2 cm.

Instead of using WGS 84, Digimap uses Ordnance Survey National Grid (also called British National Grid, BNG, OSGB 36 National Grid,EPSG: 27700). The Ordnance Survey have provided guidance on how to convert between OSGB 36 and WGS 84. The error of translation is controlled within 35 meters. Although the loss of accuracy is small, this project still tends to use OSGB 36.

Both the Ordnance Survey and Aerial map data are using OSGB 36 geodetic system. This means we can get the same area of data. In this project, the area from (425000.0, 430000.0) to (435000.0, 437000.0) was chosen. This totally covers 70 km by 70 km real landscape.

3.1.4 Products Choice

The Ordnance Survey provide various products (OS MasterMap, Meridian™ 2, Terrain™ 5, 1:10,000 Raster and so on). The best suitable product is "1:10,000 Raster". The "1:10,000 Raster" is consisted of paper and film 1:5,000 and 1:10,000 maps. The 1:10,000 map images can be used as the input of map data which is similar with the edges of a cat in the previous image-to-image demo. The format of images is "tiff" and the default resolution are 3200 pixels by 3200 pixels for 1km by 1km real area. One of the problems is that the "1:10,000 Raster" dataset is no longer updated since 2014. It is unsure how much impact there will be.

In case this dataset does not work well, it is necessary to find alternate data. Another suitable dataset is "OS MasterMap® Topography Layer" from OS MasterMap. This is topography layer can be downloaded as GML (Geography Markup Language) format file.

The GML is the XML (Extensible Markup Language) grammar defined by the Open Geospatial Consortium (OGC) to express geographical features.

GML data can be rendered to images with some GML-supported software. Digimap recommend users to use QGIS (QGIS Development Team, 2019) which is a free and open-source cross-platform desktop geographic information system (GIS) application.

The Aerial only provide one product: Aerial Imagery. It provides JEPG format satellite image files. The default resolution are 4000 pixels by 4000 pixels for 1km by 1km real area. This is the appropriate data source for training as real streets map.

Two examples from Ordnance Survey map data and Aerial map data are shown in figure 3.2.

Figure 3.2 Examples of downloaded data from Digimap. The left is map image and right is satellite image.

Through the analysis of some image samples, I found some defects in the data set which might have possible effects. The first is that there is some text (red box in left image of figure 3.3) in the map image which have mentioned earlier. Next is that some trees in the satellite images obstruct the road or building (red box in right image of figure 3.3). Finally, there may be some extra information in the satellite images that are not on the map images (e.g. cars on the roads shown in the red box in central image of figure 3.3).

Figure 3.3 Flaws in data

3.2 Program

There should be a program that can be used for training the GAN project needed and this program should have the ability to restructure its architecture for further experiments.

3.2.1 Programming language

The Python programming language was used for coding in this project. Python is a well-developed, high-level and dynamically typed programming language. It first appeared in 1990 and due to the development of computer hardware it has become more and more popular in recent years.

Python has many advantages. For example, it is a cross-platform programming language which means it support all the mainstream operating system. Meanwhile, it is garbage-collected. In this case, users do not need to consider memory allocation and release issues. It is really convenient compared to some static programming language (i.e. C++). The most important part is that Python has lots of third-party libraries with high quality. It is said that PyPI (Python Package Index) has over 130,000 various functional packages int the official repository. Some very well-known Python libraries, like NumPy, OpenCV and PyTorch, were used in the project.

3.3 Open source mature code

Although the Deep Learning library like PyTorch provide convenient development for setting up neural network, building a complex neural network from scratch on this basis is still not a simple task. In order to achieve the best results, developers may need to test and adjust the structure (what model is used by the generator) and parameters (learning rate and so on) many times. This also consumes a lot of time and may affect subsequent tasks.

It was suggested that more mature open source code could be used so that time can be used at the core part of the project.

The pix2pix GAN and Bicycle GAN introduced in Chapter 3 both have their official implementation. Their code is very mature and gives a variety of parameters to adjust any part of the network. The repository of pix2pix GAN code can be found at https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix and the Bicycle GAN can be found at https://github.com/junyanz/BicycleGAN.

3.4 Environment for developing, training and testing

The depth of Deep Neural Network makes it have excellent learning ability. But it also brings one disadvantage: a high demand of hardware resource. For instance, the styleGAN (Karras et al., 2019) requires 14 days 22 hours training time for 256 pixels by 256 pixels images on one core of a Tesla V100 GPU.

The neural network structure used in this project is not as complicated as styleGAN, but it also has very high requirements for hardware configuration. If using a laptop to train the network, it will waste a lot of time waiting for the training procedure to complete.

Fortunately, University of Leeds has a High-performance computer cloud platform which is called ARC (Advanced Research Computing). It allows users to submit task scripts to request computing resources. The maximum time for a single task is 48 hours.

According to the time requirements, one single NVIDIA P100s GPU card is enough for this project.

Chapter 4 Experiment and Analysis

After all the preparations have been completed, the next step is to do the investigation and experimentation. In this process, many methods were tried. But because some methods were completely unsuccessful, only a few core methods are described in this chapter.

4.1 Data Pre-processing

To create the training set needed, it is necessary to do some pre-processing with data. As mentioned earlier, the default map image is 3200 pixels by 3200 pixels for 1km by 1km area in real life, and the satellite image is 4000 pixels by 4000 pixels for 1km by 1km area. Although the hardware of ARC is good enough, it still cannot support direct training with such high-resolution images. The memory cost is huge. Even if it does have such huge memory, training time is still a problem which need to be considered. If one training process costs more than a month, then time will be wasted on waiting.

The solution is to cut a large image into small images for training. 256 pixels by 256 pixels is a reasonable resolution. To achieve this goal, there are two main tasks:

a) Extract small and non-repeating images from large images.

b) Match the corresponding map images and satellite images and turn them into a pair of data.

A python script and a bash script were written to finish these two task. The python script has following features:

a) It mainly accepts two file paths (a high-resolution map image and a high-resolution satellite image) for generating corresponding pairs of data.

b) Resizing these two images to 3072 pixels by 3072 pixels. This is because 3072 is exactly divisible by 256. It has been verified before that two images with different resolutions can still be perfectly matched after scaling.

c) Then cut two images separately, and the results of the cutting are stored in different folders. According to the preset resolution, a large image can be divided into 144 small images.

d) Change the pictures in two folders into the form of A to B and reasonable name them. The neural network can directly cut such input images into map and satellite images by matrix slice function supported by NumPy. An example of the final data can be shown in figure 4.1.

Figure 4.1 An example of final dataset

The bash script is used to call python script multiple times to achieve the cutting of all high-resolution images. The order of each image being processed by the bash script is based on the default file name from Digimap. Figure 4.2 shows the relationship between the naming of small images and downloaded images.

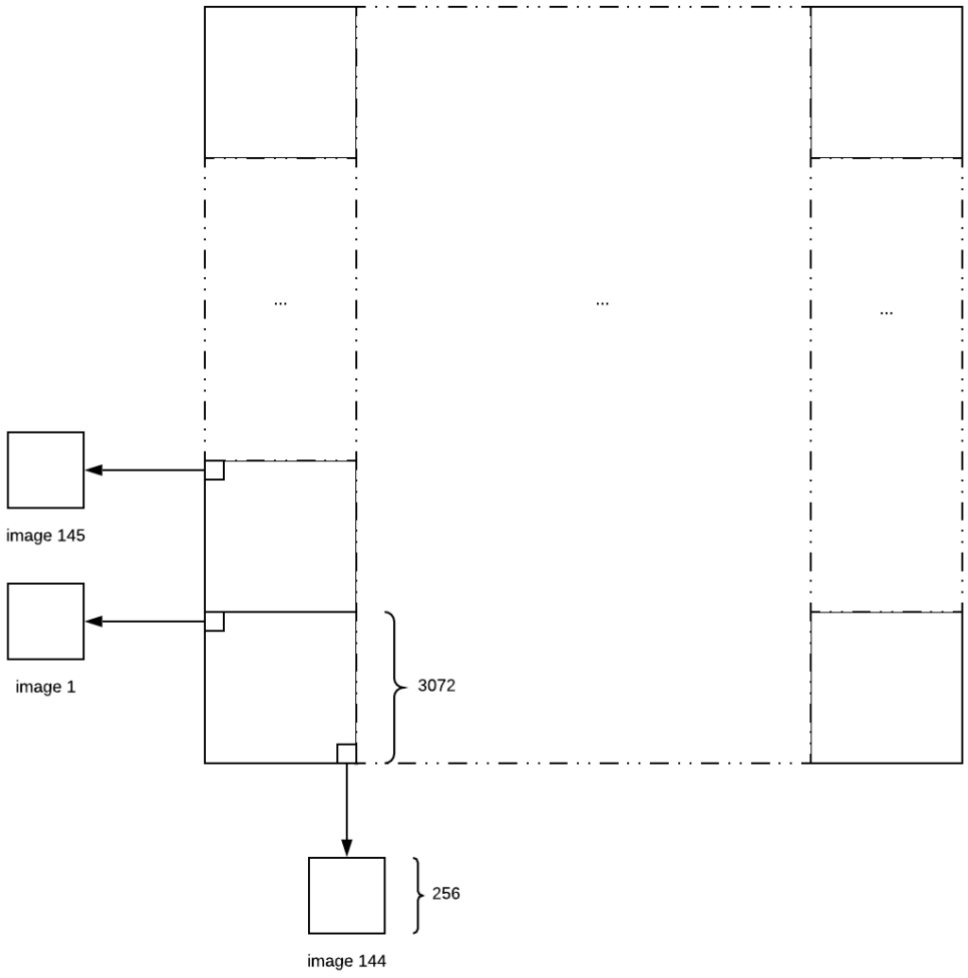

Figure 4.2 The naming relationship between the small images and large images

4.2 Basic training with Pix2Pix GAN and Bicycle GAN

In the original plan, Pix2Pix GAN and Bicycle GAN should be implemented from scratch. But after several days' attempt to develop Bicycle GAN from scratch, it was confirmed that a lot of time will be spent on this. Due to it is more hopeful to achieve high-resolution image generation, the decision of using official implementation of Pix2Pix GAN and Bicycle GAN was made ultimately. In this way, time can be spent on the core research part of the project. This turns out to be a correct decision due to the final image synthesis is extremely difficult. Using their framework as foundation saved a lot of time.

4.2.1 Official Implementation

The official code structures of Pix2Pix GAN and Bicycle GAN networks are similar. They both have four core modules based on Python:

a) Data. Create dataset for training or testing. It support various format of data. The format of “aligned” (similar to the A to B format mentioned before) is used in this project.

b) Models. This module includes a variety of basic neural network architectures (U-Net, ResNet and so on). Then use these basic networks to build up Pix2Pix GAN and Bicycle GAN.

c) Options. This module is responsible for a variety of parameters settings (basic parameters, training parameters and test parameters). It basically includes all adjustable parameters that are common in machine learning. Most of the parameters have been set to excellent default value. This can save users a lot of time to reach good results.

d) Util. This mainly focus on some useful function. For example, it provides a web server which allows users to see the loss graph of training and it illustrates test results in a suitable order in one html file (figure 4.3). This is convenient for users to observe the results.

Figure 4.3 Results views in final html file. The input is the map image used to generate output. The ground truth is the real satellite image for comparison. The encoded is the output of Encoder which is used to identify correct z value. Random sample 01 to random sample 05 are the generated streets images.

The official implementation also gave some user cases and training/test tips which are useful.

4.2.2 Basic Training on Pix2Pix GAN

The training started with Pix2Pix GAN in the beginning. These are some important parameters that have not changed for all training tasks:

a) The number of input and output channels is 3. This means the network receives a 3 channel (RGB) image and output a 3 channel (RGB) image.

b) The generator uses ResNet as its structure. The discriminator uses 70 pixels by 70 pixels Patch GAN as its structure.

c) The batch size used for training is 1.

d) The images' load size is 256. They are not resized and cropped by data augmentation.

e) The total epoch number is 200.

f) Images are flipped randomly during training phase but not in the test phase.

g) The training model is saved every 5 epochs. But only the 50, 100, 150 and 200 epoch's training models are kept because one model uses a lot hard disk space,

Initially, Pix2Pix2 was trained using a total of 1000 images (including training set, validation set and test set). This successfully generate real streets images from map images as expected. But the result is not so good overall, it can be illustrated by figure 4.4. Although houses, roads and cars have been successfully built, they are kind of curved. The roof of the houses, the shapes of the houses itself and the shapes of roads are all distorted. This is far from the real streets view.

750

Figure 4.4 Pix2Pix GAN result example 1. The real_A is the input map image. The fake_B is generated streets image. The real_B is real satellite image.

Next, some adjustments including data augmentation, different resolution, and using a larger dataset were tried. Increasing the amount of data is the most intuitive way to improve results. New results are shown in figure 4.5. The roads this time looks natural. But, still the houses and roots have bends (although these curves are already very close to the straight line).

750

Figure 4.5 Pix2Pix GAN result example 2. Houses and Roads are shaper than before compared to figure 4.4.

Then, it was expected to get better results by increasing the dataset about 10,000 images (this is the total number of all images). But it is backfired, results still have curved houses and roofs. This problem may be caused by ResNet structure used in Generator. Although ResNet has an extremely high ability of extracting image feature, the convolution operation in the convolution layer of the original image may result in the image distortion. In the

Pix2Pix GAN paper, the illustrated images of urban images processing looks similar. They have curved houses as well.

Although the results of Pix2Pix GAN is very powerful, it is not suitable for this project because the handling of houses and roads is not good.

4.2.2 Basic Training on Bicycle GAN

Bicycle GAN was tried next. Some parameters that have not changed are listed:

a) The number of input and output channels is 3. This means the network receives a 3 channel (RGB) image and output a 3 channel (RGB) image.

b) Training batch size is 2 this time.

c) The images' load size is 256. They are not resized and cropped by data augmentation.

d) Default latent vector z's size is 8 (This is the size of sample z which has been mentioned in Background Research chapter).

e) The total epoch number is 200.

f) Images are flipped randomly during training phase but not in the test phase.

g) The generator uses U-Net as its structure. Bicycle GAN has two Discriminators, one uses 70 pixels by 70 pixels Patch GAN and another uses 140 pixels by 140 pixels Patch GAN. The Encoder of Bicycle GAN is ResNet.

h) The training model is saved every 5 epochs. Because the model uses a lot hard disk space, only the 50, 100, 150 and 200 epoch's training models be kept.

i) During test phase, network generates 5 samples of each input image with 5 random sample z value.

Because Pix2Pix GAN does not perform well on dataset of 1000 images, dataset of 3000 images is initially used for Bicycle GAN. The results of the Bicycle GAN are better than Pix2Pix GAN, and there was no house bending problem. This can be seen in figure 4.6.

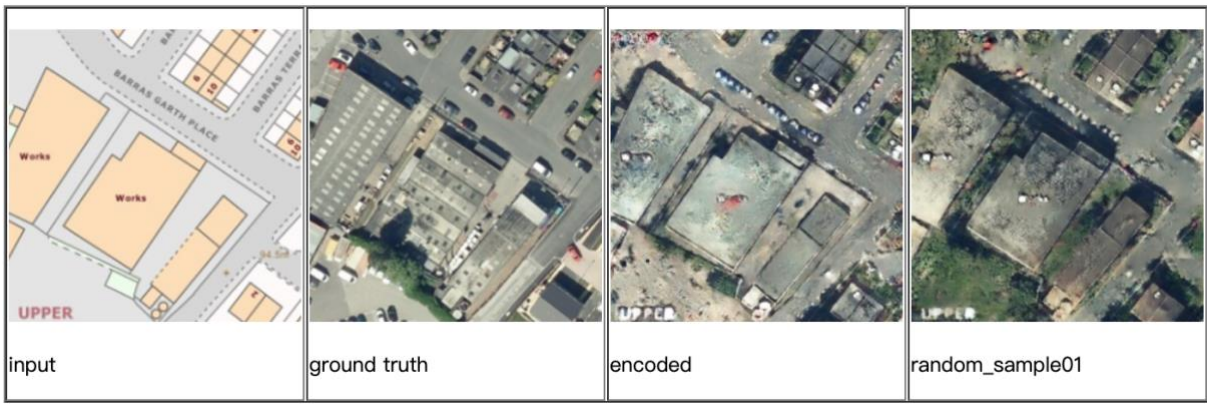

Figure 4.6 Bicycle GAN result example 1

Bicycle GAN with dataset of 1000 images and 10000 images were also tried. Surprisingly, Bicycle GAN did a good job even with 1000 images, and it failed at 10000 images. The reason why it performs well on 1000 images is because a random sample z is used each iteration of training. This means that same image in every iteration becomes a new image after adding a different sample z. This method is actually equivalent to increasing the size of dataset. The reason for the poor performance on the 10,000 image may also be due to this. 10000 images are very large themselves, and it becomes more massive after adding random samples z. The training duration of 200 epochs is too short to achieve the best training results. It can also increase the training time, but the ARC training environment can only be used continuously for 48 hours at a time. In this case, it is not necessary to increase the training time to train 10000 images. So the training following mainly is using a dataset of 3000 images.

In view of the excellent performance of Bicycle GAN, the project chooses Bicycle GAN.

4.3 Generated Images Analysis

The basic aim of the project have been achieved. Conversion from map images to real streets images has been implemented. The quality of the resulting image is also relatively high enough. But there are still some problems worth analysing.

4.3.1 Low-resolution Level Analysis

For task to combine low-resolution images into high-resolution images, First, it needs low-resolution images of high quality. Analysis of low-resolution images is quite necessary.

It can be shown in figure 4.7. The input map generated two real streets images. The difference between random sample 1 and random sample 2 is that they use different z values. Random sample 2 does differ from the style of random sample 1, but the area marked by the red rectangle has a strange pattern. This is because different sample z values are not enough for neural networks generating images of different styles. In this case, the

neural network changes some areas in the image to achieve different styles which causes this strange pattern. Issue like this also is also known as localised mode collapse.

This problem is a common issue of GAN and it has not been solved yet, since time is limit on this project and this only happens occasionally. Time should not be spent on solving this problem.

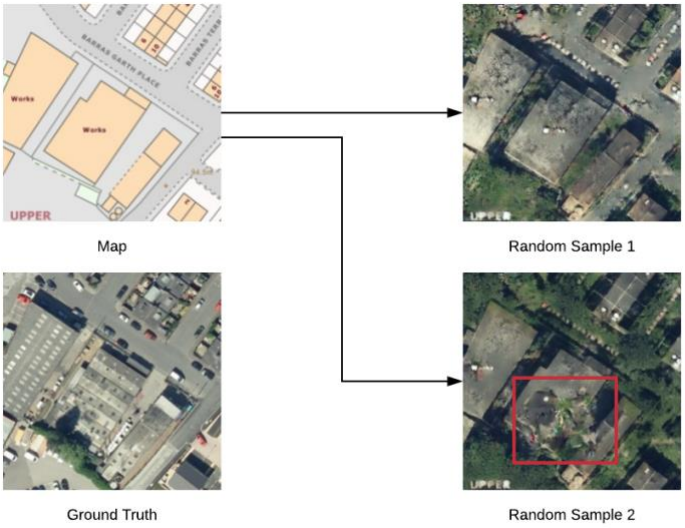

Figure 4.7 Strange pattern in Bicycle GAN result

The next problem is image inconsistence. This can be illustrate by figure 4.8. As the random sample 1 successfully generated real streets image, the random sample 2 is mis-generated as an image of the woods and grass. This is because of the uncertainty of the neural network itself and this well-known problem will not be investigated in this project.

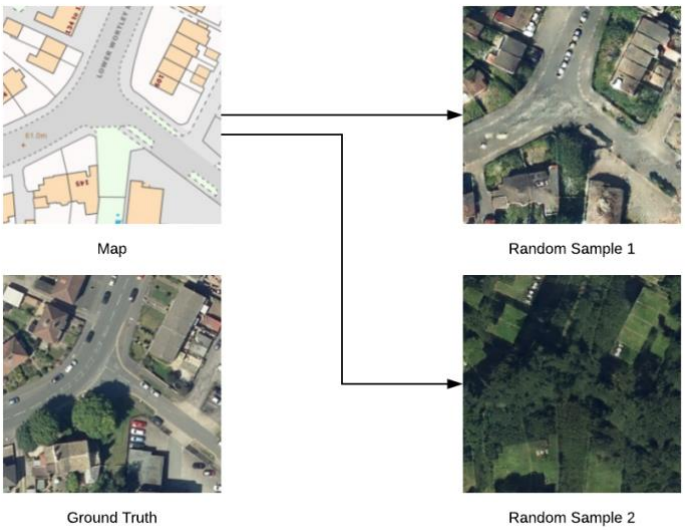

Figure 4.8 Mis-generation in Bicycle GAN result

As can be seen from figure 4.9, there are other three main problems of the generated images besides the problems mentioned before.

In Chapter 3, it has been mentioned that there are cars on the road in the ground truth images not map images. The result is the blue rectangle of figure 4.9 in the random sample 2 which points out a very strange car. The probability of this is pretty high and sometimes it leads to even more strange scenes (some twisted cars are parked on the roof of building). To solve this problem, the car could be manually erased in the satellite images from dataset. This is a very time consuming task and valuable time should not be wasted in this repetitive behaviour.

Next, the contents of the red box of figure 4.9 were also mentioned in the previous analysis of the dataset. Some trees next to the road obstructed the road which caused half of the roads in the random sample 1 to be covered by green plants. This occlusion looks unreal. This is also a problem with the dataset itself and it is difficult to solve it either.

The last problem is more serious. The map downloaded from Digimap contains text (building labels, road labels and so on). This kind of text does have a bad influence on the training of Bicycle GAN. This effect can be seen in the green box of figure 4.9. The text in the map image is also displayed in the generated image. The text is obviously trained by the neural network as a special feature which is not needed. The previous problems cannot be solved in a short time, but this problem can be. The key is the GML format data which is downloaded from Digimap mentioned previously.

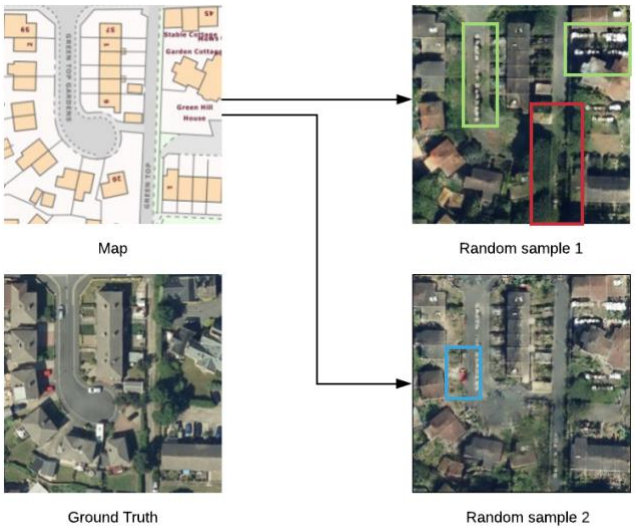

Figure 4.9 Other flaws in Bicycle GAN result

4.3.2 Generate Map Without Label

It took several attempts to create a map dataset without text.

The GML format data has been introduced before and the official operating software recommended by Digimap is QGIS. QGIS supports rendering GML to images. As long as GML data is rendered without text layer, a map image without labels can be exported.

However, there are too many description categories in the official GML data of Digimap and there are many interact descriptions. So, rendering a single image is not an easy job. While QGIS has Python plugin support, the official documentation is not detailed and it may take a longer time using Python doing this. Spending too much time on this task is not particularly cost-effective. It can be done by working on this in a week alone, but this will delay the core progress of the project. Due to these reasons, the progress of this task is not smooth.

Other methods were also tried to deal with this problem. After converting GML files to SHP (ESRI Shapefiles), rendering map images can be done by TileMill (Mapbox, 2019). TileMill is an open source architecture for handling map data based on Node.js. It does simplify the process translating SHP to image, but one problem is that TileMill does not support the OSGB36 coordinate system. Changing the coordinate system from OSGB36 to WGS84 will causes accuracy loss, so this solution is abandoned.

Finally, when looking carefully at resources provided by Digimap, a pre-defined layer style file (QML format) was found. It can be imported directly into QGIS software to make use of pre-defined styles. One problem here is that this QML file is old and is different from the description in the latest GML file. Some of the necessary changes have been applied to the QML file which makes it can be used finally. Simply selecting the coordinate field corresponding to a real satellite image can export the corresponding map image. The exported single map image also covers 1km by 1km area, but the corresponding resolution is not the same all the time (sometimes even the aspect ratio is not 1:1). After the test for adjusting the resolution of the image, it was certified that the map image generated by QML is also compatible with the satellite image.

In the process of exporting images, experiments were performed whether the lines layer is rendered or not (figure 4.10). The results was that the map with line have better performance during training. So the project uses map images with lines.

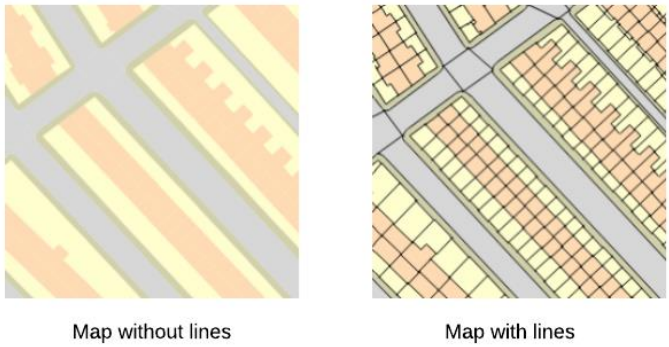

Figure 4.10 Map images exported from GML without line layer (left) and with line layer (right)

The comparison with the map images originally downloaded from Digimap is shown in figure 4.11.

Figure 4.11 Map images exported from GML (left) and map images downloaded from Digimap (right)

For the same input image that was previously shown by figure 4.5, the output example using new map image can be seen in figure 4.12. There is no more text noise in the new results.

Figure 4.12 New map images remove the influence of text

Although the content of this section is after the Low-resolution Image Analysis section, the actual process is actually parallel with the subsequent experiments. One day a week was spent on this task on average. The new map images were not generated until August 7. So the input map images in subsequent experiments may still be the original map images. This will be reflected in some following figures.

4.3.3 High-resolution Level Analysis

Now, the result of the small images can be generated and they can be combined together into a high-resolution images. A Python script that supports images synthesis has been written.

By default, it is conceivable that the combined picture must be far from the real picture. As mentioned earlier, Bicycle GAN uses five different z values to generate five different styles of output images for the same input during the test phase. The test script has one parameter, “sync”, to choose whether to use the same latent code (i.e. z value) for different input images or not. If used this parameter, all input images will produce same 5 different styles images which means that all images of random sample 1 will use same z. If didn’t use it, there will be 5 styles for each input image alone. Both of these methods have been tried. Two combined high-resolution images can be seen in figure 4.13.

Figure 4.13 Examples of combined high-resolution images. Seams can be clearly seen in these two pictures. Image on the left is result not using “sync”. The right image is result using “sync”.

The result of combined images generated using "sync" parameter shows a consistent style. But this will cause all green spaces in one shape. The result not using the "sync" parameter turns to be more variety. Green spaces have different forms in this one, but there are still many repetitions of similar shapes (the lower right corner of the image on the left of figure 4.13).

Both pictures have obvious seams can be compared to the real satellite image shown in figure 4.14.

Figure 4.14 Real satellite image

When evaluating these two results by human eyes, the results using "sync" parameter should be better (right image in figure 4.13) although it has lower variety. But it was suggested that there should be a standard of measurement. This means some evaluation criteria must be set rather than simply saying which one looks better. This is really important, especially in the situation when improvements are not obvious.

In order to have a reasonable evaluation criteria, a SE (Seams Energy) value was proposed. It has the ability to detect how many seams in one image. The detail of SE will be discussed in Chapter 5.

Returning to the analysis of the image, the next thing to do is to try to make the seams caused by the splicing disappear.

4.4 Seams Bicycle GAN

In order to solve the seams problem in high-resolution images, there were many reasonable ideas at the start. The following method discussed is the easiest way to be implemented and verified, so this method was tried first.

Its core concept is to create a new dataset which is based on the results of previous Bicycle GAN. Because the latter work in this also changed the structure of basic Bicycle GAN, this method is named as Seams Bicycle GAN instead of Seams Bicycle Dataset.

4.4.1 Basic Seams Bicycle GAN

The question how to use a neural network to eliminate a gaps in an image can also be seen as image-to-image translation. The difference between it and generating streets images from the map images is that it attempts to convert an image that contains seams into an image that does not.

In order to achieve this goal, the central part can be extracted as one single image (also 256 pixels by 256 pixels) from 4 adjacent generated images' combination. Then pair it with the corresponding part of the real satellite image to synthesize the new input and ground truth images pair form data. This procedure can be illustrated by figure 4.15.

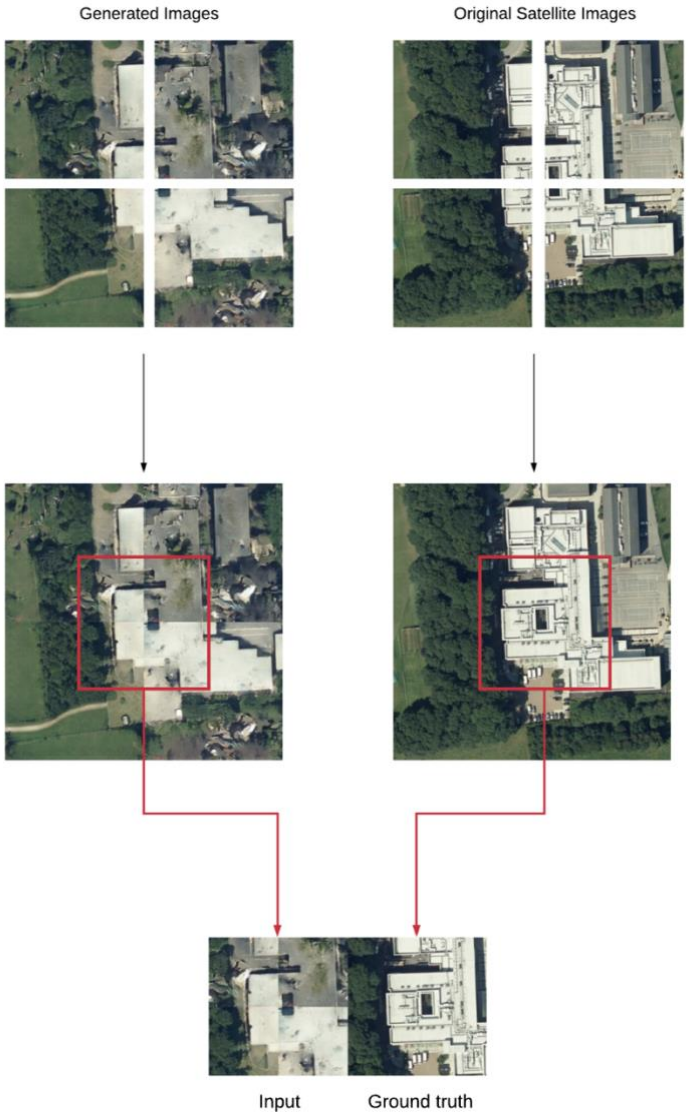

Figure 4.15 Creating new seams dataset

The dataset was created using result without the "sync" parameter. This is because images generated without “sync” are more diverse. It was expected that the neural network can remove the seams and transform diverse styles cross the seam image into a unified image at the same time.

The result of this training does eliminate the seams in the picture although sometimes it still has four different styles. This result is not very simple to get, many problems like lack of images are solved in the process.

One representative result can be seen in red box of figure 4.16. The most critical issue is that the road in original input is completely distorted after being processed by the Generator. This also led to the distortion of the road inside the final result. The shapes of the houses are also somewhat distorted, but not as severe as the results of the previous Pix2Pix GAN.

Seams Bicycle GAN generated new pictures without seams, but these pictures' quality and realism have been decreased.

Figure 4.16 Road distortion in results

4.4.2 6-channel Seams Bicycle GAN

In order to improve the results of basic Seams Bicycle GAN, a new idea was raised during thinking about if there is a way to add additional information to enhance the shape information of roads and houses.

The original map image data is the most suitable extra information the neural network can use. With the map images, a new method was proposed:

a) Modify the number of channels of input data to 6 channel. This means now the basic Seams Bicycle GAN has 6 input channels.

b) Add a new class of dataset to handle 6 channel input. A new Python file was added to the program as "aligned6c_dataset.py".

c) For new input data, the corresponding area from the map images also need to be extracted for each seams image.

This method is named as 6-channel Seams Bicycle GAN.

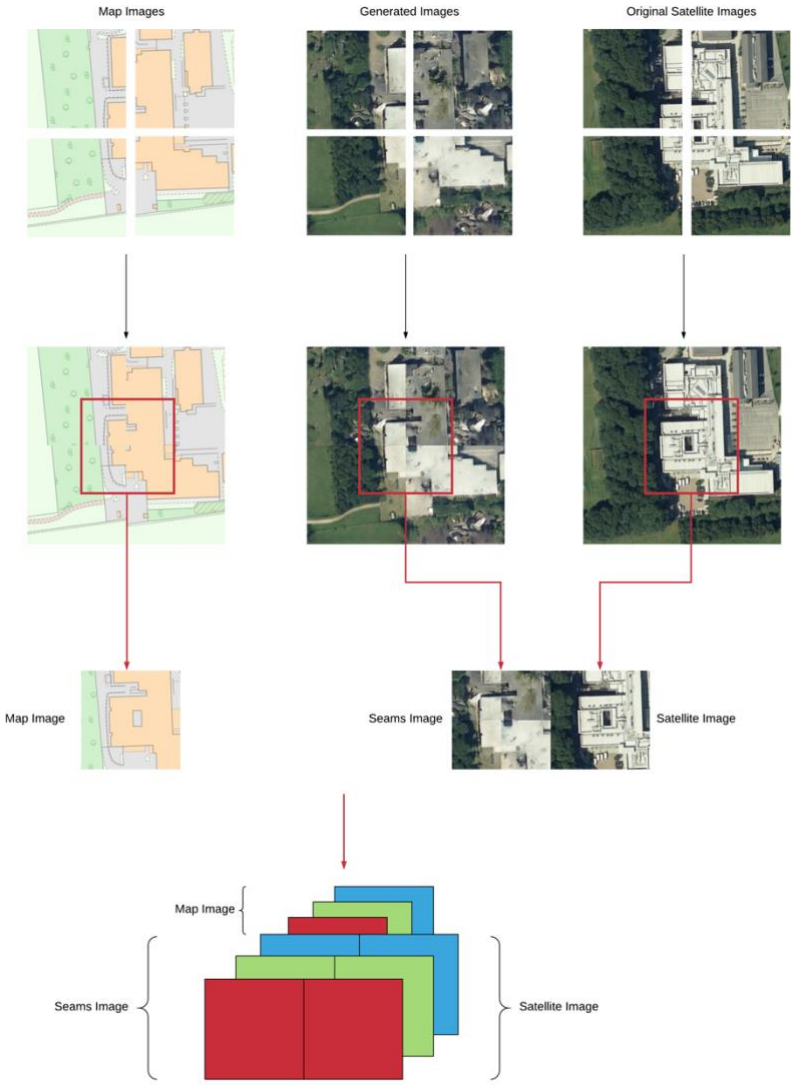

Figure 4.17 Add extra 3 channel in Seams Bicycle GAN

This method also encountered some problems like there are no suitable format for saving a 6 channel image. But fortunately, they have all been solved. The 6-channel Seams Bicycle GAN does achieve data supplementation to the input data as expected. The result is much better than the basic Seams Bicycle GAN. Figure 4.18 shows the results of same input in 6-channel Seams Bicycle GAN. First of all, it can be found that the road in the red circle area is basically the same as the real streets. Next, if look closely, the edge of the houses are also sharper than before.

Figure 4.18 Results of 6-channel Seams Bicycle GAN

4.4.2 A Trick Successful Case

Now the 6-channel Seams Bicycle GAN can eliminate seams and the quality of its results is still high enough. However, if the generated image without seams is directly applied to the previously combined high-resolution image, it still might not work.

The results of the previous Seams Bicycle GAN and 6-channel Seams Bicycle GAN show that the seams in one image are eliminated, but the edge areas of the generated image were also changed. In this case, new seams will be introduced when replace the extracted part to generated results. This is illustrated by the figure 4.19.

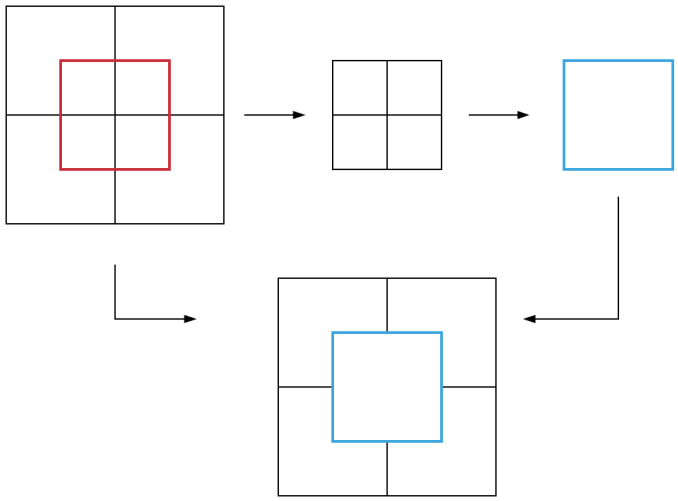

Figure 4.19 New seams will be introduced if replace result directly

The trick here is to use different resolution during the test phase. Seams Bicycle GANs are trained by 256 pixels by 256 pixels resolution images. When the training is complete, input the combined 3072 pixels by 3072 pixels resolution image directly to the neural network. This will generate a 3072 pixels by 3072 pixels resolution output image.

The structures of Seams Bicycle GANs support such operations. The only change required is to disable the encoder during test phase using parameter ("--no_encode"). This is because encoder's network structure does not support different resolution. The encoder is only responsible for providing encoded images during the test phase, so disabling the encoder has no effect on the final results.

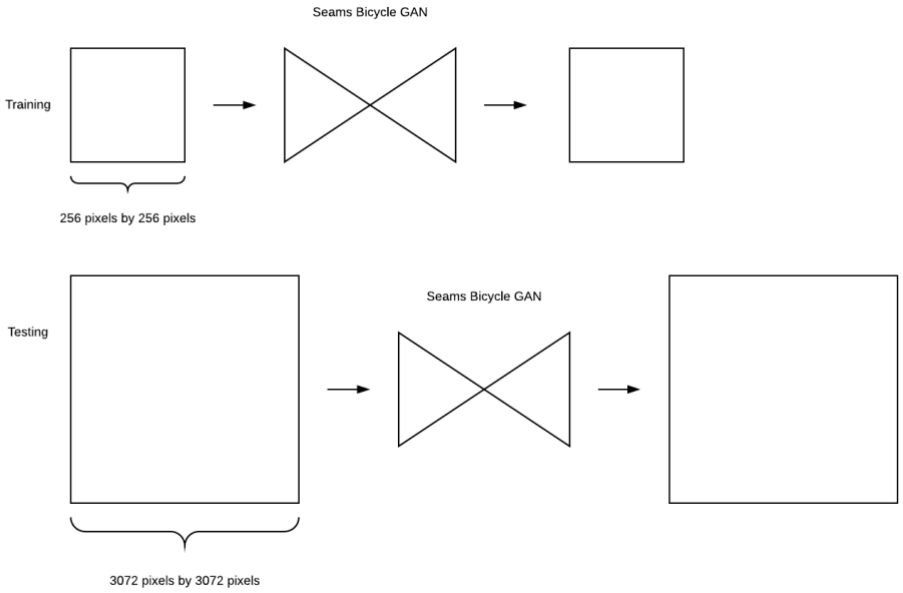

Figure 4.20 Using different resolution in test phase

This method can generate large images without seams. Both Seams Bicycle GAN and 6-channel Seams Bicycle GAN were tested and the results are shown in the figure 4.21. The real satellite image can be used to compare with these is illustrated in the figure 4.14.

Figure 4.21 Results of 3072 resolution images with seams being processed by Seams Bicycle GANs

The results show that although the overall image still looks like it is made up by piece images, it does eliminate the seams. The difference between 6-channel Seams Bicycle GAN and Seams Bicycle GAN is that the extra 3 channel map data in 6-channel Seams Bicycle GAN makes the results more consistent. For example, the colour of the roofs tend to be same.

Although these results prove the feasibility of Seams Bicycle GAN, it still cannot be said that the result of this generation is good. Since the image is too large, neural network generated strange texture in the edge area (orange box in figure 4.21). These images also does not have good views with human eyes. So this success is a trick success.

This still takes a lot of time and need excellent hardware resources to generate one high-resolution image. If the image is larger, the result might be worse. This approach also does not support consistently generating infinitely large image.

In this case, trying to generate a low-resolution image with suitable edge texture for direct synthesis is the next attempt.

4.5 Mask and New Loss Function

4.5.1 Add Mask

The 6-channel Seams Bicycle GAN removed seams, but it introduced new ones. In order to solve this, a new idea is to continue modifying the 6-channel Seams Bicycle GAN and try to let the neural network eliminate the seams while keeping the edge area of the output image as close as possible to the input image.

It has been known from the 6-channel Seams Bicycle GAN that adding extra channels can provide much better information. Based on this, new idea is to add an extra mask channel to see if the neural network can increase the pixels' weights of some areas in some images. In this case, the 6-channel Seams Bicycle GAN becomes 7 channel of input (7-channel Seams Bicycle GAN) after adding the mask channel.

In general, a mask is a grayscale image. All the mask images used for training are shown in figure 4.22. Some of them are black in the central and white in the border. This is trying to add spatial context to the input and let neural network focus on central of the input. Some masks are exactly the opposite, the middle is white and border is black. This is trying to let neural network focus on the border areas of images. Some are cross shapes trying to add weight on central seams. A circle shape is tried alone. Because of the time limitation, there were some masks that are not individually trained. They were trained in conjunction with the new loss function added in next section.

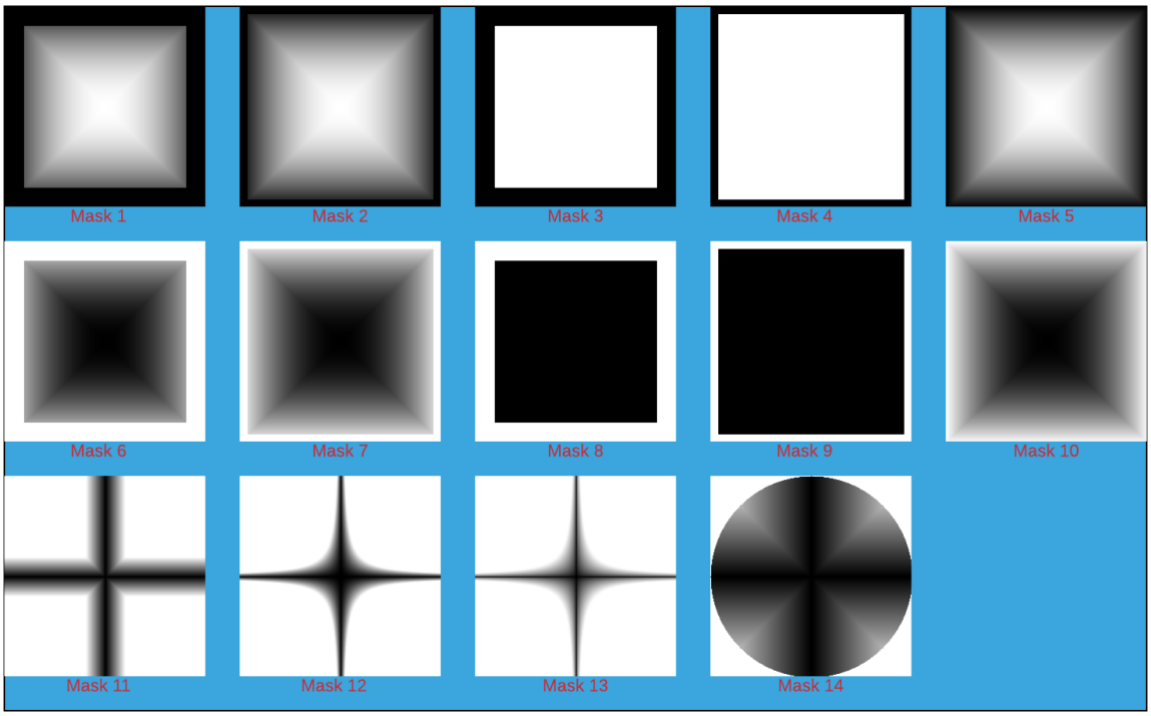

Figure 4.22 Masks added into the neural network

Some representative results are illustrated in figure 4.23. From these results, it seems to be difficult to tell whether these masks are in effect or not. But it is clear to see its styles are not as rich as before. Furthermore, neural networks have a big chance to generate large green areas for some masks with large black areas in the center (mask 8 random sample 3 and mask 14 random sample 2). This also shows that the additional channel of information is useful.

If these new generated images without seams are replaced by combined high-resolution image (a python script for this has been written), there is still seams in the image and there is no significant improvement by visual inspection. Two examples is shown in figure 4.24.

To measure the improvements, SE is used again to calculate a value that can be compared, these values will be discussed in Chapter 5.

Figure 4.23 Examples of Seams Bicycle GAN with different masks

Figure 4.24 Seamless results replaced back into high-resolution combined image

4.5.1 Add New Loss Function (Mask Loss)

The result of just adding a mask has no visible improvement. The reason might be because the neural network does not pay enough attention of this extra channel of mask. So, in order to enhance the effect of the mask, a new function loss called mask loss was introduced. These code is added in the “networks.py” file.

The mask loss function is based on MSE calculations. The main idea is to extend the mask to 3 channel by replicating it and multiply it by both generated image and input image during training phase. This multiplication is feasible because the colour values are normalized from -1 to 1 during training. In this way, the weights of some areas in images are enhanced.

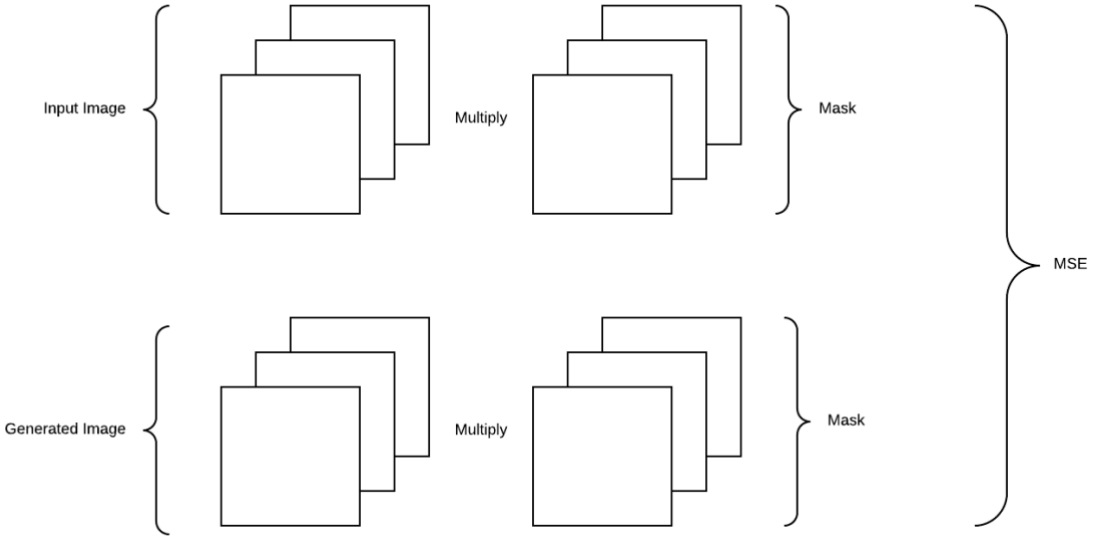

Figure 4.25 mask loss function working process

Figure 4.25 shows the work process of mask loss function. The equation of mask loss function is :

$$L_{ML}(\mathrm{G}) = \frac{1}{n}\sum_{i=1}^{n}\big((\mathrm{G}(\mathrm{x}) * \mathrm{mask}) - (\mathrm{x} * \mathrm{mask})\big)^2.$$

The total equation of 7-channel Seams Bicycle GAN is :

$$G^*, E^* = arg \min_{G,E} \max_{D} (L_{GAN}^{VAE}(G, D, E) + \lambda L_1^{VAE}(G, E) + L_{GAN}(G, D) + \lambda_{latent} L_1^{latent}(G, E) + \lambda_{KL} L_{KL}(E) + \lambda_{ML} L_{ML}(G)).$$

Although this new added loss is simple to explain, many bugs were fixed before it would run. The range of loss values with different masks is different. It takes a lot of time to find the appropriate weights for this value.

Some results of these training are illustrated in figure 4.26. The style is less diverse, but there is no problem of generating large green areas which happened before. The images generated in many different situations are indeed close to the input image.

Some examples of replacing the combined high-resolution image with these results are shown in figure 4.27. There is still no improvement that can be seen directly. So SE is needed again to calculate values for comparison. This will also be discussed in Chapter 5.

Figure 4.26 Examples of Seams Bicycle GAN with different masks adding new loss

Figure 4.27 Seamless results replaced back into high-resolution combined image

4.6 Spatial Varying Z

Finally, a method called “Spatial Varying Z” was tried in the remaining time.

As explained earlier, the diversity of Bicycle GAN comes from the added random z value. From this perspective, z is a very important value for image generation. Two ways to add z to the neural network are shown in figure 4.28. The left one is to add z in the input layer. The right one is to add z in each down layer in Generator (the size of z will be modified according to the size of the current layer in neural network).

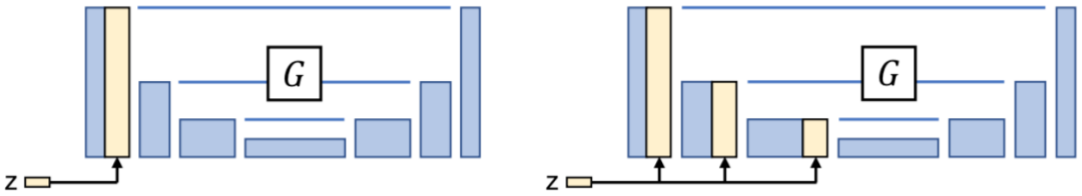

Figure 4.28 Two ways of Bicycle GAN adding z value (Zhu et al., 2017)

The spatial varying z is to find a way to modify the value of z and try to make the adjacent parts of two images have similar z value. How to change the value of z according to the space location can be seen in figure 4.29. Other ways of spatial varying z have also been tried, but only this one will be discussed.

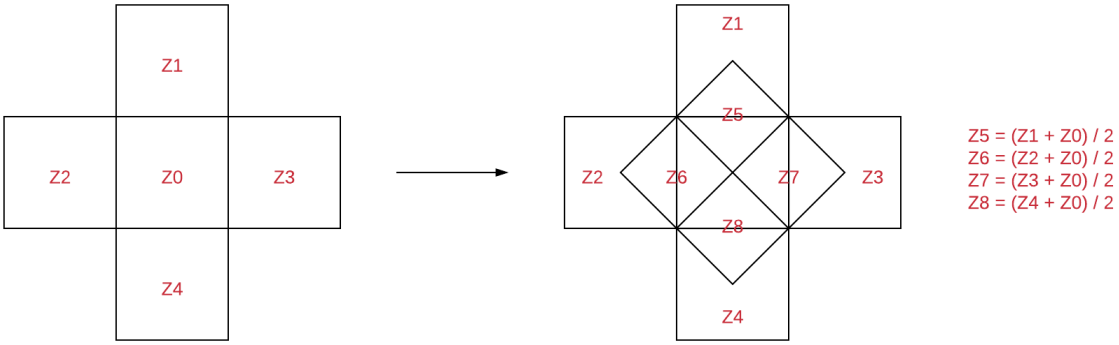

Figure 4.29 Spatial varying z value

In order to implement z with four different values, the previous neural network needs to be modified. This time z value is only added to the input layer. Then, the z value is calculated by a constant output from encoder and a random latent value. If this latent value randomly generated each time, the continuity of z value cannot be guaranteed. This means that each image must have a constant latent value. This is easy to guarantee. Because the random numbers in the computer program are all pseudo-random, the latent value can be the same every time if using image's unique number as seed for random value. In this way, all small pictures must be arranged and numbered according to their true position. When creating dataset, a new way must be used to ensure the order of images and a positioning algorithm that can find images around one image is also needed.

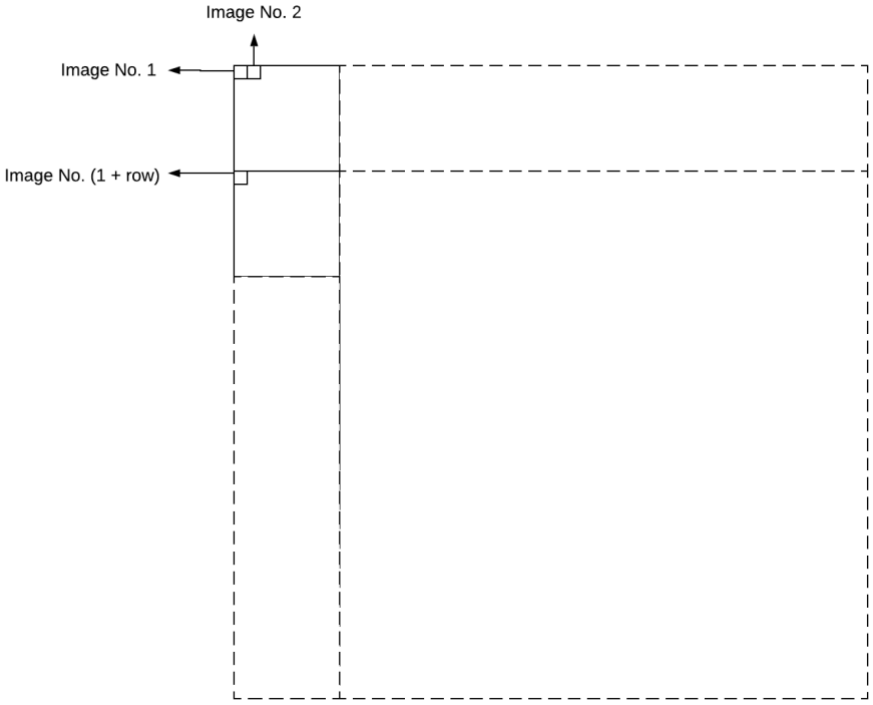

Figure 4.30 ordered numbers of small images

The new method actually requires four adjacent images to be processed at the same time for each image which means that the training time is longer than before. It turns out that this is true, this kind of training takes about 60 hours to complete 200 epochs training.

The results of this method still do not solve this problem. One example can be found in figure 4.31. It is true that the traces of the z of the triangle can be seen from the results (orange box in figure 4.31). This shows that modifying the sample z does change the generated results. But the attempts so far are still not enough to generate seamless images which can achieve unlimited combination. High-resolution image which is combined together by images generated in this way is displayed in figure 4.32. Also, the improvement is not obvious and it is still necessary to use the SE equation for comparison.

Figure 4.31 Results of spatial varying z value

Figure 4.32 Combined high-resolution image by spatial varying z value

There are still some reasonable ways to try, but unfortunately time is not enough. This is the end of experiments and investigation in this project.

Chapter 5 Evaluation

For results evaluation in this project, some significant features can be reflected by analysing image. But as mentioned in Chapter 4, some improvements cannot be intuitively represented. Results should also be evaluated by some reasonable methods. These methods may not be perfect, but they should be able to show some characteristics of the results.

5.1 Basic Generated Images Evaluation

In general, there are two metrics for evaluating the performance of a generator: whether the generated image is clear or not and whether the generated image is diverse or not. One representative methods fulfilled these two metrics is Inception Score. To be specific, enter the generated image x into Inception V3 network (Szegedy et al., 2016) and output a 1000-dimensional vector y. The vector y represents the possible category. Then the KL divergence is calculated using this probability and the distribution of the vector y. Inception Score has limitations. For example, it only considers generating samples without real data and it relies on Inception V3 network's dataset.

FID (Fréchet Inception Distance) is a better way to do this. The difference of FID is that it calculates the distance between real images and fake images at the feature level using fréchet distance (Dowson et al., 1982). So, it makes more sense. Its equation is shown below :

$$\mathrm{FID} = \|\mu_r - \mu_g\|^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}).$$

The variables in it are followed by the mean of the features in the real images, the mean of the features of the generated images, the covariance matrix of the features of the real images, and the covariance matrix of the features of the generated images. FID can measure the distance of two multivariate normal distributions. It only uses Inception V3 network as a feature extractor and does not rely on the data set used for training. But FID also has problems. The distribution of features is from a certain layer. Are these features enough to measure the distance between the distribution of real data and the distribution of generated data is still uncertain. Despite this, FID is a suitable assessment tool.

For the original image generated by Pix2Pix GAN and Bicycle GAN, FID score is used as a reference of the output quality. The results of the FID are shown in table 5.1. These results are calculated by open source code based on PyTorch which is available at https://github.com/mseitzer/pytorch-fid.

Table 5.1 Results of FID

Number	Network Type	Characteristic	Score
1	\	Real images compared to itself	-0.00004
2	Pix2Pix GAN	Trained with 1000 images dataset	154.61
3	Pix2Pix GAN	Trained with 3000 images dataset	100.63
4	Bicycle GAN	Trained with 3000 images dataset	127.20
5	Bicycle GAN	Trained with 3000 images dataset and use same sample z value for all input	151.67

From the results in the table, Bicycle GAN, which generates better results obviously, does not get a better value. This is what was said before, FID is not necessarily the embodiment of the real situation, it can only be used for reference. All in all, Bicycle GAN gets a better results in this project.

5.2 Combined Image Evaluation

There also needs a method that can analyse the quality of combined high-resolution images. Of course, FID can also be used for this, but it requires a large number of images which there is not enough. It also is not designed for seams detection, so it can't reflect whether the situation of seams has improved. A new way to evaluate the seams of images is more reasonable.

5.2.1 Pixels Distance Energy

Initially a very simple approach called pixels distance energy seemed very reasonable. It is to randomly extract a certain number of horizontal and vertical lines from the image. Then calculate the sum of the differences between every two adjacent pixels of these lines. When this kind of calculation encounters seams, the difference between the two pixels will be particularly large. This can reflect the situation of seams in the image to a certain extent.

But when this method was implemented, a problem was discovered. There are more details in the real images then generated images (shown in figure 5.1). It means that generated images' colour is more consistent. This will lead to a real image get a higher value than the fake one. So this method was abandoned.

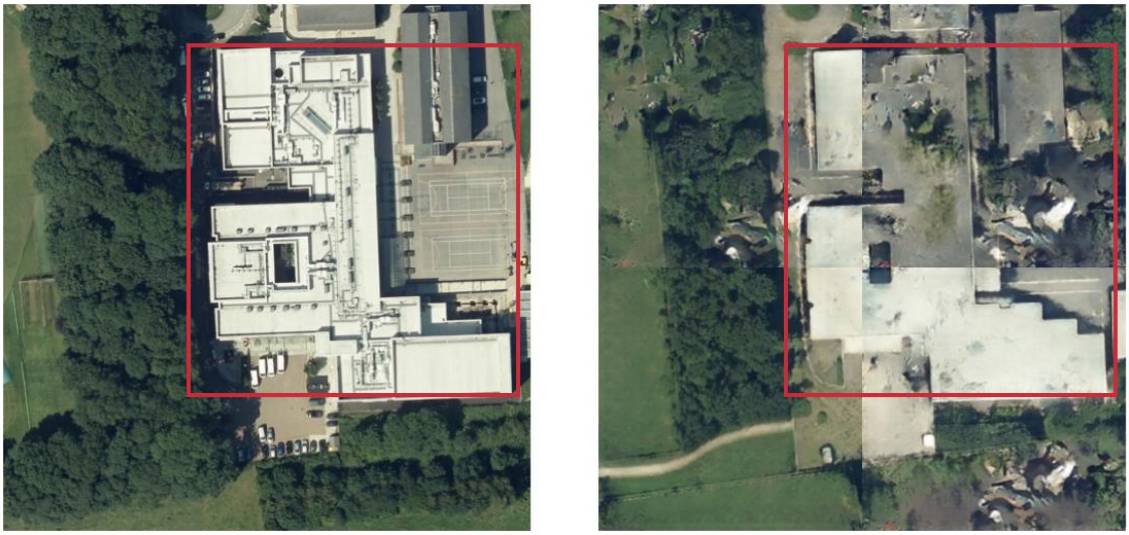

Figure 5.1 More details on the building roof in real satellite image

5.2.2 Seams Energy

Some image processing methods were tried next. This is trying to extract the seams in one image and calculate the seams to get an energy value.

The core idea is to use edge detection techniques to find the gap lines in the image. This method is carried out as follows:

a) Convert the image to a grayscale image.

b) Use Canny edge detection (Canny, 1986) function in OpenCV library to get all the edges.

c) Use Hough transform (Duda, 1972) to obtain straight lines from edges. In this process, the non-horizontal and non-vertical lines are filtered out due to the seams line are always horizontal or vertical.

d) Lines generated last step are eroded and dilated by a straight line kernel using Morphological Transformations.

e) The remaining lines can be seen as seam lines. Seams energy value can be calculated by summing the pixels of these lines. The result is the better with if one image has a lower SE value.

f) Reuse neural networks to generate images 10 times, then compose them into a high-resolution image and calculate the SE value 10 times. Average results of 10 times and get final Seams Energy value.

These steps are processed by a new written Python script. Figure 5.2 shows what does the image looks like during this process. The combined image's view after Canny edge detection is on the left and the final seam lines view is on the right. As can be seen from this figure, not

all gaps can be detected. This is because of algorithms such as Canny and Hough transform have limitations. To make it work better, a long time was spent on parameters tuning and finally the it can maximize the extraction of the seams and minimize the extraction of other lines. Although this result is not perfect, it reflects whether the splicing has improved to some extent.

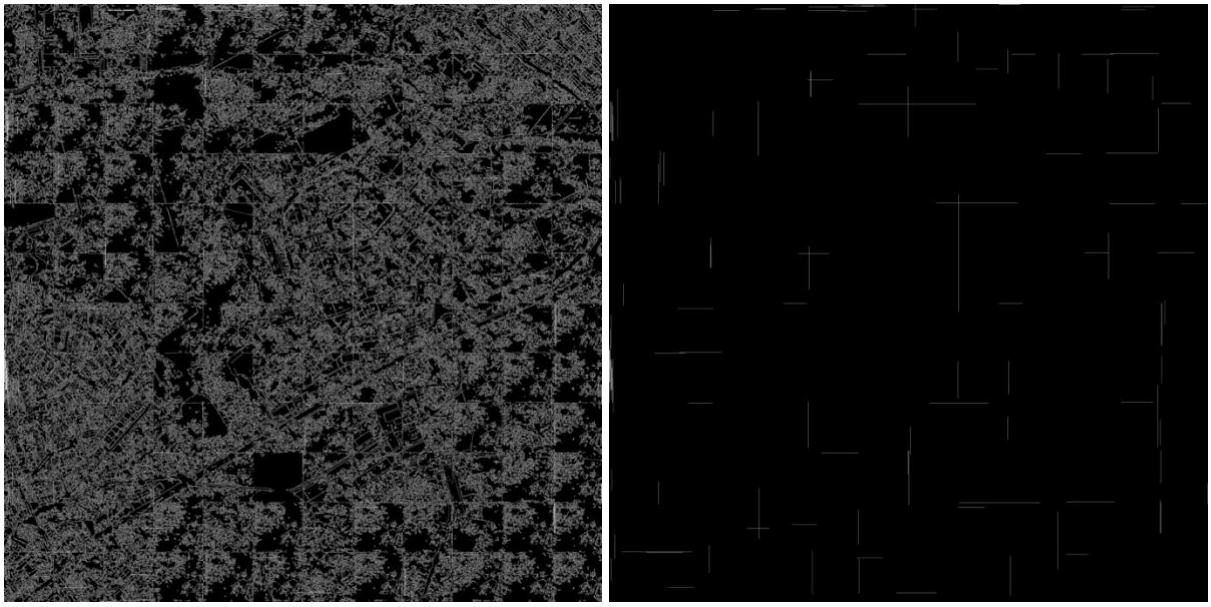

Figure 5.2 Image view during calculating SE value

The results of the Seams Energy value for some example are shown in table 5.2. In order to illustrate results better, the histogram of this table can be seen in figure 5.3.

Table 5.2 Results of Seams Energy

Number	Description	SE Value
1	Real satellite image	1418565
2	Bicycle GAN combined image	5317260
3	Bicycle GAN combined image using same z value for all input	4731372
4	Seams Bicycle GAN image (3072 input)	13153002
5	6-channel Seams Bicycle GAN image (3072 input)	10982034
6	7-channel Seams Bicycle GAN replaced image with mask 1	4890288
7	7-channel Seams Bicycle GAN replaced image with mask 3	2575908
8	7-channel Seams Bicycle GAN replaced image with mask 5	3866616
9	7-channel Seams Bicycle GAN replaced image with mask 6	4296496
10	7-channel Seams Bicycle GAN replaced image with mask 8	4547058

11	7-channel Seams Bicycle GAN replaced image with mask 10	3685617
12	7-channel Seams Bicycle GAN replaced image with mask 11	4025430
13	7-channel Seams Bicycle GAN replaced image with mask 13	3718155
14	7-channel Seams Bicycle GAN replaced image with mask 14	4167924
15	7-channel Seams Bicycle GAN replaced image with mask 1 and new loss function	3329688
16	7-channel Seams Bicycle GAN replaced image with mask 3 and new loss function	2681427
17	7-channel Seams Bicycle GAN replaced image with mask 5 and new loss function	2806734
18	7-channel Seams Bicycle GAN replaced image with mask 6 and new loss function	3027309
19	7-channel Seams Bicycle GAN replaced image with mask 8 and new loss function	3858762
20	7-channel Seams Bicycle GAN replaced image with mask 10 and new loss function	3702192
21	7-channel Seams Bicycle GAN replaced image with mask 11 and new loss function	6685149
22	7-channel Seams Bicycle GAN replaced image with mask 13 and new loss function	2614005
23	7-channel Seams Bicycle GAN replaced image with mask 14 and new loss function	2635068
24	Spatial varying Z value	4668795

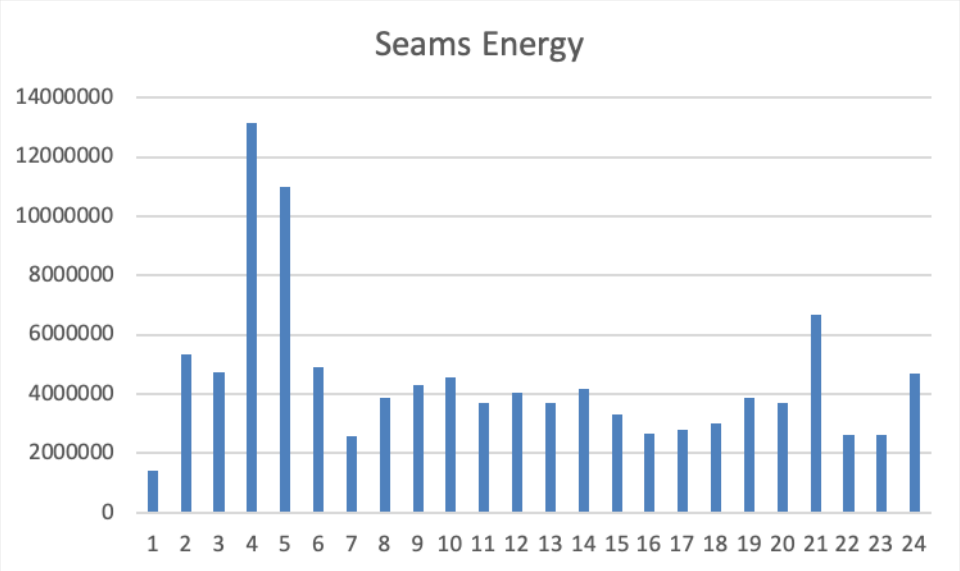

Figure 5.3 Histogram of Seams Energy value

It can be seen from the table that simply adding a mask has not improved the results. But with the combined effect of the newly added loss function and mask, some results are not bad. However, even some of these got a good score of SE, it still does not represent a good perception. The seams are still very obvious through human eyes.

More surprisingly, the two images from "a trick successful case" did not get a good score. After reviewing this operation, It has been found out that this is because the edge texture of the two images mentioned in Chapter 4 has a great influence.

The spatial varying z value method was just started and didn't get a good score.

Chapter 6 Conclusion

6.1 Conclusion

The goal of this project is to generate high quality real streets images from ordinary map images which has been reflected in the word "Deep" in the project name.

There is not a lot of time spent on making one image-to-image translation system which can process low-resolution images conversion. The whole procedure of project mainly focused on solve the question which is how to use neural networks to generated suitable low-resolution images that can be combined into high-resolution images.

But the final result did not achieve the desired result. The Seams Bicycle GAN, 6-channel Seams Bicycle GAN, adding extra mask channel, adding mask loss function or spatial varying z value both cannot generate extremely good results. The only success case is to use different resolution input in test phase and this is the reason why it is only “a trick successful case”.

Although the project was not successful in general, there are some useful conclusions in the process:

a) It is indeed possible that neural networks can be used to build image-to-image system. From map images to real streets images and from images with seams to images without seams are feasible as long as there is suitable data for training.

b) Removing noise data from the data can lead to more significant results. In this case, removing the text on map images eliminates the text in the generated image.

c) Adding additional information to the input data can correct the training results. This is why 6-channel Seams Bicycle GAN have better results than Seams Bicycle GAN.

d) The results can also be optimized by adding a reasonable custom loss function.

e) For Bicycle GAN, the value of sample z is critical. Keep trying to modify the z value may have better results.

6.2 Future Work

Due to limitations in hardware resources, project time, the difficulty of the project itself and unwilling to use techniques other than neural network, this project did not produce high-quality and high-resolution streetscapes images generation. If there is more time, this project can continue in the following directions:

a) Seams Bicycle GAN adds a random z during each iteration of training. The input image changes every iteration in this way. If the edge areas of the output image need to be kept same with the input image, it might need to modify the way z value works.

b) The way of spatial varying z value has not been attempted deeply because of the time limitation. Since the effect of z can be seen in the results, it is also possible to reach project's aim by optimizing this implementation.

c) Try to define a new loss function, this time based on the edge areas of the adjacent image. This way the neural network might generate images with seamless texture on its edge areas.

List of References

Bao, J., Chen, D., Wen, F., Li, H. and Hua, G. 2017. CVAE-GAN: fine-grained image generation through asymmetric training. *Proceedings of the IEEE International Conference on Computer Vision*. pp.2745-2754.

Bishop, C.M. 2006. *Pattern recognition and machine learning*. New York: springer.

Brownlee, J. 2016. Deep Learning & Artificial Neural Networks. 16 August. *Machine Learning Mastery*. [Online]. [14 August 2019 Accessed]. Available from: https://machinelearningmastery.com/what-is-deep-learning/

Canny, J. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*. **PAMI-8**(6). pp.679-698.

Dowson, D.C. and Landau, B.V. 1982. The Fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*. **12**(3). pp.450-455.

Duda, R.O. and Hart, P.E. 1972. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*. **15**(1). pp.11-15

Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A. 2017. Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp.1125-1134.

Frühstück, A., Alhashim, I. and Wonka, P. 2019. [Pre-print]. TileGAN: Synthesis of Large-Scale Non-Homogeneous Textures. *arXiv preprint arXiv:1904.12795*.

Goodfellow, I., Bengio, Y. and Courville, A. 2016. *Deep learning*. Cambridge: MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems.* **2**. pp.2672-2680.

He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp.770-778.

Karras, T., Laine, S. and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp.4401-4410.

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai*. **14**(2). pp.1137-1145.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. pp.1097-1105.

Kwatra, V., Schödl, A., Essa, I., Turk, G. and Bobick, A. 2003, July. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (ToG).* **22**(3). pp.277-286.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. **86**(11). pp.2278-2324.

MapBox. 2019. *Tilemill* (1.0.1). [Software]. [Accessed 25 July 2019]

Mitchell, T. 1997. *Machine Learning.* New York: McGraw-Hill Science. [Accessed 10 August 2018]. Available from: www.cs.ubbcluj.ro

QGIS Development Team. 2019. *QGIS* (3.6). [Software]. [Accessed 20 July 2019]

Ronneberger, O., Fischer, P. and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention.* pp.234-241.

Simonyan, K. and Zisserman, A. 2014. [Pre-print]. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp.2818-2826.

Tecuci, G. 2012. Artificial intelligence. *WIREs Comp Stat*. [Online]. **4**(2), pp.168-180. [Accessed 22 August 2018]. Available from: https://doi.org/10.1002/wics.200

Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O. and Shechtman, E. 2017. Toward multimodal image-to-image translation. *Advances in Neural Information Processing Systems.* pp.465-476.

Zhou, Z. 2016. *Machine Learning*. Beijing: Tsinghua University Press.

Appendix A
External Materials

a) ARC: The HPC platform in the University of Leeds.

b) Bicycle GAN official code: The official implementation of Bicycle GAN. It is available https://github.com/junyanz/BicycleGAN.git.

c) Digimap: All the training and testing data are downloaded from Digimap. Specifically, data in these catalogue is used: Ordnance Survey (map images and map GMLs) and Aerial (satellite images). These two are available at https://digimap.edina.ac.uk/os and https://digimap.edina.ac.uk/aerial.

d) FID: The FID score computation code used for evaluation is available at https://github.com/mseitzer/pytorch-fid.git.

e) Pix2Pix GAN official code: The official implementation of Pix2Pix GAN. It is available at https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.git.

f) Python library: Some Python third-party libraries are used for some specific functions. For example, NumPy for matrix calculation and OpenCV for image processing.

g) QGIS: A software for dealing with GML format geographic data.

h) TileMill: A framework for dealing with SHP format geographic data.

Appendix B
Ethical Issues Addressed

This empirical investigation project is void of any ethical issue.

Appendix C
Code

Although this project used the official implementation of Bicycle GAN, some code has been modified and some code has been added. There are also a lot of scripts for data pre-processing, dataset creation, high-resolution image combination, high-resolution image displacement and results evaluation. The new code can be found at https://github.com/Misaliet/BicycleGAN.git. The README.md file has more details on how to train or test them and it also has the instruction for to how to use the newly added scripts.

As for dataset, according to Digimap's user agreement, I am not allowed to share these images or GMLs. But the data is available from Digimap for education usage and the scripts for data pre-processing is provided in code repository.

Some of the models that have been already trained are available at https://mega.nz/#!BlgEyaRa!jZ8-9r_z2s4WRq4DynDhs3mlv4_4v6LwrqUucofUXns. This zip file is 5.13 GB.

Appendix D
Representative Samples

There are some representative samples of this project's results.

Default Bicycle GAN combined results:

Default Bicycle GAN combined results using same z value for all input:

Seams Bicycle GAN (using 3072 as input resolution):

6-channel Seams Bicycle GAN (using 3072 as input resolution):

7-channel Seams Bicycle GAN replaced image with mask 3:

7-channel Seams Bicycle GAN replaced image with mask 3 and loss function:

7-channel Seams Bicycle GAN replaced image with mask 8:

7-channel Seams Bicycle GAN replaced image with mask 8 and new loss function:

7-channel Seams Bicycle GAN replaced image with mask 13:

7-channel Seams Bicycle GAN replaced image with mask 13 and new loss function:

Spatial varying Z value: