Department of Computer Science

What is the drone looking at?

Road Tracking in Aerial Images

Student: Zhaoyu Zhang

Supervisor: Dr. Tim Morris

Degree: MEng (Hons) Computer Science

April 2021

Abstract

The purpose of this project is to find a way to self-locate outdoor drones by analyzing the aerial images sent by the drone, extracting the key information and matching the topography on the map.

Traditional methods of positioning drones rely on GPS, accelerometers, various sensors, even differential GPS to improve positioning accuracy. However, these methods are costly and unreliable. In the case of bad weather, poor signal or GPS unavailability, how to locate drones is a topic worthy of study. This project will focus on researching different methods of finding the linear structure such as roads in aerial images, and comparing their results to choose the fastest and most effective one to achieve the positioning of the drone.

This project successfully implemented two different machine learning models: per pixel segmentation and U-NET to complete the basic operation of extracting roads in aerial images. At the same time, it introduced and studied other similar methods in depth, and proved that in theory their feasibility. However, the map matching and positioning of the pictures taken after the road extraction is not the focus of this report, which only be discussed and implemented in a much simpler way without any further evaluation.

Acknowledgements

I have many people to thank for their help and support in this project. My most sincerely thanks goes to my supervisor, Dr. Tim Morris, who gave me a lot of project guidance and feedback throughout the university's third year, patiently answered all my related questions and correct my project research direction whenever I made a mistake. It can be said that the completion of my project has his indispensable contribution.

I also want to thank my family and friends for their silent support behind my back. This year is a tough year for everyone. We international students have to insist on remote learning and implement projects when the Covid-19 epidemic is spreading. All of these support and encouragement my parents and friends gave me when I felt tired and lost confidence contributes a lot to this project as well.

The training data set used for U-net model comes from the Massachusetts Roads Dataset which is uploaded by Balraj Ashwath 7 months ago, it contains 1171 aerial images of the state of Massachusetts. Each image is 1500×1500 pixels in size, covering an area of 2.25 square kilometers. The data is randomly split into a training set of 1108 images, a validation set of 14 images and a test set of 49 images. I also bring my heartfelt gratitude to him, thank him for sharing so many high-quality training data sets, and for his contribution to the final output of my model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

How to locate UAVs[1] under complex outdoor conditions is a long-term research project of many countries and institutions. According to the "Military Use of Unmanned Aerial Vehicles – A Historical Study" [14] published by the Poland faculty of National Security and the "Unmanned Aerial Vehicles: Implications for Military Operations" [9] published by the US Air Force, the future UAVs should have autonomous flight, high-efficiency integrated reconnaissance perception and precision strike capabilities[13]. In order to achieve these goals, it is an indispensable process for the development and optimization of the precise positioning and navigation system of the drone.

1.1 Motivation

The traditional method of relying on GPS to locate the drone is not reliable in most of the time. The precise positioning of the drone must have a gyroscope sensor and a magnetic field sensor to determine the flight direction, and an air pressure sensor to correct the distance of the flight by the height difference before and after it takeoff[29]. In the case of poor GPS signal or GPS unavailability, it is very necessary to conduct research on how to locate the UAV without using GPS or any other sensor.

Since the flying direction, maximum flying height, and maximum flying speed of the drone during take-off are known, we can easily create a range with the drone's take-off point as the center, whose radius is the maximum flying distance of the drone in a unit time, and then circle the possible locations of the drone on a map which found on Google Map or other map resources. By analyzing the satellite image returned by the drone, we can extract linear structures from the picture, such as the outline and shape of the road, and then compare the generated lines with the lines in the pre-built circle to determine the location of the drone.

1.2 Aim and Objectives

This project mainly studied different methods of finding the linear structure in aerial images. By comparing their processing speed, processing accuracy and matching results, finding an efficient method that is most likely to be used in this field.

[1] UAV stands for Unmanned Aerial Vehicles

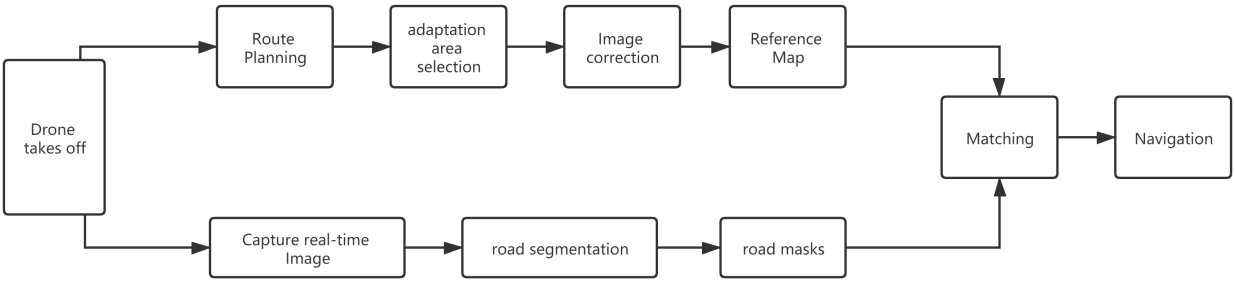

Figure 1.1: Determine drone position by matching diagrams

1.3 Project Structure

The report contains the following contents:

- An introduction to the basic theory of this project
- Background Knowledge of drone positioning and linear structure extraction from aerial images
- An overview of the road segmentation algorithms and models
- Experiments and Evaluations of the final model
- Conclusion that summarise the work done

1.4 Covid-19 Impact

Due to the outbreak of the Covid-19 in year 2020, the progress of this project has been slower than expected. As it took a lot of time to ensure self-isolation and hospitalization, a large number of tasks and deadlines had to be postponed accordingly. The time spent on this project is much less than scheduled, as well as there are remote connection and communication problems caused by the university's lockdown. Hence the project only focus on the road extraction part in front, and have a simple understanding and implementation of the image matching part in the back.

Chapter 2

Background Knowledge

2.1 Road Segmentation in Aerial Images

There are a variety of approaches to implement road segmentation in aerial images, it requires knowledge about computer vision, road database as well as image-related knowledge[4] including Gaussian blur, Neural networks, feature extracting and feature matching. Most of them are based on a few assumptions here[27][32]:

- The aerial images are taken vertically to the horizon
- Roads are elongated
- Road surfaces are usually homogeneous
- Roads have a maximum curvature
- there is adequate contrast between road and adjacent areas.

However, during the actual experiment, due to different shooting environments and photography equipment, even pictures taken at the same location may get completely different results, so the following issues have to be dealt with accordingly in order to control and minimize their influence on the experimental results.

- Roads may not be elongated at crossings, bridges
- Ground objects such as cars and pedestrians may hinder the recognition of the road
- The contrast between the road and the objects adjacent to it may not be very high, which is not conducive to effectively distinguish the road and non-road objects.
- The resolution of the input satellite image will have a significant effect on the accuracy of results of different image processing algorithms

Once these completely random and unpredictable features appear in an established automated program, they will affect the entire prediction result. In most cases, these problems are solved by manual elimination, but now there are some good ways to try to reduce their influence on the forecast outcomes.

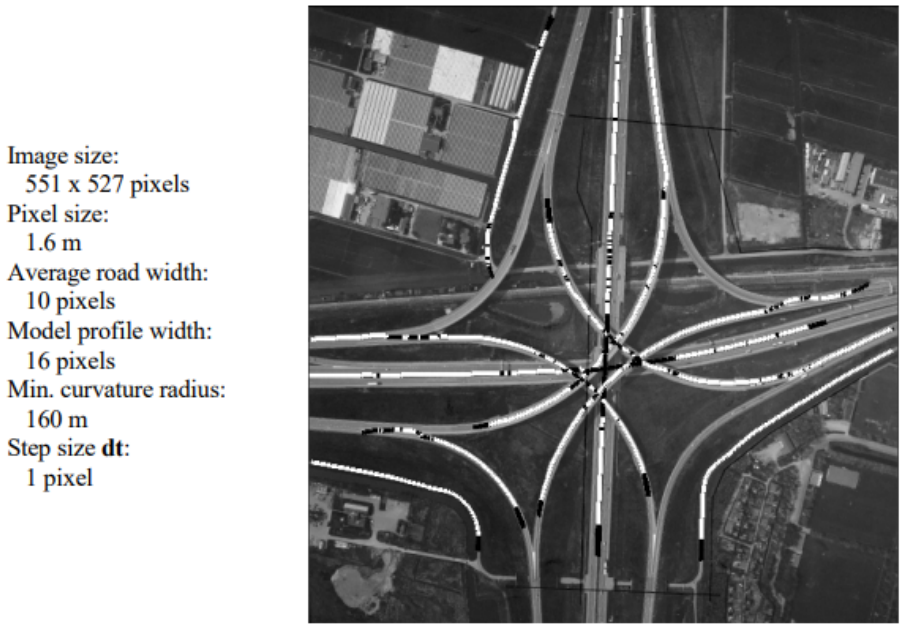

Figure 2.1: Road tracing result based on HCI, profile matching and Kalman filtering[27]

One way to solve this problem is to introduce a semi-automated program, which ensures that the aerial image is processed with the corresponding computer vision algorithm under the condition of manual assistance[21][22] to distinguish the road features. In this approach, the dynamic knowledge of human separating the roads and surrounding areas is being transferred to computers, also to guide and tell computers how to classify each pixel if they belong to a road or not when necessary.

As early as the 1990s and the beginning of the 21st century, many experts in human computer interaction (HCI) proposed the theoretical feasibility of similar methods and simply created an experimental prototype. McKeown and Denlinger[16] introduced two low-level methods, where each low-level method works independently with a human-aid model to initiate the start point, direction and road width. The first one simply established a centre line of the main road, followed by an intermediate-level operation to evaluate it by analysing the road features including surface texture and potential cars on it. The second one, which was a high-level module, combining the basic capabilities of all the models they have made before, from analyzing different road surface textures, road widths, to potentially complex road conditions, such as overpasses or vehicles on the road surface, a symbolic description and implementation were given. Also, Vosselman and de Knecht[28] found that it was difficult to create a fully automated road model in satellite imagery at the time, so they also adopted a human-computer interaction method to help the computer accelerate its image processing speed and accuracy, which initialised the mapping process by human indicating a small sector of the input satellite image to be mapped, followed by the corresponding road tracing algorithms to compute with, which will be discussed on Chapter 3 of this report. Figure 2.1 shows the result of their experiment. They also used Kalman filter in their model, which is an algorithm that uses a series of noisy measurements observed over time to produce estimates of unknown variables which in this case are the future position and shape of the road. To improve the tracking they also used spectral value matching to a reference road model by least squares.

Figure 2.2: An aerial image(left) and its binary mask(right) that labelling the road-pixels

2.2 Related works

The following section will introduce some past works on road tracing and road segmentation done by previous experts while using different techniques for outlining the development under study.

2.2.1 Massachusetts Aerial Image Labeling

As time enters the 21st century, the development of artificial intelligence and machine learning has allowed the semi-automatic road tracking model to become a fully automatic model that could not be done in the past 10 years. The premise is that in the process of training the machine, the training data used comes from a data set that has been manually resolved, so as to generate a program that can automatically distinguish road segmentation once the training is completed.

Mnih Volodymyr's PhD thesis paper[20] about labelling roads in aerial images introduced a machine learning approach that trained the model using aerial images and their corresponding binary masks, which labelled all road-pixels as 1, showing white color, and all non-road pixel as 0, showing black color. Figure 2.2 shows an simple example of this, where the data set he used contains a total of 2344 aerial images, each of them contained 1500x1500 pixels resolution.

2.2.2 Google Earth's Road Layer

As one of the largest road extraction applications, Google Earth has completed road recognition in almost all regions of the world, including various complex road conditions, such as overpasses, underground tunnels, bridges, railways, subways, and so on. When you use the web version of Google Earth, there is a hidden road layer which you can enable from the left hand side panel, which labelled all main roads on the satellite image in light yellow color.

The original Google Earth is called the "Keyhole EarthViewer". It used to be just an electronic version of the globe that can be zoomed in and rotated. Until it was acquired by Google in June 2005, it received data and financial support from the government and NASA which allows it to accelerate its development process. The mid-April 2008 version update added the

Figure 2.3: An example of with(right) and without(left) road labelling for same region on Google Earth, roads are labelled in light yellow color

new feature of ”Google View”, which included the feature of the road layer. From the Google Earth documentation, it uses both supervised and unsupervised machine learning while applying the tensorflow library for training to get the road extraction results. Their workflow of classification on road segmentation is straight-forward:

1. Collect training data. Assemble features which have a property that stores the known class label and properties storing numeric values for the predictors.
2. Instantiate a classifier. Set its parameters if necessary.
3. Train the classifier using the training data.
4. Classify an image or feature collection.
5. Estimate classification error with independent validation data.

And hence this project development uses a similar way to implement the road tracing model.

Chapter 3

Approaches and Algorithms

This chapter describes a few common road extraction methods and image matching methods, in which road extraction discusses the use of both machine learning methods non-machine learning methods. Their respective advantages and disadvantages and scope of application are also explained accordingly, and the reasons and justifications for using certain methods in this project are also given. In order to obtain better experimental results, this project uses machine learning methods to conduct experiments. Considering that machine learning requires a lot of learning costs and data materials, the corresponding popular non-machine learning methods are also reviewed.

It also give a brief description of the satellite image pre-processing function of geometric transformation, which deals with the situation when the input aerial image is not taken perpendicular to the ground–it has a z-axis deflection. However, this situation is relatively rare at present. With the technological development of the drone industry, the current drone aircraft can usually reach a flying height of 2km or more. For such a high degree of shooting effect, the z-axis deflection of the picture has a relatively smaller impact on the road extraction process, comparing to other major effects like road surface texture or potential non-road linear segments[2]. The shooting angle issue from pictures obtained through current drones and the pictures taken by satellites under the same condition can be ignored, so it will just give a simple example solution over this problem.

3.1 Image Pre-processing: Geometric Transformation

A z-axis tilted image has a certain deflection angle with the region of interest (ROI) of the subject, so the captured image must be subjected to Perspective Transformation to correct the deviation on the z-axis, and then use the Affine Transformation to rotate the image to achieve the same effect of shooting the object vertically.

Perspective transformation, also known as Projective Mapping, is to project all the points on the original picture (three dimensional objects) onto a new plane (two dimensional objects) through a matrix multiplication, while maintaining the characteristics between all points: the straight lines in the original image will still be straight lines after the transformation. It uses a 3x3 transformation matrix and 4 non-parallel points to project the quadrilateral area framed by the 4 points onto another plane of the same scale.

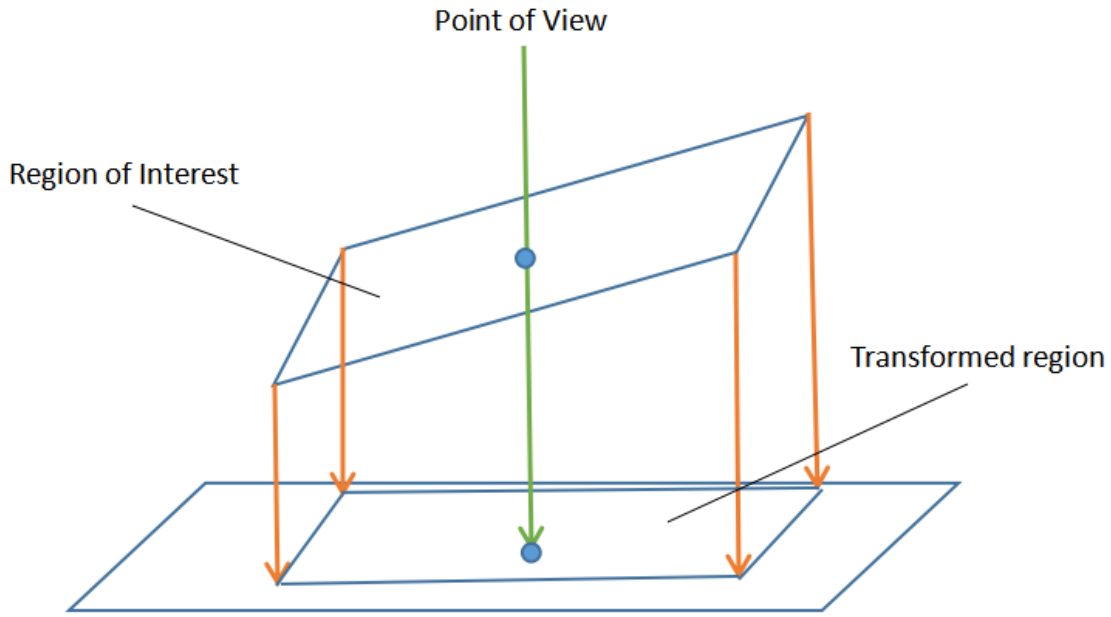

Figure 3.1: An example of visualised perspective transformation

$$[x^{'}, y^{'}, w^{'}] = [u, v, w] \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{3.1}$$

where (u, v) are the coordinates of the original points, (x, y) are the transformed coordinates, with

$$x = x^{'}/w^{'}, y = y^{'}/w^{'}$$

and w = 1 for two dimensional transformation. Basic perspective transformation operations are also implemented in OpenCV, where *cv2.getPerspectiveTransform()* gives the transformation matrix, and *cv2.warpPerspective()* gives the result of the transformed image. There are published papers analysing perspective correlation in aerial images[12], which proves the feasibility of this method.

The operation of Affine Transformation is similar to the perspective transformation, the difference is that it preserve lines and parallelism but not necessary distance and angle among points relationships. In more generous words, it is a transformation which retains the shape of the object. Also, OpenCV has implemented such approaches, which states as *cv2.getAffineTransform* and *cv2.warpAffine*.

3.2 Linear Structure Extracting: Non-Machine Learning Methods

Although non-machine learning methods of extracting roads are very limited, the results reflected in dealing with complex and unpredictable input satellite images are far less than machine learning methods. A deep understanding of these basic road information extraction algo-

rithms can help us understand more thoroughly when applying machine learning methods, and also pave the way for rare situations where machine learning is not possible.

3.2.1 Finding Contours

When the UAVs fly to a fairly high altitude, one simpler method can be used to analyze the returned aerial image, because the content displayed in the picture no longer contains complex and random interference content, such as houses, trees, cars, pedestrians, etc., on the contrary, are areas with very clear boundaries such as land and sea, small islands, large tracts of farmland, and urban areas. For this reason, one thing can do is to draw and divide the contours of different regions to generate masks that contain the dividing lines of different areas (such as coastlines dividing ocean and land), and then compare these dividing lines and contours with our reference map to achieve the same effect. The basic principles are as follows:

- **Use binary masks for better accuracy.** In aerial images containing both ocean and the land, with the boundary setting as the coastline between them, classify the different regions by color (ocean looks green and land looks yellow) or apply threshold before to make sure the regions of interest have distinct features.
- **Regions of interest are continuous.** In fact, the contour of a specific region is drawn through a list of coordinates (x, y) of boundary points of the object, the more the boundary points, the more accurate the contours drawn can get. So it is important to keep the region apart from other "noisy objects" as they would affect the points taken from the contour algorithm. At the same time, because our aerial satellite images usually have high resolution, they show all the features of a topography, including irregular protrusions and depressions of the coastline. We hope to ignore them when we divide the outline, so that the contour shape that comes out is an approximate polygon or a smooth curve, rather than the sudden high and low as in the electrocardiogram, which we can use Gaussian blur on binary masks to ignore some unnecessary details.
- **Try to eliminate unwanted lines in contours.** Once the contours have been drawn, there must be a bit of lines or segments or circles that related to other unwanted objects, so try to delete them from the contour lines in order to minimize the effect on the matching process with the reference map.

3.2.2 Orientation-Weighted Hough Transform

The Hough transform is to detect the linear structure by converting the input image into the Hough space, selecting the maximum value in the Hough space, and using these maximum values to identify the linear lines in original image space. In the normal Hough transform, the possibility that each pixel in the original image contains a linear structure is equal, so the result obtained of the Hough transform when faced with a linear structure (such as a roof) with random noise and other areas of non-region of interest will be distorted.

To minimize the false contributions, a possible solution to this[6] is to introduce a weighting strategy which assigns more weights on those pixels whose orientation O(x, y) at pixel coordinate (x, y) match a linear line segment. An orientation map is built by determining local orientations at all image positions while input image is filtered with average Gabor filters

Figure 3.2: An example of drawing contours on an aerial image

$$G(x,y) = exp(\frac{x'^2 + y'^2}{2\sigma^2})cos\frac{2\pi x'}{\lambda} \tag{3.2}$$

with

$$x' = xcos\theta + ysin\theta \tag{3.3}$$

$$y' = -xcos\theta + ysin\theta \tag{3.4}$$

where λ represents the wavelength of the cosine carrier, σ defines the scale of Gaussian envelope, and θ is the filter orientation. As the contributing weight to accumulator cell (θ, ρ) in Hough space should be larger when θ is close to o(x, y), the weight can thus be defines as

$$W_{\theta,o(x,y)} = |cos(o(x,y) - \theta)| \tag{3.5}$$

which is appropriate with a cosine value in radians as it increases when θ is closer to o(x, y). The limitations of using orientation-weighted Hough Transform is pretty straight-forward: It only detect short lines and edge segments. For an aerial image, random line segments like roofs or rivers would probably result in false maximum values in Hough space.

3.2.3 Classification

Classification is the process of identifying unknown patterns and organizing data into categories or classes with different attributes. Specifically, in image classification, the problem is to classify different image features. There are a lot of literature materials that have compiled an overview of different methods of classifying images. It is impossible to spend too many chapters to cover all the content here. Therefore, this section only briefly outlines some common methods and how they can be used to extract road segments from aerial image.

Generally, the classification method calculates the probability value for each image feature and class pair. The feature is classified as belonging to the category corresponding to its highest probability value. The categories can be specified as a *priori* (knowledge considered to be true

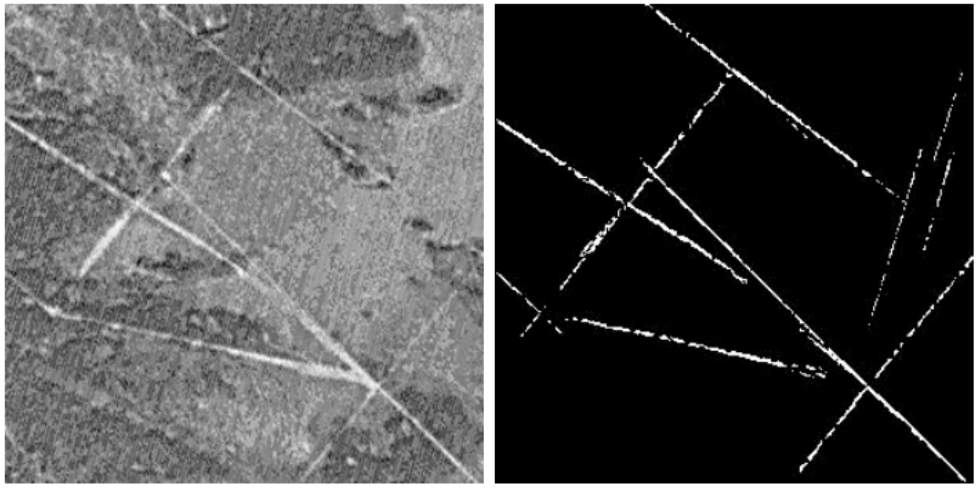

Figure 3.3: Line detection using orientation-weighted Hough Transform[6]

without being based on previous experience or observation. In this sense, a *priori* describes the knowledge that requires no proof), which is also known as supervised classification, or features can be automatically clustered, known as unsupervised classification. In supervised classification, the algorithm uses a training set to learn the main attributes of different categories. The training set is composed of image features of known category members. Instead, in unsupervised classification, the algorithm relies on clustering operations to automatically segment the data.

One of the most significant aspects of successful image classification is the construction of appropriate categories, which are discriminatory, that is, the overlap between the categories is as small as possible. It is often difficult to achieve discriminatory categories in road extraction, for example, roads can have the same color and geometric characteristics as long buildings. The most common segmentation combination in road extraction, and also the most practical one, is to classify all pixels in a picture into road pixels and non-road pixels. As for those more detailed road classification combinations, such as overpasses, tunnels, and crossroads, more cost and calculation will be consumed, and more time will be sacrificed in training and application.

A common classification method is to create a so-called feature vector v = (x_1, x_2, ..., x_n), where each element x specifies specific attributes of image characteristics, such as color, width, and curvature, followed by the probability that a certain feature belongs to a certain category is defined by the Euclidean distance between the feature and the category in the feature space. The properties used by the road extraction classifier are spectral, texture, geometric or contextual.

- **Spectral classifiers** uses the original spectral values of all the features in the aerial image, that is, the RGB values to classify them. An example work of using spectral classifiers for road extraction is [17]. They use a statistical classifier based on RGB spectrum to classify pixels, and calculate the distance between pixels and neighboring pixels by using Mahalanobis[1] and Bhattacharyya distance[2].

[1]Mahalanobis distance is used to determine whether a sample is an outlier, whether a process is in control or whether a sample is a member of a group or not.

[2]Bhattacharyya distance measures the similarity of two probability distributions.

- **Textural Classifiers** classify image features according to their texture attributes (such as structure, contrast, roughness, and orientation). Some texture classification methods used in road extraction are texture cubes [18], in which the texture feature of each pixel is recognized by establishing a rotatable rectangular window. A histogram can be obtained by calculating the mean variance of all windows, and then by analyzing the peak structure of this histogram, the road segments can be classified. For example, when a rectangular window is aligned or parallel with a road, it should have a smaller variance.
- **Geometrical Classifiers** classify image features according to their structural features. The basis of the geometric classifier in road extraction is that roads look like elongated homogeneous regions in very high-resolution images. Although satellite images taken in different places may have different road textures, this phenomenon can be improved after some basic operations such as image edge detection or segmentation algorithms. Some examples of using geometric classifiers to detect roads are [30] and [26].
- **Contextual Classifiers** uses context information to guide multiple road extraction systems. The principle is that road extraction techniques can be optimized for different types of surrounding areas. The contextual information used can be anything from nearby objects (such as cars, road signs, and traffic lights) to the color of the area. Some notable works that have used contextual classifiers are [11], where Hinz and Baumgartner used shadows, buildings and vehicles to classify roads within urban areas.

3.3 Linear Structure Extracting: Machine Learning Methods

3.3.1 Generative Adversarial Network (GAN)

Generative Adversarial Network, also known as *GAN*, is a framework for estimating generative models via an adversarial process, which simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. In the road segmentation case, the two models are responsible for training road-pixels and non-road pixels respectively. In the case where G and D are defined by multi-layer perceptrons, the entire system can be trained with backpropagation.[10]

GAN as a kind of deep learning model is inspired by the zero-sum game theory.[31] It corresponds to a generator model G and a discriminative model D that interact with each other to ensure the training process of road extraction is completely binary. So in [25][3], as the deep learning model was introduced to solve the road extraction problem, FCN (Full convolutional network) was used as the architecture of the generator, and CNN (convolutional neural network) as the discriminator. Shi Qian et al [25] used an encoder-decoder architecture in generator, and added a term of entropy loss in their loss function, while [3] chose the method that includes a two-step framework, in which two GAN models were first used to extract roads and intersections from satellite images, and then a smoothing-based graph optimization procedure was selected from among them with higher road discrimination accuracy.

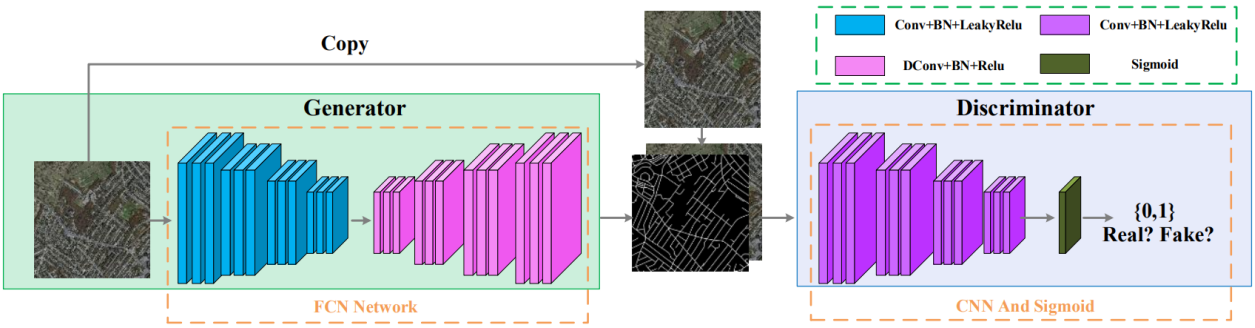

Figure 3.4: The framework of GAN using FCN and CNN[31]

3.3.2 Bayesian Line Tracking

The Bayesian line tracking process involves the construction of a posterior probability density function of the current state based on the accumulated observations. It is similar to a Bayesian Updating process: where a prediction model, a matching model and an updating model are required. Firstly, the prediction model is used to find all possible continuations of a road, and assign a probability to each predicted state. Secondly, the matching model is used to match the characteristics extracted at these predicted states to the reference characteristics. Thirdly, a Bayesian tracking updating model is used to recursively update the optimal tracking state. The following algorithms and definitions are quoted from this paper[8].

- **Prediction Model**
  The state vector at time k, denoted by X_k, which states the information relevant to a road tracking system is defined as

$$X_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{3.6}$$

  where (x_k, y_k) is the line axis position, and θ_k is the line direction.
  The state vector X_k depends on the previous state X_{k-1} and a process noise w_k according to [1]:

$$X_k = f_k(X_{k-1}, w_k) \tag{3.7}$$

  which is updated by

$$X_k = \begin{bmatrix} x_{k-1} + d\cos(\theta_{k-1} + \theta_k') \\ y_{k-1} + d\sin(\theta_{k-1} + \theta_k') \\ \theta_{k-1} + \theta_k' \end{bmatrix} \tag{3.8}$$

  where θ' is the direction shift from state X_{k-1} to X_k, and d is a constant, representing the distance covered in each tracking step.
  Each predicted state is assigned a probability $P(X_k|X_{k-1})$ that is called prediction probability. Assuming the road is straight, there is a small distance changing slowly each

update in line curvature, hence the probability density function (PDF) of the road tracking prediction can be defined as

$$P(X_k|X_{k-1}) = \frac{1}{Z}|\cos(\theta_k - \theta_{k-1})| \tag{3.9}$$

where Z is the normalization constant. Unlike ordinary single-state prediction models, a multi-state prediction model is used to give each state an appropriate weight (probability) which ensures the correct state is promised even when the line orientation changes rapidly or randomly.

- **Matching Model**

  In order to match the prediction model to the corresponding road features, an orientation map o(x, y) is defined for each pixel (x, y) as the orientation θ of the Gabor filter (see algorithm 3.2) with maximal response magnitude, and a texture map g(x, y) is defined as the maximal response magnitude. Hence the matching model contains a measurement vector including a textural and an orientation characteristics from the texture map and the orientation map

$$D_k = \begin{bmatrix} g_k \\ o_k \end{bmatrix} \tag{3.10}$$

  where g_k and o_k are the textural and orientation values at pixel (x_k, y_k). The matching model is used to match the predicted model with the corresponding features in the reference map, which is denoted by

$$D^{'} = [g^{'}, \sigma^{'}]. \tag{3.11}$$

  The matching process can be modelled as the sum of two Gaussian functions as below:

$$P(D_k|X_k) = \frac{1}{Z}(exp(\frac{(g_k - g^{'})^2}{2\sigma_g^2}) + exp(\frac{(o_k - o^{'})^2}{2\sigma_o^2})) \tag{3.12}$$

  In order to match the characteristics dynamically with the reference map stated in algorithm (3.11), it is determined by the previous state sequence $X_{k-n:k-1}$[3] for

$$\begin{bmatrix} g^{'} \\ o^{'} \end{bmatrix} = \sum_{i=1}^{n} P_{k-i} \begin{bmatrix} g_{k-i} \\ o_{k-i} \end{bmatrix} \tag{3.13}$$

  where P_{k-i} represents the weighting coefficient that is determined by the posterior probability of state X_{k-i}, and n is the number of past observations involved to estimate the reference pixel in each road tracking step.

- **Tracking Model**

  As the state sequence $X_{0:K-1}$ and the observation sequence $D_{0:k-1}$ is already known, the tracking process can be modelled as a posterior probability $P(X_k|D_{0:k})$. According to Bayes rule

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)} \tag{3.14}$$

[3] colon here represents the sequence from k-n to k-1, same meaning for all colons afterwards

the posterior probability is stated here as

$$P(X_k|D_{0:k}) = P(X_k|D_{0:k-1}, D_k) \tag{3.15}$$

$$= \frac{P(D_k|X_k, D_{0:k-1})P(X_k|D_{0:k-1})}{P(D_k|D_{0:k-1})} \tag{3.16}$$

As the observation process of state sequence D_k is independent from its previous observation $D_{0:k-1}$, $P(D_k|D_{0:k-1})$ can be seen as a normalization constant (the denominator), so we have

$$P(D_k|X_k, D_{0:k-1}) = P(D_k|X_k) \tag{3.17}$$

which is deduced from the algorithm (3.12) above. And another part of the numerator can be estimated by Chapman-Kolmogorov equation as

$$P(X_k|D_{0:k-1}) = \sum_{k=1}^{N} P(X_k|X_{k-1})P(X_{k-1}|D_{0:k-1}) \tag{3.18}$$

where N is the number of possible states at time k - 1, and $P(X_k|X_{k-1})$ can be estimated by the prediction model from algorithm (3.9). So the final tracking model of maximal posterior probability will be defined as (combination of algorithm 3.17 and 3.18)

$$P(X_k|D_{0:k}) \propto P(D_k|X_k) \times \sum_{k=1}^{N} P(X_k|X_{k-1})P(X_{k-1}|D_{0:k-1}) \tag{3.19}$$

The tracker will process all pixels in order and select those pixels with the highest probability of being a road, thereby maximizing the probability of $P(X_{0:k}|D_{0:k})$.. This tracking process can be stopped artificially by setting an appropriate threshold to avoid the entire tracker from spending too much time to achieve an unnecessary high accuracy.

3.3.3 Artificial Neural Network (ANN)

Artificial neural network is a machine learning algorithm derived from biological neural networks. It generates output by filtering and assigning weights to a series of input data. When it contains more than one hidden layer, It is also called a multi layer perceptron (MLP). It calculates values for every neuron sequentially layer by layer starting from the input layer. By summing all the previous neuron values with corresponding weights and adaptive bias, and processing with an activation function for each layer, the neuron values can be calculated as

$$y_j^l = \sigma^l \left(b^l + \sum_{i=1}^{N^{(l-1)}} w_{ij}^{(l-1)} y_i^{(l-1)} \right) \tag{3.20}$$

where

- y_j^l is the output value of neuron j in layer l
- σ^l is the activation function for layer l

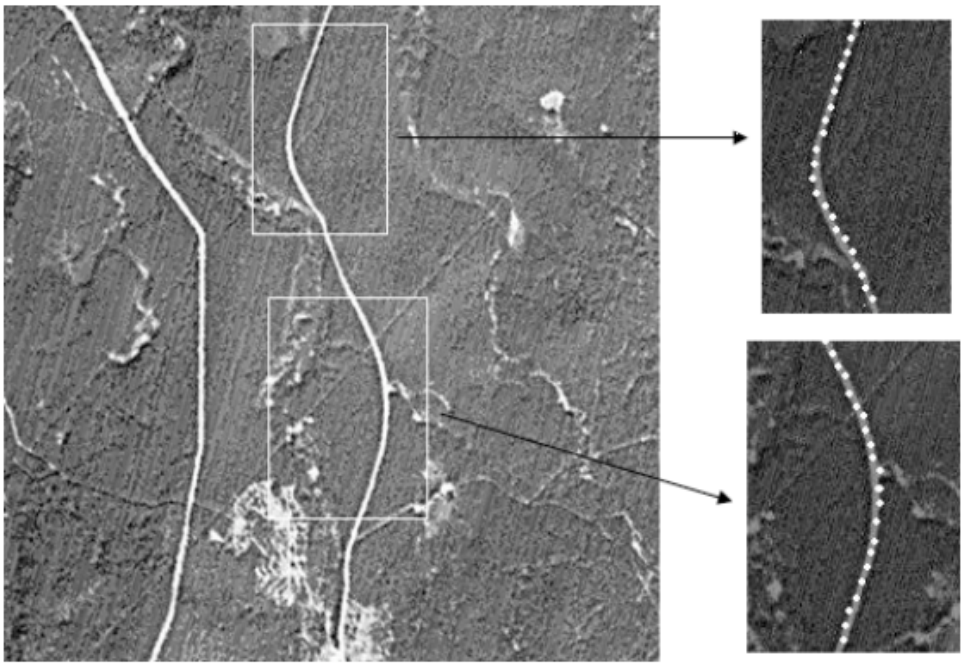

Figure 3.5: Example of aerial image with linear structures detected by Bayesian line tracker[8]

- b^l is the adaptive bias term for layer l
- N^{l-1} is the number of neurons in layer (l-1)
- $w_{ij}^{(l-1)}$ is the weight connecting neuron i in layer (l - 1) to neuron j in layer l
- $y_i^{(l-1)}$ is the output value of neuron i in layer (l - 1)

,

In order for the artificial neural network to produce the corresponding output, for each set of output and input pairs, there should be a learning algorithm to adjust the corresponding weights and biases. After the training is over, a loss function is required to be applied to the training model to compare the expected output and the actual output. Therefore, the whole process of training the model is actually continuously reducing the value of this loss function.

At the same time, in order to avoid over-fitting the training model, there must be a validation set to evaluate the performance of the neural network during the training process to prevent this situation. Under normal circumstances, training a complete neural network model will consume a lot of time, so in order to avoid the appearance of loop results, the order of application of the learning algorithm appears randomly during the training process. The entire training model will stop when there is no more improvement compared with the data in the validation set.

3.4 Topology Matching: Template Matching

Template matching is a computer vision algorithm that distinguishes the part of an image that matches to a predefined reference. The algorithm takes the following steps:

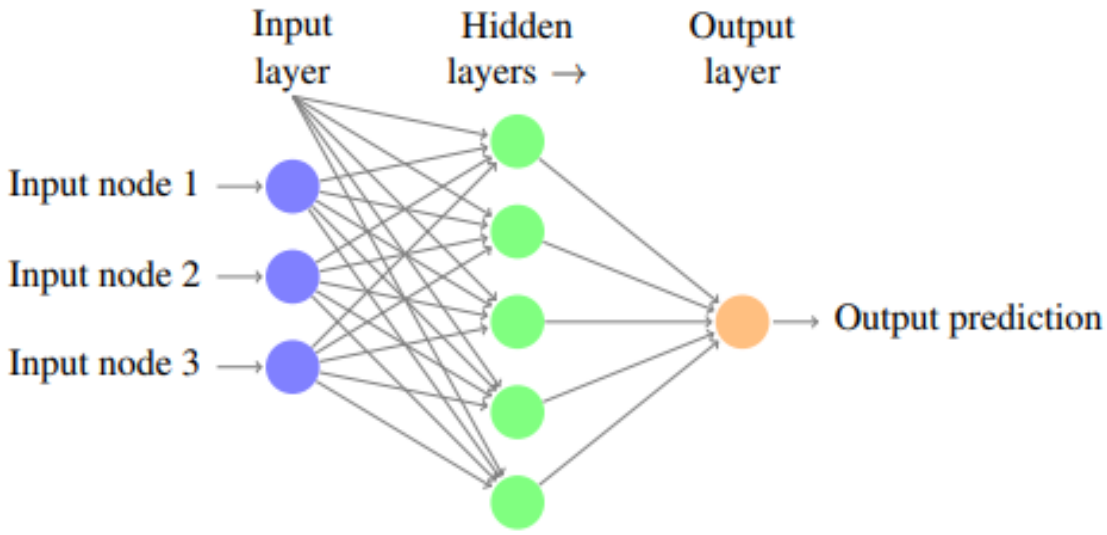

Figure 3.6: Example architecture of ANN[5]

1. A picture containing the target object is treated as a reference picture (the template) and converted into a gray scale image.
2. An image required to be matched is taken as the input and converted into a gray scale image.
3. Input image is being pre-processed, algorithms like Gaussian smooth or thresholding or noise pixel removal etc.
4. All objects in the input are marked in a bounding box and treated as different objects.
5. All objects are resized to the size of the template, and then being compared to the template in a pre-defined matrix.
6. The correlation coefficients of the template image and input objects are calculated using Pearson correlation coefficient or any other correlation algorithm.
7. The input-template pair with the highest correlation coefficient is identified as the matching object.

For a road tracking model, the input-output pair will be the road masks generated and the reference map to be compared with. One way to compare them is to

- Divide the reference map into small block areas, and perform the operation of extracting roads separately for each area (using the model that has been created).
- Compare these reference masks with our input road mask to get their correlation coefficient.
- Similarly, the input and output pair with the highest input and output is the corresponding matching image.

- Return the reference mask to the reference map and find its map location.
- Mark its approximate location on the map, so that the shooting point of the aerial image can be determined .

Chapter 4

Development

This chapter describes how to find the most efficient and practical way of extracting roads from the various methods discussed in the previous chapter, giving their example output and the corresponding reasons why they can and cannot be used, and use this as the base theory on the creation of the extracted road model. A table of implemented approaches is given as below, see table (4.1). The entire experiment process includes two machine learning methods: (Per pixel road segmentation and U-NET road segmentation). At the same time, all attempts to optimize the model, the output results and the accuracy of the final model, training errors, etc. will also be given in a visualisation form.

Step	Implementation Method
Image Pre-processing	grayscale, blurring, resize, Affine Transformation
Road Segmentation	Per-pixel Segmentation (DNN using TensorFlow and PIL) U-NET Segmentation (U-net model using Keras and PIL)
Image Matching	Template Matching

Table 4.1: Implementation methods in this project

4.1 Choosing of Methods

The first obvious method is to use the finding contour mentioned in the previous chapter to locate the location of the aerial picture. When the shooting location is high enough and the object being photographed has enough high contrast with other objects, this method can generate Group of images containing terrain contours, by comparing these generated terrain contours, it is possible to roughly locate the location where the picture was taken. Figure (4.1) shows an example of the matching point of a certain region in a reference map.

However, when the same method is applied to the image segmentation of urban roads, the results are not very satisfactory. The main reason for this is that there is no obvious boundary (for example coastlines separating sea and lane features) in the urban satellite images. It is obviously very inapplicable to distinguish roads where objects do not contain very high contrast (referring to color and shape) by simply drawing the contour shapes, it can not eliminate noisy pixels or non-road pixels from the input image. If this method needs to be applied to the actual program, the most effective way is to add a human-computer interaction platform (see Chapter

Figure 4.1: An example of template matching at high altitude

2 on Page 9) to allow humans to distinguish between road and non-road pixels, but once this approach is really implemented in the program, the entire project will lose its meaning in the first place. Therefore, in order to create a program that extracts roads fully automatically, this method is not feasible.

With regard to the advantages and disadvantages of machine learning methods against the non-learning methods, machine learning algorithms are more suitable for this specific project of road segmentation from aerial images[19][15][7], which are described below:

- **Advantages**:
  1. *There is no human intervention required*: The entire process does not involve any sort of human computer interaction, the output is fully based the inputs and training parameters.
  2. *It can improve itself continuously*: As the machine learning algorithm gains experience from the training process, they keep improving and maximize their accuracy and minimize their error rate. Generally speaking, the machine learning model will predict more accurate provided a large amount of input data.
  3. *No restriction of input data*: Machine learning algorithms can cope with input data that are non-linear, multi-dimensional or multi-variety since they update the model dynamically.
  4. *Error Tolerance*: During the training of a machine learning network like Artificial neural network, one or a few cells' corruption does not have a major impact on generating the result, so if there exist unidentifiable or unpredictable pixels in the input satellite image, the output image still can be created, with those pixels identified as non-road pixels automatically.
- **Disadvantages**:
  1. *Data set should be target-related*: The data set used for machine learning should have enough samples and have a small deviation from the expected result and be unbiased. Any errors in the samples used for training may lead to inaccurate predictions of the trained model.
  2. *Resource Dependency*: The hardware used for machine learning must have the ability to process data in parallel and be fast enough. Normally, the training of machine learning will take several hours. The specific time and prediction accuracy depend on the number and accuracy of samples.

Therefore, in this project, two machine learning methods for predicting roads were implemented. They separately used a deep neural network (DNN) and a U-net neural network to generate road masks in aerial images, which will be discussed in the next Chapter.

4.2 Implementation and Evaluation

This part mainly describes two different machine learning methods to extract roads in aerial images. While giving their corresponding training structure, principles and parameters used, their respective example output and model accuracy, model evaluation results such as the training time, training/validation loss and so on will also be given in the form of diagrams and plots.

4.2.1 Per-Pixel Segmentation

Problem:

Road segmentation is to detect roads in aerial images usually taken by satellites or UAVs. Specifically, given an aerial image, a binary mask needs to be output for the input image which shows whether each pixel in the input image belongs to a road pixel or not.

Solution:

A per-pixel classification technology is used to construct the output binary mask containing the road pixels of the input satellite image, that is, whether each pixel is independently classified as a part of the road.

In order to classify the pixel (x, y), the pixels contained in the surrounding window of the predefined side length 'L' centered on pixel (x, y) will be used as features, so the input feature vector will contain (r, g, b) values of all these pixels (including the current target pixel at (x, y)), which makes the feature vector size = 3xLxL.

As a classifier, a deep neural network (DNN) with multiple hidden layers is used. The output layer is composed of two neurons, which represent two output classes in one-key vector notation.

Implementation:

The solution mentioned above is implemented in python using the TensorFlow (version 1.13.1) machine learning algorithm and pillow (version 8.2.0). The entire implementation process contains 4 main components of python code snippets:

- Converting the input training data set into three csv text files: train.csv, test.csv and valid.csv used for training, testing and validation respectively.
- Building a classifier, training it using the generated training and testing csv text file, and saving it in a Tensorflow format. Each pixel is compared with the surrounding 24 pixels and itself in a window size of 5, hence the feature vector containing rgb values of these

pixels has the size of 75(3x5x5). The activation function is default by Rectified Linear Unit (ReLU).

- Test the trained classifier using the generated validation data set to calculate accuracy and loss.
- Load the trained classifier to generate road masks for some sample aerial images.

The first step is to convert the image dataset into a csv data file, which can then be put into the classifier for training, testing, and validation. These images are divided into training images, test images and verification images in a predefined directory structure. These directories are scanned by the first script mentioned above.

In order to obtain good results, the ratio between samples of the two categories should not be too large, and naturally, the ratio between non-road pixels and road pixels in the image is also very large, which is why it is necessary to perform filtering to balance the data. However, it must be performed in a random manner to ensure that the classifier is exposed to different sample sets for each category. In order to perform dropout for each picture, the script will load the picture and its corresponding expected output together, and then divide it into two groups of pixels (road pixels and non-road pixels), and randomly shuffle these two groups of pixels, and then for each road pixel, take two non-road pixels (ensure that the ratio between the road sample and the non-road sample in the output csv file is 1:2), generate the feature vector as described above, and write them to the output file. This is done for training images, test images and validation images to generate three files.

The second step is to train the classifier. The second script uses Tensorflow to initialize a deep neural network with four hidden layers of size (100, 150, 100, and 50) neurons, and then loads the training and test csv files, and uses it to train the neural network for scheduling number of iterations (5000 steps). The test data is used to evaluate the accuracy and loss of the model at the end of training. After the training is completed, the script saves the model on the file system in a Tensorflow-specific format. The model can be loaded to recalculate its accuracy or used to classify other input satellite images.

The third step is actually included in the source code of tensorflow. By using *tensorflow.contrib.learn.DNNClassifier.evaluate(inputfn, steps = 1)['accuracy']*, the training accuracy of the model can be obtained, while the model loss is done by using *tensorflow.contrib.learn.DNNClassifier.evaluate(inputfn, steps = 1)['loss']* command.

The fourth script is used to classify sample input images, use the input and output image names as command line parameters, load the model and input image from the "SampleInput" directory, generate a feature vector for each pixel, and use the loaded classifier to predict it , and then generate a zero/one value output image for each pixel based on the pixel's classifier prediction (road pixels are zero/white, and non-road pixels are one/black), and then save the output images to "SampleOuput" directory using the output image name entered in the command line arguments.

The evaluation of trained classifier has a **accuracy of 82.425886 %** and **loss of 40.38165 %**. As mentioned above, each feature vector generated from one single pixel contains 3xLxL values, when the data set images have very high resolution, the number of generated feature vectors will be very large (for example, a 1000x1000 image will generate 1,000,000 vectors), and training those much data on a personal computer will consume a lot of time. In order to ensure the efficiency of the training model, the number of feature vectors contained in each satellite is limited to 200,000.

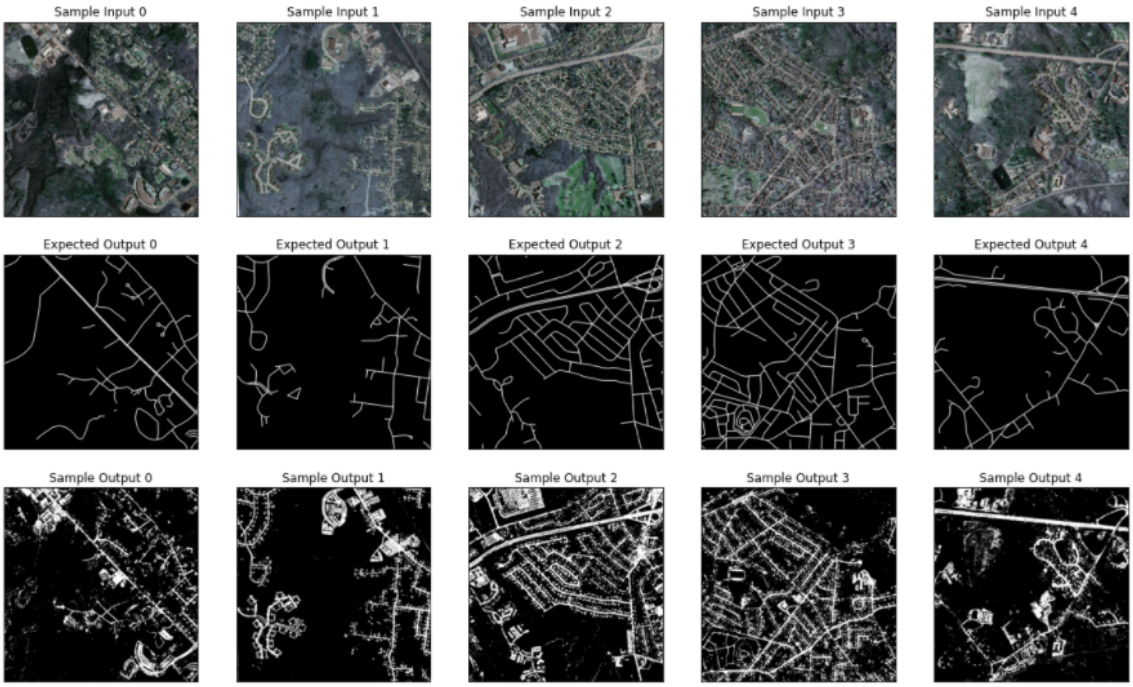

Figure 4.2: Results using per-pixel segmentation

Figure 4.3: Location computed using per-pixel segmentation (showing in rectangle window)

The way to evaluate this method is very simple: by comparing a reference map with a small part of it, using the template matching mentioned in the previous article can get the corresponding location. The picture (4.3) gives an example.

4.2.2 U-NET Segmentation

U-net is a special architecture for image segmentation purposes which is an arrangement of deep learning tools like convolutional neural networks and max pooling, it is also an encoder-decoder type network where feature maps from convolution part in down sampling step are fed to the up-convolution part in up-sampling step. It was a convolutional neural network that was first developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg.[23] It's an improvement and development of FCN: Evan Shelhamer, Jonathan Long, Trevor Darrell (2014). "Fully convolutional networks for semantic segmentation".[24]

The u-net model is actually an **Artificial Neural Network with Back Propagation** (ANN

with BP). The architecture of a u-net model (see Figure 4.5) for road segmentation is constructed by a range of machine learning algorithms, as explained below:

- The input image is resize to (256x256) to reduce the number of pixels being processed in order to improve train efficiency
- It uses a 3x3 convolutional neural network (CNN) with ReLU as its activation function and a 2x2 max pooling[1] to down-sample from the convoluted images. It also has a dropout layer which randomly sets input pixels to 0 with a frequency of rate at each step during training time that helps prevent model overfitting.
- Once the first 2 steps are done, the image will then back-propagate from a size of (16x16) to target size (256x256), while merging the 2 images from previous convolution to up-sampling[2] the image data. The up-sampling zooms in a small region of the image and eliminates the pixelation effect which often arises when a low resolution image is displayed in a relatively larger frame.
- The image after convolution and merging will be then convoluted to a binary mask with the size of (256x256x2) which has only 2 colors by applying sigmoid activation function, hence the model is complete and a output segmentation mask is generated.

The data set used for u-net model contains a total of 2216 aerial images in *.tiff* format, where half of them is training sample and rest are training labels. Hence the final model has 31,032,837 trainable parameters which takes roughly 10 hours to complete model fitting. See Appendix for data saved during the u-net model training process.

The model training results are shown in Figure (4.6). The trained model is used to extract roads from 5 random aerial images, including the original image, the expected output and actual output results are shown in the Figure (4.4).

[1] A 2x2 max pooling is the process of taking the highest value pixel in each 2x2 non-overlap region along the original image and create a new 2x2 image where each pixel is the max of a region in the original input.

[2] Up-sampling is the increase of the spatial resolution while keeping the 2D representation of an image

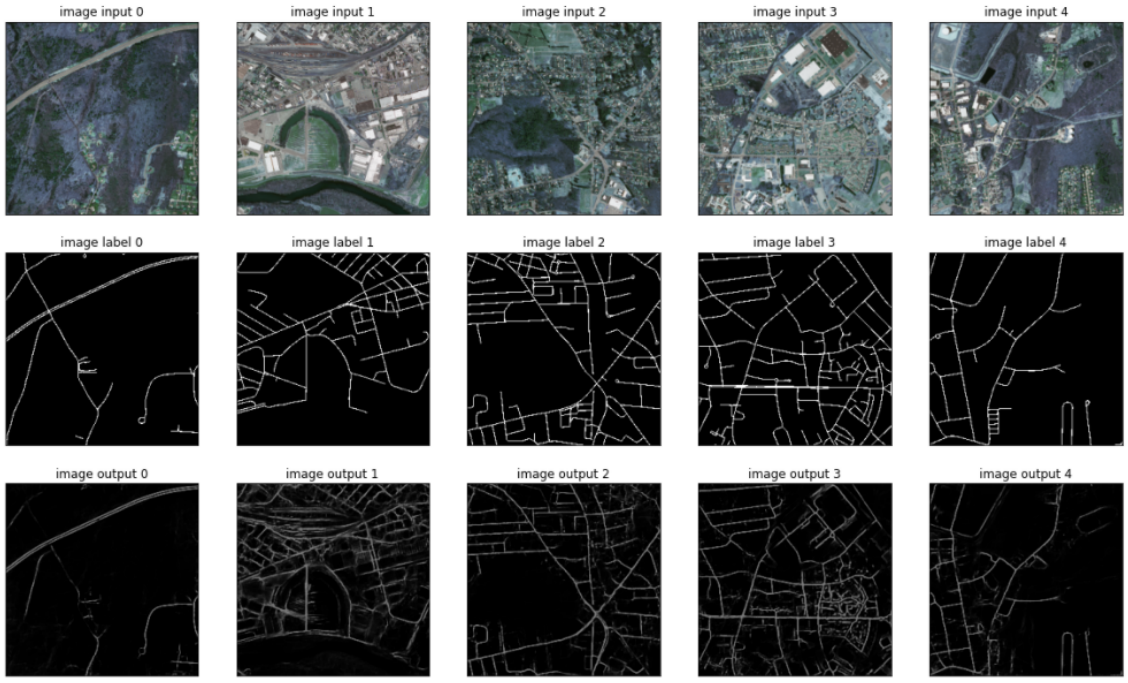

Figure 4.4: U-net model road segmentation result

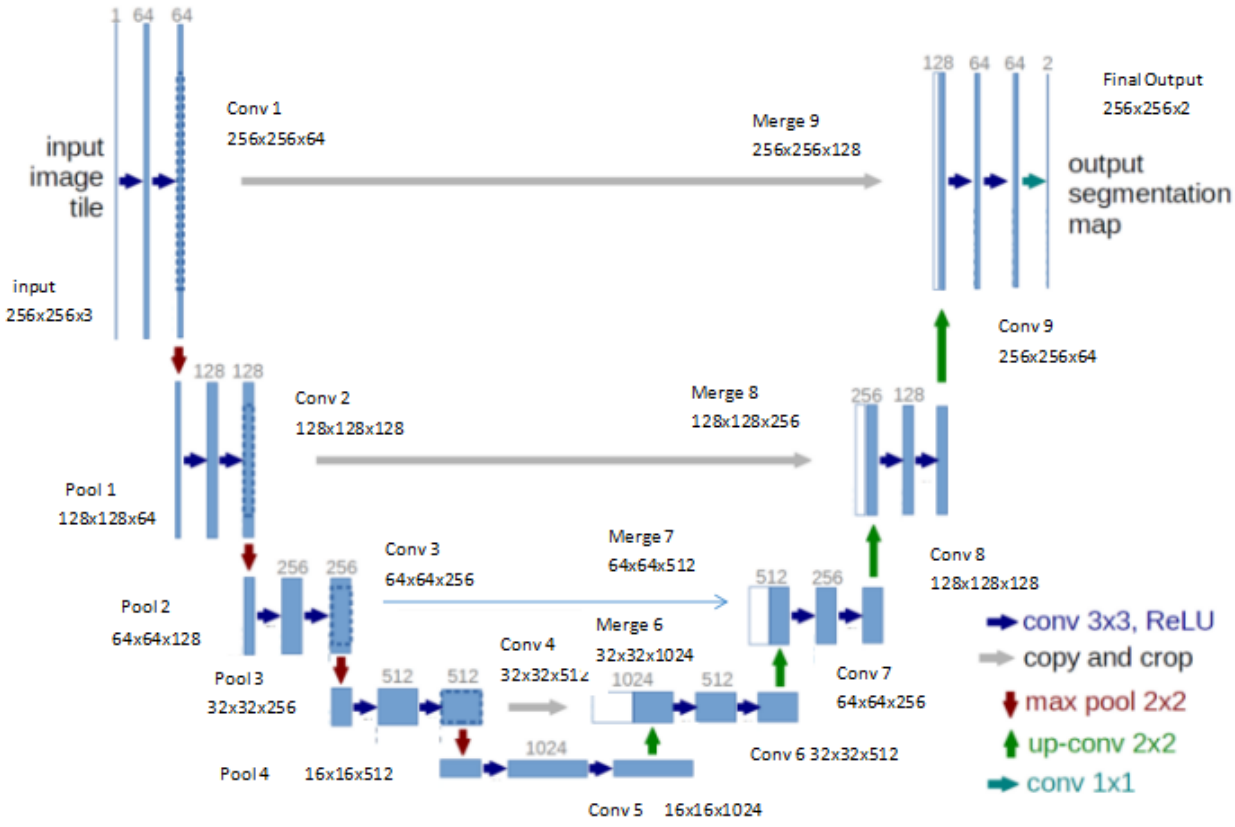

Figure 4.5: U-net architecture in this project

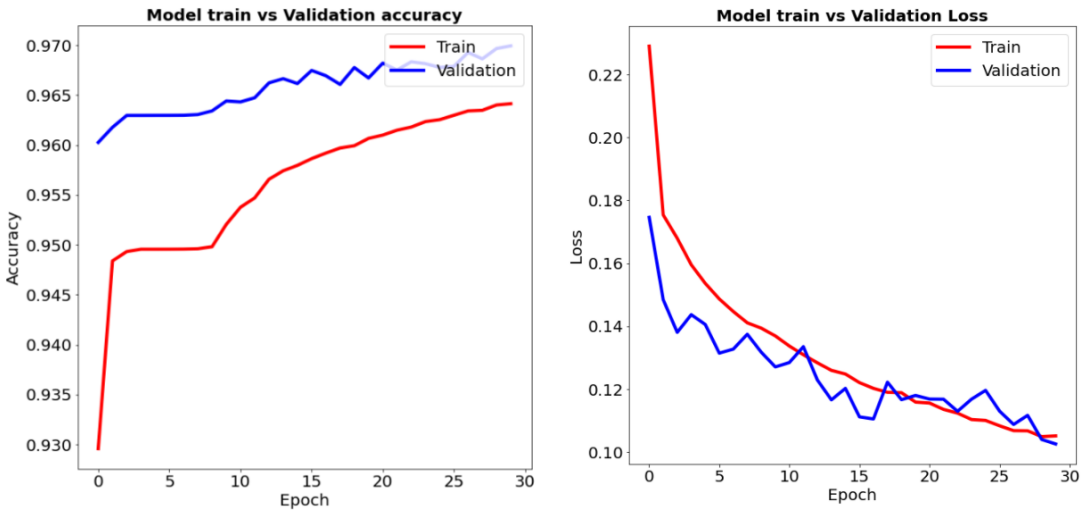

Figure 4.6: U-net model accuracy and loss in this project

Chapter 5

Conclusion

Through the comparison of two different machine learning experiments, the u-net model is superior in terms of the effect of extracting roads. The road mask it generates contains less impurities (referring to the bright non-road white pixels) and the accuracy is higher, it can identify more effective roads than per pixel segmentation, and the location of the distinguished road pixels is also more accurate.

From the saved data of the u-net model, it can be seen that when more epochs are trained, the training accuracy of the generated model is higher, and the error rate is also lower. Although u-net requires better hardware requirements and longer training time, the benefits it brings are worth the cost.

Even so, the road extracted by the u-net model has a certain gap with the expected output. According to the results displayed by the two bar graphs, if more epochs are used during training or the data set resolution used is higher, the accuracy of the trained u-net model should be closer to the expected output.

Finally, the project itself did not extract the road from the aerial image while generating the location where the aerial image was taken, but just made a simple guess with template matching. When the actual aerial image of the urban area is used to locate the drone, there are still many improvements and challenges that need to be done. I very much hope that I can have the opportunity to fill this vacancy in the near future and complete the implementation of a project for a truly automated self-positioning UAV.

Bibliography

[1] M. Arulampalam et al. “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”. In: *Signal Processing, IEEE Transactions on* 50 (Mar. 2002), pp. 174–188. DOI: 10.1109/78.978374.

[2] R. Barzaghi et al. “Computing the Deflection of the Vertical for Improving Aerial Surveys: A Comparison between EGM2008 and ITALGEO05 Estimates”. In: *Sensors* 16 (July 2016), p. 1168. DOI: 10.3390/s16081168.

[3] Dragos Costea et al. “Creating Roadmaps in Aerial Images with Generative Adversarial Networks and Smoothing-Based Optimization”. In: Oct. 2017, pp. 2100–2109. DOI: 10.1109/ICCVW.2017.246.

[4] Daniel Crevier and Richard Lepage. “Knowledge-Based Image Understanding Systems: A Survey”. In: *Computer Vision and Image Understanding* 67.2 (1997), pp. 161–185. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.1996.0520. URL: https://www.sciencedirect.com/science/article/pii/S1077314296905202.

[5] ADDI DJIKIC. “Segmentation and Depth Estimation of Urban Road Using Monocular Camera and Convolutional Neural Networks”. PhD thesis. Aug. 2018.

[6] Sahibsingh Dudani and Anthony Luk. “Locating straight-line edge segments on outdoor scenes”. In: *Pattern Recognition* 10 (Dec. 1978), pp. 145–157. DOI: 10.1016/0031-3203(78)90023-7.

[7] Data Flair. *Advantages and Disadvantages of Machine Learning Language*. Mar. 2021. URL: https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/.

[8] Rui Gao and Walter Bischof. “Bayesian Tracking of Linear Structures in Aerial Images”. In: May 2009, pp. 306–312. DOI: 10.1109/CRV.2009.8.

[9] David Glade. “Unmanned Aerial Vehicles: Implications for Military Operations”. In: (July 2000), p. 39.

[10] Ian Goodfellow et al. “Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems* 3 (June 2014). DOI: 10.1145/3422622.

[11] Stefan Hinz and Albert Baumgartner. “Automatic extraction of urban road nets from multi-view aerial imagery”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 58 (June 2003), pp. 83–98. DOI: 10.1016/S0924-2716(03)00019-4.

[12] Brayan Jaimes and Cristiano Castro. *PERSPECTIVE CORRECTION IN AERIAL IMAGES*. Dec. 2018. DOI: 10.13140/RG.2.2.34885.29926.

[13] Michael Jordan J. “Merging the Tribes: Streamlining DoD’s Acquisition of Unmanned Aerial Systems”. In: (Feb. 2006), p. 20.

[14] Cyprian Kozera. “Military Use of Unmanned Aerial Vehicles – A Historical Study”. In: *Safety & Defense* 4 (Apr. 2018), pp. 17–21. DOI: 10.37105/sd.4.

[15] Jahnavi Mahanta. *Introduction to Neural Networks, Advantages and Applications*. July 2017. URL: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207.

[16] D. McKeown and J. L. Denlinger. “Cooperative methods for road tracking in aerial imagery”. In: *Proceedings CVPR ’88: The Computer Society Conference on Computer Vision and Pattern Recognition* (1988), pp. 662–672.

[17] J.B. Mena and J.A. Malpica. “An automatic method for road extraction in rural and semi-urban areas starting from high resolution satellite imagery”. In: *Pattern Recognition Letters* 26 (July 2005), pp. 1201–1220. DOI: 10.1016/j.patrec.2004.11.005.

[18] J.B. Mena and J.A. Malpica. “An automatic method for road extraction in rural and semi-urban areas starting from high resolution satellite imagery”. In: *Pattern Recognition Letters* 26 (July 2005), pp. 1201–1220. DOI: 10.1016/j.patrec.2004.11.005.

[19] Maad M. Mijwil. *Artificial Neural Networks Advantages and Disadvantages*. Jan. 2018. URL: https://www.linkedin.com/pulse/artificial-neural-networks-advantages-disadvantages-maad-m-mijwel.

[20] Volodymyr Mnih. “Machine Learning for Aerial Image Labeling”. PhD thesis. CAN, 2013. ISBN: 9780494961841.

[21] Brad Myers, Scott Hudson, and Randy Pausch. “Past, Present, and Future of User Interface Software Tools”. In: *ACM Trans. Comput.-Hum. Interact.* 7 (Mar. 2000), pp. 3–28. DOI: 10.1145/344949.344959.

[22] Vladimir Pavlovic, Rajeev Sharma, and T Huang. “Visual Interpretation of Hand Gestures for Human-Computer Interaction A Review”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19 (Aug. 1997), pp. 677–695. DOI: 10.1109/34.598226.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: vol. 9351. Oct. 2015, pp. 234–241. ISBN: 978-3-319-24573-7. DOI: 10.1007/978-3-319-24574-4_28.

[24] Evan Shelhamer, Jonathon Long, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (May 2016), pp. 1–1. DOI: 10.1109/TPAMI.2016.2572683.

[25] Qian Shi, Liu Xiaoping, and Li Xia. “Road Detection From Remote Sensing Images by Generative Adversarial Networks”. In: *IEEE Access* PP (Nov. 2017), pp. 1–1. DOI: 10.1109/ACCESS.2017.2773142.

[26] C. Steger. “An unbiased detector of curvilinear structures”. In: *IEEE T-PAMI* 20 (Jan. 1998), pp. 311–326.

[27] George Vosselman and Jurrien de Knecht. “Road Tracing by Profile Matching and Kaiman Filtering”. In: *Automatic Extraction of Man-made Objects from Aerial and Space Images* (Jan. 1995).

[28] George Vosselman and Jurrien de Knecht. “Road Tracing by Profile Matching and Kaiman Filtering”. In: *Automatic Extraction of Man-made Objects from Aerial and Space Images* (Jan. 1995).

[29] Jean-Paul Yaacoub et al. “Security Analysis of Drones Systems: Attacks, Limitations, and Recommendations”. In: 11 (May 2020), p. 100218. DOI: 10.1016/j.iot.2020.100218.

[30] Qiaoping Zhang and Isabelle Couloigner. “Benefit of the angular texture signature for the separation of parking lots and roads on high resolution multi-spectral imagery”. In: *Pattern Recognition Letters* 27 (July 2006), pp. 937–946. DOI: 10.1016/j.patrec.2005.12.003.

[31] Xiangrong Zhang et al. “Aerial Image Road Extraction Based on an Improved Generative Adversarial Network”. In: *Remote Sensing* 11 (Apr. 2019), p. 930. DOI: 10.3390/rs11080930.

[32] Jun Zhou, Walter Bischof, and Terry Caelli. “Road tracking in aerial images based on human–computer interaction and Bayesian filtering”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 61 (Nov. 2006), pp. 108–124. DOI: 10.1016/j.isprsjprs.2006.09.002.

Appendix A

U-net training process

Model: ”model”

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 256, 3)	0	
conv2d (Conv2D)	(None, 256, 256, 64)	1792	input_1[0][0]
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36928	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 128, 128, 128)	73856	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 128)	147584	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 64, 64, 256)	295168	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None, 64, 64, 256)	590080	conv2d_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 256)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 32, 32, 512)	1180160	max_pooling2d_2[0][0]
conv2d_7 (Conv2D)	(None, 32, 32, 512)	2359808	conv2d_6[0][0]
dropout (Dropout)	(None, 32, 32, 512)	0	conv2d_7[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 512)	0	dropout[0][0]
conv2d_8 (Conv2D)	(None, 16, 16, 1024)	4719616	max_pooling2d_3[0][0]

conv2d_9 (Conv2D) (None, 16, 16, 1024) 9438208 conv2d_8[0][0]

==========

dropout_1 (Dropout) (None, 16, 16, 1024) 0 conv2d_9[0][0]

==========

up_sampling2d (UpSampling2D) (None, 32, 32, 1024) 0 dropout_1[0][0]

==========

conv2d_10 (Conv2D) (None, 32, 32, 512) 2097664 up_sampling2d[0][0]

==========

concatenate (Concatenate) (None, 32, 32, 1024) 0 dropout[0][0]
conv2d_10[0][0]

==========

conv2d_11 (Conv2D) (None, 32, 32, 512) 4719104 concatenate[0][0]

==========

conv2d_12 (Conv2D) (None, 32, 32, 512) 2359808 conv2d_11[0][0]

==========

up_sampling2d_1 (UpSampling2D) (None, 64, 64, 512) 0 conv2d_12[0][0]

==========

conv2d_13 (Conv2D) (None, 64, 64, 256) 524544 up_sampling2d_1[0][0]

==========

concatenate_1 (Concatenate) (None, 64, 64, 512) 0 conv2d_5[0][0]
conv2d_13[0][0]

==========

conv2d_14 (Conv2D) (None, 64, 64, 256) 1179904 concatenate_1[0][0]

==========

conv2d_15 (Conv2D) (None, 64, 64, 256) 590080 conv2d_14[0][0]

==========

up_sampling2d_2 (UpSampling2D) (None, 128, 128, 256) 0 conv2d_15[0][0]

==========

conv2d_16 (Conv2D) (None, 128, 128, 128 131200 up_sampling2d_2[0][0]

==========

concatenate_2 (Concatenate) (None, 128, 128, 256) 0 conv2d_3[0][0]
conv2d_16[0][0]

==========

conv2d_17 (Conv2D) (None, 128, 128, 128) 295040 concatenate_2[0][0]

==========

conv2d_18 (Conv2D) (None, 128, 128, 128) 147584 conv2d_17[0][0]

==========

up_sampling2d_3 (UpSampling2D) (None, 256, 256, 128) 0 conv2d_18[0][0]

==========

conv2d_19 (Conv2D) (None, 256, 256, 64) 32832 up_sampling2d_3[0][0]

==========

concatenate_3 (Concatenate) (None, 256, 256, 128) 0 conv2d_1[0][0]
conv2d_19[0][0]

==========

conv2d_20 (Conv2D) (None, 256, 256, 64) 73792 concatenate_3[0][0]

==========

conv2d_21 (Conv2D) (None, 256, 256, 64) 36928 conv2d_20[0][0]

```
==========================================================================================
conv2d_22 (Conv2D)              (None, 256, 256, 2) 1154        conv2d_21[0][0]
==========================================================================================
conv2d_23 (Conv2D)              (None, 256, 256, 1) 3           conv2d_22[0][0]
==========================================================================================
Total params: 31,032,837
Trainable params: 31,032,837
Non-trainable params: 0
==========================================================================================
```

A.1 U-net Model fitting

```
******************************
Fitting model...
******************************
Epoch 1/30
56/56 - 1489s - loss: 0.2057 - accuracy: 0.9496 - val_loss: 0.1581 - val_accuracy: 0.9630
Epoch 2/30
56/56 - 1491s - loss: 0.1739 - accuracy: 0.9496 - val_loss: 0.1510 - val_accuracy: 0.9630
Epoch 3/30
56/56 - 1493s - loss: 0.1662 - accuracy: 0.9496 - val_loss: 0.1489 - val_accuracy: 0.9630
Epoch 4/30
56/56 - 1498s - loss: 0.1606 - accuracy: 0.9496 - val_loss: 0.1326 - val_accuracy: 0.9630
Epoch 5/30
56/56 - 1489s - loss: 0.1541 - accuracy: 0.9496 - val_loss: 0.1459 - val_accuracy: 0.9630
Epoch 6/30
56/56 - 1491s - loss: 0.1496 - accuracy: 0.9496 - val_loss: 0.1297 - val_accuracy: 0.9630
Epoch 7/30
56/56 - 1490s - loss: 0.1463 - accuracy: 0.9496 - val_loss: 0.1324 - val_accuracy: 0.9630
Epoch 8/30
56/56 - 1488s - loss: 0.1432 - accuracy: 0.9496 - val_loss: 0.1387 - val_accuracy: 0.9630
Epoch 9/30
56/56 - 1492s - loss: 0.1410 - accuracy: 0.9496 - val_loss: 0.1285 - val_accuracy: 0.9630
Epoch 10/30
56/56 - 1489s - loss: 0.1382 - accuracy: 0.9509 - val_loss: 0.1305 - val_accuracy: 0.9655
Epoch 11/30
56/56 - 1490s - loss: 0.1361 - accuracy: 0.9550 - val_loss: 0.1263 - val_accuracy: 0.9654
Epoch 12/30
56/56 - 1489s - loss: 0.1332 - accuracy: 0.9557 - val_loss: 0.1323 - val_accuracy: 0.9655
Epoch 13/30
56/56 - 1492s - loss: 0.1312 - accuracy: 0.9563 - val_loss: 0.1178 - val_accuracy: 0.9660
Epoch 14/30
56/56 - 1490s - loss: 0.1297 - accuracy: 0.9568 - val_loss: 0.1207 - val_accuracy: 0.9656
Epoch 15/30
56/56 - 1489s - loss: 0.1279 - accuracy: 0.9573 - val_loss: 0.1214 - val_accuracy: 0.9665
Epoch 16/30
56/56 - 1489s - loss: 0.1261 - accuracy: 0.9579 - val_loss: 0.1264 - val_accuracy: 0.9652
Epoch 17/30
56/56 - 1487s - loss: 0.1272 - accuracy: 0.9580 - val_loss: 0.1142 - val_accuracy: 0.9669
Epoch 18/30
56/56 - 1486s - loss: 0.1232 - accuracy: 0.9589 - val_loss: 0.1198 - val_accuracy: 0.9670
Epoch 19/30
56/56 - 1487s - loss: 0.1242 - accuracy: 0.9590 - val_loss: 0.1157 - val_accuracy: 0.9668
Epoch 20/30
56/56 - 1491s - loss: 0.1213 - accuracy: 0.9598 - val_loss: 0.1135 - val_accuracy: 0.9681
Epoch 21/30
56/56 - 1489s - loss: 0.1197 - accuracy: 0.9602 - val_loss: 0.1100 - val_accuracy: 0.9683
```

Epoch 22/30
56/56 - 1487s - loss: 0.1180 - accuracy: 0.9607 - val_loss: 0.1115 - val_accuracy: 0.9680
Epoch 23/30
56/56 - 1486s - loss: 0.1167 - accuracy: 0.9611 - val_loss: 0.1187 - val_accuracy: 0.9673
Epoch 24/30
56/56 - 1484s - loss: 0.1162 - accuracy: 0.9614 - val_loss: 0.1168 - val_accuracy: 0.9680
Epoch 25/30
56/56 - 1484s - loss: 0.1148 - accuracy: 0.9618 - val_loss: 0.1239 - val_accuracy: 0.9673
Epoch 26/30
56/56 - 1486s - loss: 0.1139 - accuracy: 0.9621 - val_loss: 0.1099 - val_accuracy: 0.9687
Epoch 27/30
56/56 - 1689s - loss: 0.1128 - accuracy: 0.9625 - val_loss: 0.1112 - val_accuracy: 0.9678
Epoch 28/30
56/56 - 2025s - loss: 0.1115 - accuracy: 0.9629 - val_loss: 0.1171 - val_accuracy: 0.9685
Epoch 29/30
56/56 - 2080s - loss: 0.1113 - accuracy: 0.9631 - val_loss: 0.1185 - val_accuracy: 0.9678
Epoch 30/30
56/56 - 1868s - loss: 0.1100 - accuracy: 0.9634 - val_loss: 0.1213 - val_accuracy: 0.9670