Aircraft Detection using Single Shot Scale-invariant Face Detector with Viridis Saliency Map from Remote Sensing Image

Mr. Teepakorn Lilek

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Applied Mathematics and Computational Science
Department of Mathematics and Computer Science
FACULTY OF SCIENCE
Chulalongkorn University
Academic Year 2019

การตรวจจับอากาศยานโดยใช้ตัวตรวจจับใบหน้าแบบชอทเชิงเดี่ยวที่ไม่แปรเปลี่ยนตามสเกลกับแผนภาพเด่นชัดแบบวิริดิสจากภาพรับรู้ระยะไกล

นายทีปกร ลิเหล็ก

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

Thesis Title	Aircraft Detection using Single Shot Scale-invariant Face Detector with Viridis Saliency Map from Remote Sensing Image
By	Mr. Teepakorn Lilek
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Associate Professor NAGUL COOHAROJANANONE, Ph.D.

Accepted by the FACULTY OF SCIENCE, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

..... Dean of the FACULTY OF SCIENCE
(Professor POLKIT SANGVANICH, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Associate Professor KHAMRON MEKCHAY, Ph.D.)

..... Thesis Advisor
(Associate Professor NAGUL COOHAROJANANONE, Ph.D.)

..... Examiner
(Associate Professor RAJALIDA LIPIKORN, Ph.D.)

..... Examiner
(Assistant Professor KRUNG SINAPIROMSARAN, Ph.D.)

..... External Examiner
(Suriya Natsupakpong, Ph.D.)

ทีปกร ลิเหล็ก : การตรวจจับอากาศยานโดยใช้ตัวตรวจจับใบหน้าแบบชอทเชิงเดี่ยวที่ไม่แปรเปลี่ยนตามสเกลกับแผนภาพเด่นชัดแบบวิริดิสจากภาพรับรู้ระยะไกล. (Aircraft Detection using Single Shot Scale-invariant Face Detector with Viridis Saliency Map from Remote Sensing Image) อ.ที่ปรึกษาหลัก : รศ. ดร.นกุล คูหะโรจนานนท์

การตรวจจับอากาศยานจากภาพรับรู้ระยะไกลเป็นงานที่มีประโยชน์และสำคัญในหลายด้าน ยกตัวอย่างเช่น ในด้านการทหารระบบตรวจจับอากาศยานถูกนำมาใช้เพื่อค้นหาและตรวจสอบเครื่องบินของศัตรูจากภาพรับรู้ระยะไกล เป็นต้น ปัจจุบันการตรวจจับอากาศยานจากภาพรับรู้ระยะไกลได้รับความสนใจมากขึ้น เนื่องจากงานนี้เป็นงานที่มีความท้าทายเป็นอย่างมาก งานนี้ทำได้ยากเนื่องด้วยมีปัญหามากมายเกิดขึ้นในภาพรับรู้ระยะไกล ยกตัวอย่างเช่น เงา พื้นหลังที่ซับซ้อน และขนาดที่ต่างกันของเครื่องบิน เป็นต้น ในงานวิจัยนี้เราสร้างแผนภาพเด่นชัดแบบวิริดิสโดยใช้เทคนิคต่าง ๆ จากการประมวลผลภาพ ยกตัวอย่างเช่น การส่งแบบสี การประมวลผลภาพทางสัณฐาน และการทำขีดแบ่งภาพ เป็นต้น ซึ่งแผนภาพเด่นชัดแบบวิริดิสสามารถช่วยลดสิ่งรบกวนจากพื้นหลังของภาพรับรู้ระยะไกลได้เป็นอย่างดี ยิ่งไปกว่านั้นเราได้รวมแผนภาพเด่นชัดแบบวิริดิสของเราเข้ากับเครื่องมือตรวจจับที่มีชื่อเสียงมีชื่อว่าตัวตรวจจับใบหน้าแบบชอทเชิงเดี่ยวที่ไม่แปรเปลี่ยนตามสเกล ซึ่งถูกนำมาใช้ในการตรวจจับใบหน้ามนุษย์ในภาพ จากผลการทดลองพบว่าวิธีการที่นำเสนอนี้สามารถรับมือกับปัญหาทางด้านขนาดได้อย่างมีประสิทธิภาพ

สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคณนา	ลายมือชื่อนิสิต .....
ปีการศึกษา	2562	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6071942523 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORD: aircraft detection, remote sensing image, Viridis saliency map, image processing, Single Shot Scale-invariant Face Detector

Teepakorn Lilek : Aircraft Detection using Single Shot Scale-invariant Face Detector with Viridis Saliency Map from Remote Sensing Image. Advisor: Assoc. Prof. NAGUL COOHAROJANANONE, Ph.D.

Aircraft detection from remote sensing images is a useful task and essential in many fields; for example, in the military, aircraft detection systems are used to search and inspect an enemy aircraft from remote sensing images. Nowadays, aircraft detection from remote sensing images has gained increasing research attention because this task is very challenging. This task is difficult due to many occurred problems in remote sensing images, such as a shadow, a complicated background, and various aircraft scales. In this research, we create a Viridis saliency map using many techniques from image processing such as color mapping, morphological image processing, and image thresholding. The Viridis saliency map can help reduce the noise from the background of remote sensing images well. Moreover, we combine our Viridis saliency map with a famous detector called Single Shot Scale-invariant Face Detector, which is used to detect a human face in the image. From the results, this method can handle the scale problem efficiently.

Field of Study:	Applied Mathematics and Computational Science	Student's Signature .....
Academic Year:	2019	Advisor's Signature .....

ACKNOWLEDGEMENTS

I would like to offer my special thanks to my thesis advisor, Associate Professor Nagul Cooharojananone, for his valuable and creative suggestions during the development of this research. This thesis would not have been possible without his assistance and the many opportunities that he gave.

I would also like to thank the full scholarship from the Development and Promotion of Science and Technology Talent Project (DPST) for financial support in my education.

Moreover, I would like to express my very great appreciation to the Department of Mathematics and Computer Science Program and my friends who always assist and inspire me.

Teepakorn Lilek

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

Introduction

Remote sensing is the process that gathers information about the object, area, and phenomenon without making physical contact using a remote sensing device called a remote sensor. We can collect information on our target by gauging the nature of the reflectance of light energy, and the received information will be used for analysis and interpretation based on the user's intentions.

Nowadays, remote sensing technologies have been developed rapidly and are used in numerous fields, such as agriculture, geography, economic, commercial, and military. Frequently, remote sensing technologies use the information from aerial images obtained from a drone or an airplane and satellite images to analyze and apply with social works such as weather forecasting, land surveying, urban planning, object classification, and object detection.

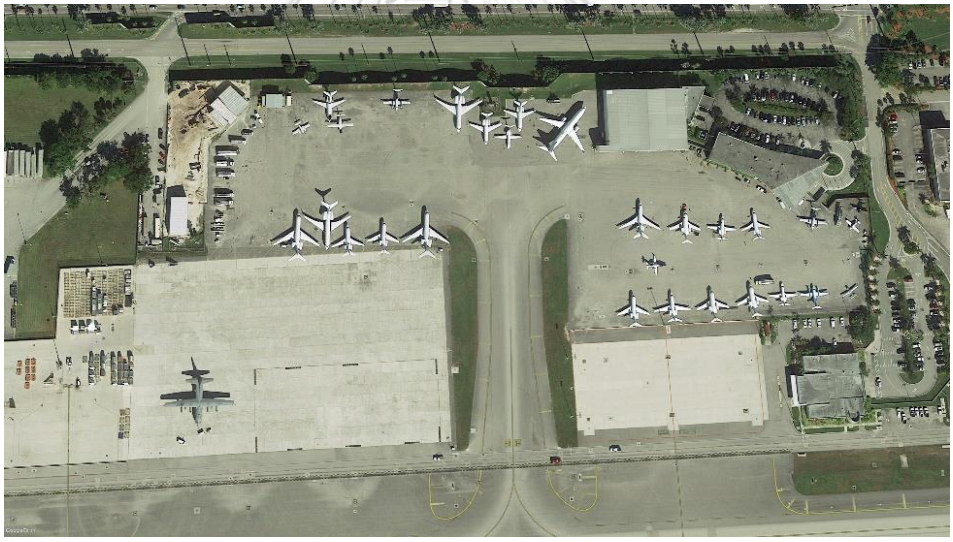
Object detection is a process of finding the location of objects and classify the class of objects at the same time. Object detection is used to apply in many ways; for example, in traffic, we use the object detection system to measure a traffic density by detecting the number of vehicles on the road and in the military field, we use the object detection system to detect an aircraft from remote sensing image for searching and inspecting the enemies' aircraft.

Aircraft detection from remote sensing images is one of the most exciting works that use the information from aerial images and satellite images to identify the aircraft's location in the image. Anywise, this task is very challenging due to many factors such as the various rotations of the aircraft, the complexity of background, the various scales of the aircraft, and the shadow that occurred around the aircraft. Figure 1 shows examples of problems that affect the detection of aircraft from remote sensing images.

(a)

(b)

(c)



(d)

Figure 1 (a) The aircraft rotation problem, (b) the background complexity problem, (c) the aircraft scale problem, and (d) the aircraft shadow problem

In recent years, deep learning has become influential in many computer works. Deep Learning uses multiple convolutional layers to extract higher-level features from the input data so that way, deep learning can deal with the complex problems well compared to conventional detection methods. Many researchers

apply deep learning to detect the aircraft from remote sensing images because it computes very fast and gives high accuracy. For example, in 2013, Wu et al. [1] proposed the aircraft detection method based on binarized normed gradients (BING) to provide candidate regions of aircraft. Then used the convolutional neural network (CNN) model to determine which region contains the aircraft. In 2014, Chen et al. [2] proposed the aircraft detection framework based on gradient thresholding images and deep convolutional neural networks (DNN). This work used image processing techniques, such as gradient filtering and image thresholding, to extract the input image features and then used the DNN to identify the aircraft's locations from these features. In 2017, Han et al. [3] used a region locating network (RLN) to find the candidate regions which contain the aircraft from large-scale remote sensing images. Then Faster R-CNN is used to detect the aircraft's locations from these regions. These methods work well and give a good result, but it still has some problems; for example, the dense small aircraft cannot be identified, and the objects that have characteristics similar to the aircraft are detected as aircraft.

In 2017, Single shot scale-invariant face detector (S^3FD) is an interesting deep learning face detection method that can handle the different scales of faces and can reduce an incorrect prediction very well. However, this method is always confused by the object that looks like a human face, such as an animal face and a toy face.

In this work, we propose a fusion between a Viridis saliency map and a single shot scale-invariant face detector that can reduce much noise from background, solve the problem of aircraft's scale well, reduce the confusion between aircraft and aircraft-like, and detect the aircraft from remote sensing images efficiently.

1.1 Objective

To propose an algorithm that can detect the aircraft from remote sensing images effectively.

1.2 Scopes and Assumptions

1. The remote sensing images used in this work need a height between 0.8 and 1.5 km from a ground level.
2. All aircraft must park on the ground.
3. The remote sensing images used in this work are gathered from Google Earth.
4. The remote sensing images used in this work are RGB color images with the .jpg format files.

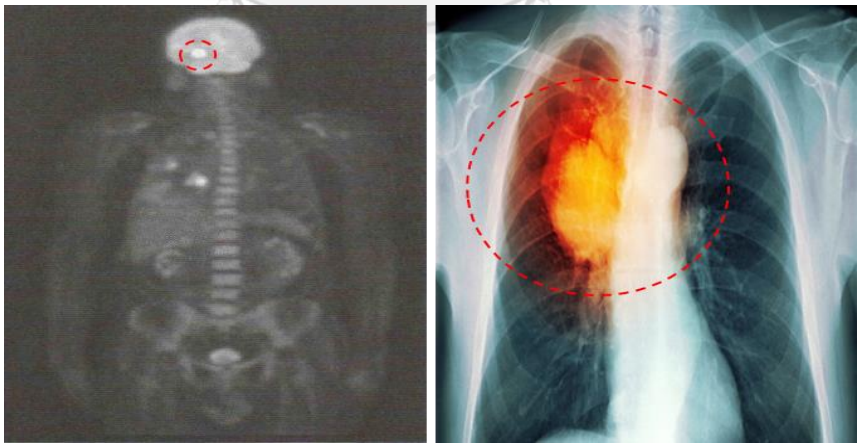
CHAPTER II

Background Knowledge

2.1 Image Processing

Image processing is a set of techniques and algorithms for analyzing, enhancing, and transforming a digital image by using a digital computer to process. The image processing technique is used to apply with various fields such as medicine, biology, astronomy, and engineering.

Image processing is an essential technique in the medical field and is used to process with many types of images, for example, gamma-ray and x-ray images. Figure 2(a) shows an example image of a brain scan by using gamma-ray imaging and using a technique from image processing to enhance and obtain a tumor in the brain. Figure 2(b) shows the example image of a lung scan by using x-ray imaging and using a technique from image processing to enhance the cancer visibility in the lung.



(a) **(b)**

Figure 2 (a) Tumor detection using gamma-ray imaging and (b) lung cancer detection using x-ray imaging

Source: Adapted from [4] and [5]

2.1.1 Digital Image Definitions

A digital image is denoted as a two-dimensional function $f(x, y)$, where x and y are discrete coordinates, and the value of f at any coordinates (x, y) is called gray level or intensity. The image $f(x, y)$ is an array (matrix) that contains M rows and N columns, and the intersection between a row and a column is called a pixel. The integer value is assigned for the discrete coordinate: $x = 0,1,2, \dots, M-1$ and $y = 0,1,2, \dots, N-1$. Thus, the image $f(x, y)$ with $M \times N$ array can be written as

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}.$$

Figure 3 shows the example image and its intensity array where 0, 0.5, 1 represent black, gray, and white, respectively.

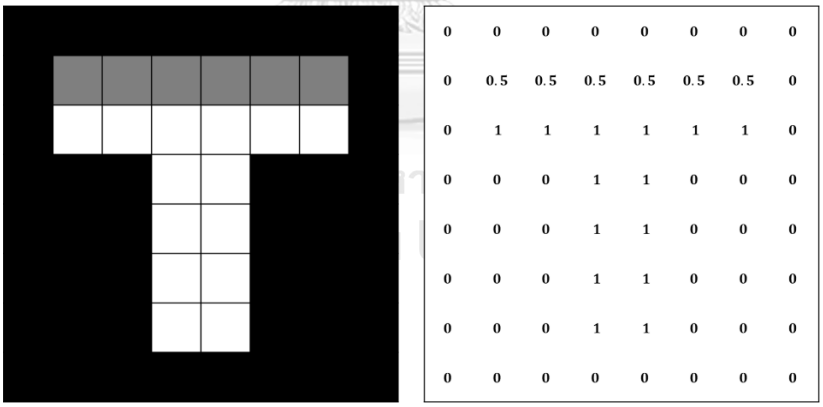


Figure 3 The example of a grayscale image (left) and its corresponding intensity array (right)

2.1.2 Basic Mathematical Tools Used in Image Processing

For image processing tasks, mathematical tools are very useful and essential because they can help in many ways, such as image enhancement, noise reduction, and feature extraction.

2.1.2.1 Elementwise Operations

An elementwise operation between one or two images is used to perform on a pixel by pixel basis. For example, consider the following 3x3 array of images:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \text{ and } \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

The elementwise product ($\odot$) between these two examples is defined by this formula:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}.$$

2.1.2.2 Arithmetic Operations

Arithmetic operations, such as addition, subtraction, multiplication, and division, are a kind of elementwise operation often used to apply with the image tasks. For example, subtraction between two images can be used to detect the differences. Arithmetic operations between two images of size $M \times N$, $f(x,y)$ and $g(x,y)$, are denoted as

Addition: $A(x,y) = f(x,y) + g(x,y)$,

Subtraction: $S(x,y) = f(x,y) - g(x,y)$,

Multiplication: $M(x,y) = f(x,y) \times g(x,y)$,

Division: $D(x,y) = f(x,y) \div g(x,y)$,

where $A(x,y)$, $S(x,y)$, $M(x,y)$, and $D(x,y)$ are also images of size $M \times N$.

2.1.2.3 Logical Operations

Logical operations, such as AND, OR, and NOT, are often used to combine two binary images, where the binary image is the image that consists of two colors: black (0) and white (1). For integer images, the logical operation is applied in an elementwise (bitwise) way. Table 1 shows an example of a truth table for AND (∧), OR (∨), and NOT (~) operators. The example results of using AND, OR, and NOT operators with a binary image are shown in Figure 4.

Table 1 The example of a truth table for AND, OR, and NOT operators

a	b	a AND b	a OR b	NOT(a)
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

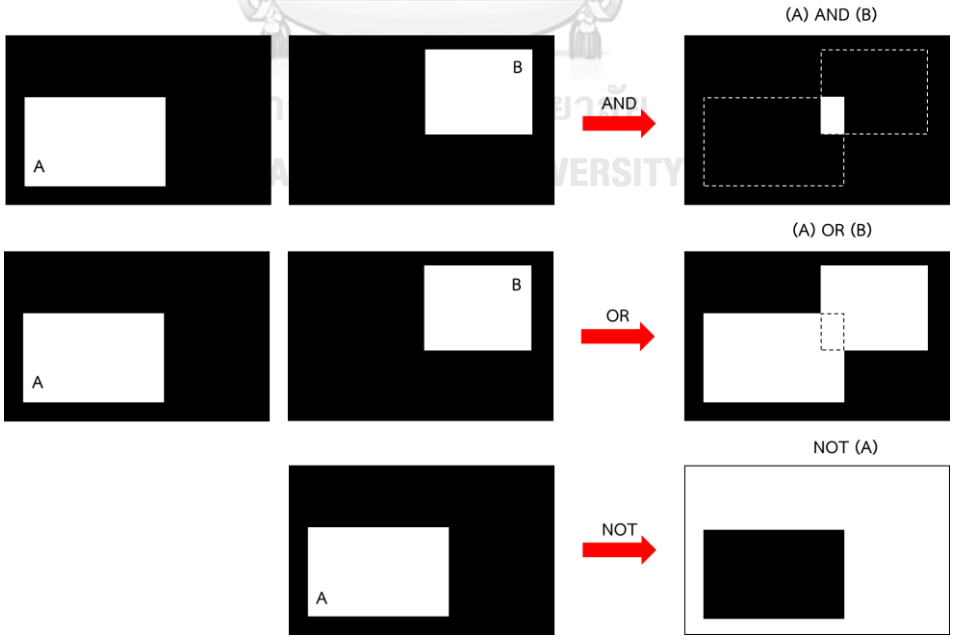


Figure 4 The visualization of logical operators (AND, OR, and NOT)

2.1.3 Spatial Filtering

Spatial filtering is the technique that uses to process or transform a spatial domain of an input image into the desired output. There are many kinds of a filter such as negative filter, log transformation filter, and gaussian filter.

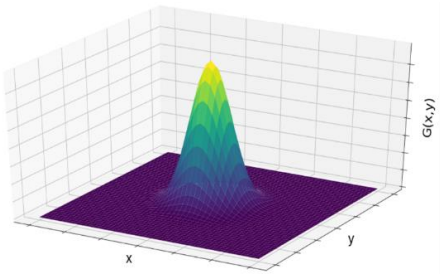
Gaussian filter is one of the popular filters that use to blur an image to avoid the noise. For example, many types of research, such as medicine, engineering, and military use the gaussian filter to adjust and enhance an image before using it with the primary model.

2.1.3.1. Gaussian Filters

The Gaussian filter is the spatial filter that uses to blur and reduce noise in images. It generates a window called kernel (see in section 2.6.1), and the coefficient of this Gaussian kernel is defined by this equation:

$$G(x,y) = Ke^{-\frac{x^2+y^2}{2\sigma^2}},$$

where K is the amplitude and σ is the standard deviation. Figure 5(a) shows a graphical representation of the Gaussian distribution with $K = 1$, and $\sigma = 1$, and Figure 5(b) shows the example of a 3x3 Gaussian kernel.



(a)

$\frac{1}{4.8976} \times$

0.3679	0.6065	0.3679
0.6065	1	0.6065
0.3679	0.6065	0.3679

(b)

Figure 5 (a) The graphical representation of Gaussian distribution with K=1, and σ=1, and (b) a 3x3 Gaussian kernel

2.1.4 Color Image Processing

Color is one of the essential descriptors of image processing. Color is often used for classifying the objects and assigning a class of each pixel in the image. Many kinds of research used color descriptors for achieving their goals; for example, in engineering, color is used to detect a defect on the circuit board, and in medicine, color is used to inform the difference between a bad cell and good cell.

2.1.4.1 Color Models

A color model is a mathematical model describing how colors can be represented as tuples of numbers. There are many kinds of color models, such as the RGB color model and the grayscale model.

2.1.4.1.1 The RGB Color Model

The RGB color model (also called RGB color space or RGB color system) produces all colors by mixing between three primary colors: red (R), green (G), and blue (B). This model is based on the Cartesian coordinate system, and the color subspace is the cube shown in Figure 6(a). This model's color values are points within this cube, which are defined by vectors extending from the origin, and all values are normalized to be in the range [0,1]. For each pixel of RGB color image, store triplet values of (R, G, B) and have a depth of 24 bits where the number of bits will scale the range of color values in the image. The example of 24-bit RGB color is shown in Figure 6(b).

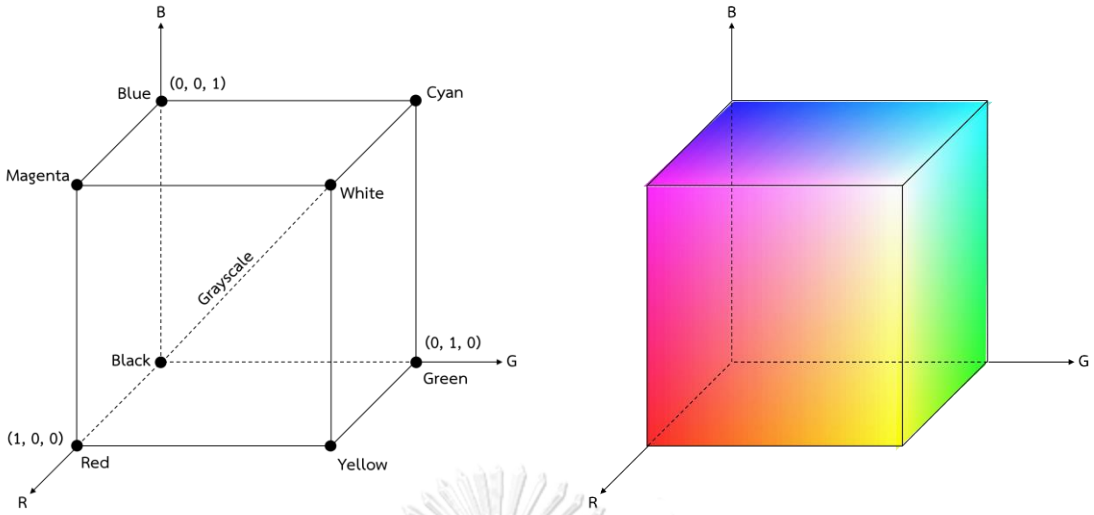


Figure 6 (a) The color subspace and (b) the example of 24-bit RGB color

2.1.4.1.2 The Grayscale Model

Grayscale is a range of shades of the gray color where the intensity is stored as an 8-bit integer giving 256 different shades of gray. An example of grayscale shade is shown in Figure 7.

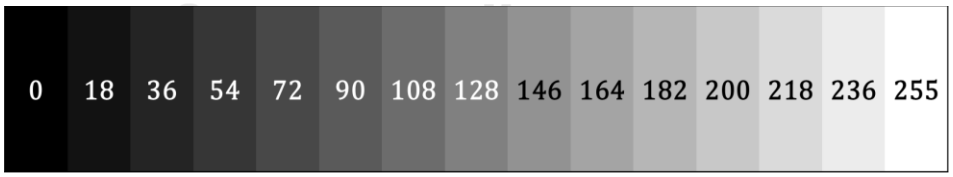


Figure 7 The example of grayscale

To transform the RGB color image to the grayscale color image, there are many methods available, but the most popular method which is widely used among the researcher is the following equation:

$$Gray(x,y) = 0.299 \cdot R(x,y) + 0.587 \cdot G(x,y) + 0.114 \cdot B(x,y),$$

where $R(x,y)$, $G(x,y)$, and $B(x,y)$ is the intensity of the red, green, and blue channels of RGB color image at any pixels (x,y), respectively.

2.1.4.2 Color Mapping

Color mapping is a process that maps the colors of the source image to the other colors. Color mapping plays a significant role in data visualization as it can enhance the effectiveness and efficiency of data and provide more insights into data. Nevertheless, choosing a poor color mapping will make the insights of data obscure and decrease the algorithm's efficiency. One of the best colormaps is Viridis colormap, which was developed in 2015 by Stéfan van der Walt and Nathaniel Smith [6].

The color scales in Viridis Colormap were designed to be:

1. Colorful, spanning a palette as broad as possible to make it easy to observe. The color scales of Viridis colormap is shown in Figure 8.

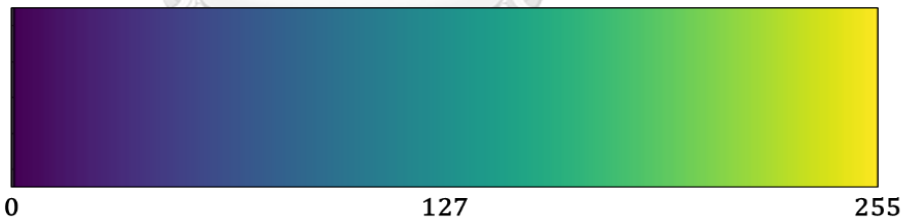


Figure 8 The Viridis color scales

2. Perceptually uniform, meaning that the closed color values will provide a similar color appearance, and the color values that are far away from each other will provide a more different color appearance consistently.
3. Robust to color blindness, making the color blindness more easily distinguish colors.

Figure 9 shows the evaluation of Viridis colormap. We can observe that whether there is a change in the color scale of Viridis colormap, the perceptual derivatives and lightness derivatives still provide very close values.

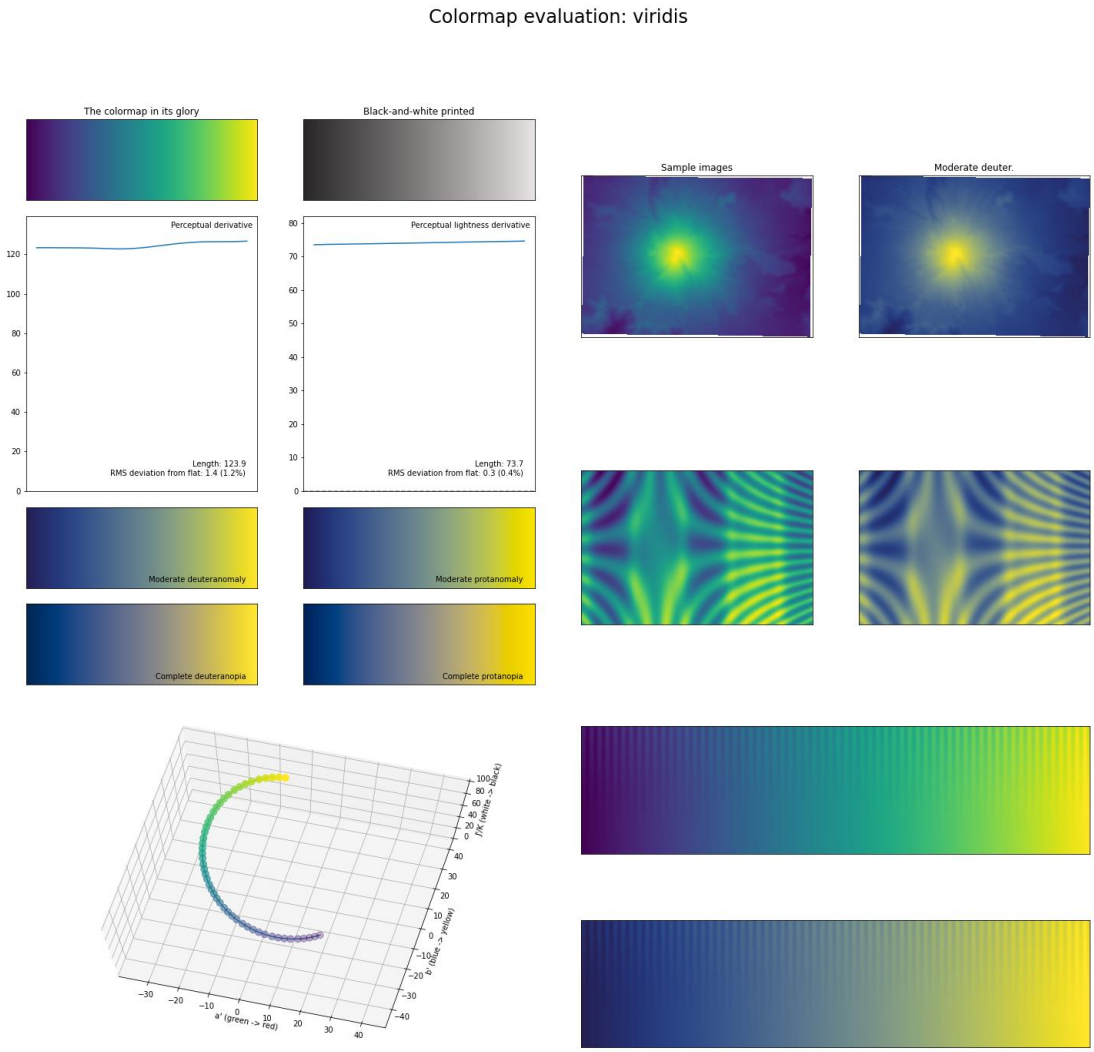


Figure 9 The evaluation of Viridis colormap

2.1.5 Morphological Image Processing

Morphological operations in image processing, such as erosion and dilation, are a set of non-linear operations that process images based on shapes. Morphological operation is a useful technique used to adjust each pixel and improve the shape of objects in the image. A small template called the structuring element is applied to process all possible locations in the image by comparing it with the corresponding neighborhood pixels.

Besides, the concept of set reflection and translation is widely used in morphological operations. The reflection of a structuring element B denoted by $\hat{B}$ is defined as

$$\hat{B} = \{w | w = -b, for\ b \in B\},$$

where $\hat{B}$ is the set of points in B whose coordinates have been reversed.

The translation of a structuring element B by point z denoted by $(B)_z$ is defined as

$$(B)_z = \{c | c = b + z, for\ b \in B\},$$

where $(B)_z$ is the set of points in B whose coordinates have been shifted by point z.

2.1.5.1 Erosion and Dilation

Erosion and Dilation are useful techniques that have been widely used in many image processing tasks. These techniques are used to reduce noise from the background and used to join some parts of the separated objects.

2.1.5.1.1 Erosion

Erosion is the technique used to decrease the thickness and remove small anomalies of objects in the binary image by subtracting objects which are smaller

than the structuring element. Given A is a binary input image and B is a structuring element, then erosion of A and B denoted as $A \ominus B$, is defined as

$$A \ominus B = \{z | B_z \subseteq A\},$$

where B_z is the translation of a structuring element B by point z.

Figure 10 shows the example of erosion using different structuring elements, while Figure 11 shows the result of using erosion to remove image components with a square structuring element of size 7×7 whose components are all 1s.

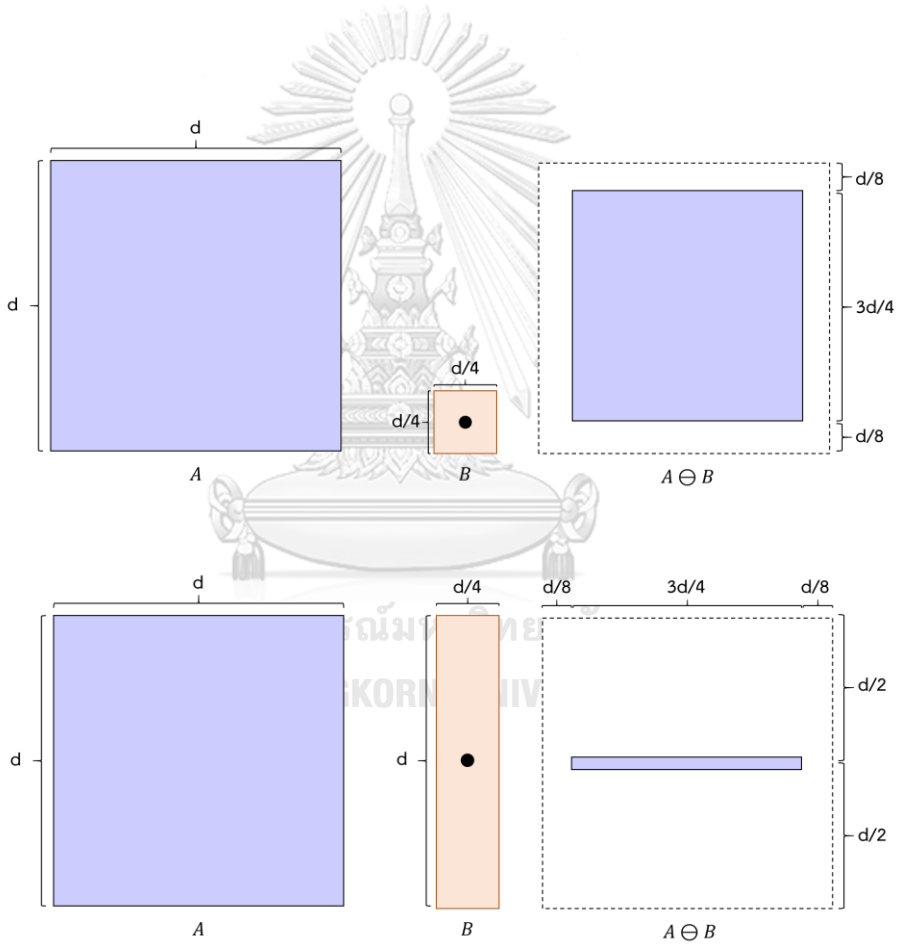


Figure 10 The example of erosion using different structuring elements

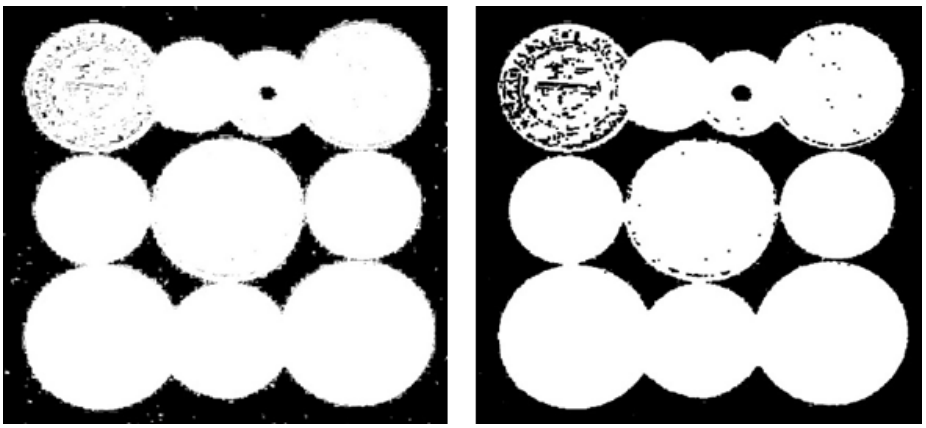


Figure 11 The result of using erosion with a structuring element of size 7×7 whose components are all 1s

2.1.5.1.2 Dilation

Dilation is the technique used to increase the size, fill in holes, and connect broken parts of the object in the binary image. Given A is a binary input image and B is a structuring element, then dilation of A and B denoted as $A \oplus B$, is defined as

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset)\},$$

where $(\hat{B})_z$ is the translation of the reflection of structuring element B by point z.

Figure 12 shows the example of using dilation with different structuring elements, while Figure 13 shows the result of using dilation to connect image components with a square structuring element of size 31×31 whose components are all 1s.

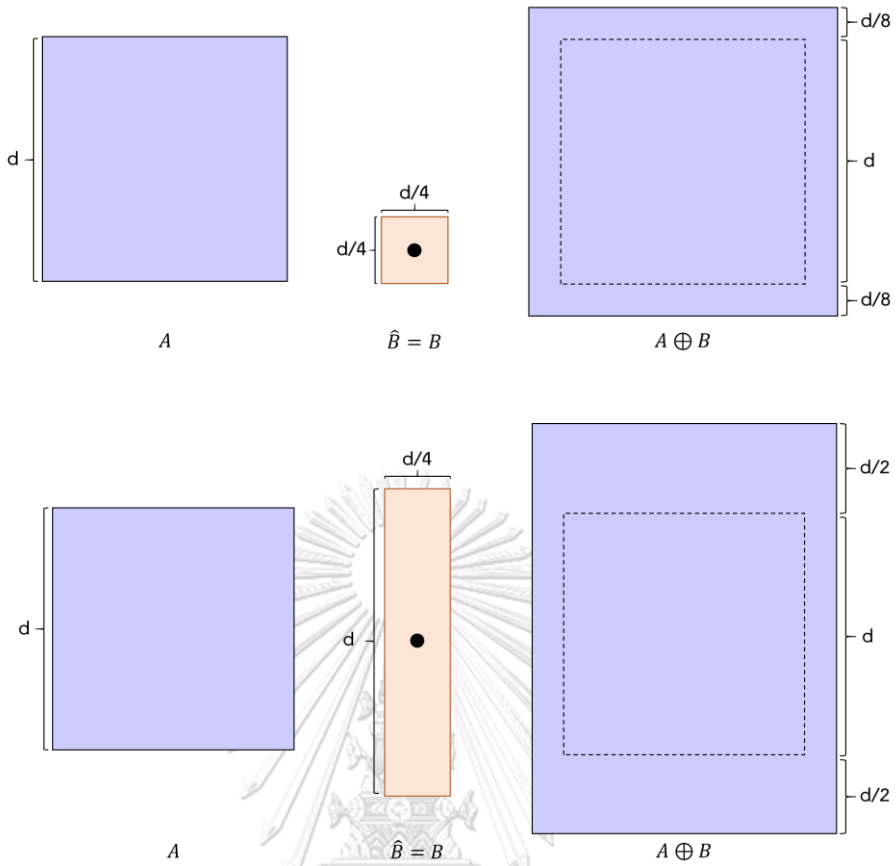


Figure 12 The example of dilation using different structuring elements

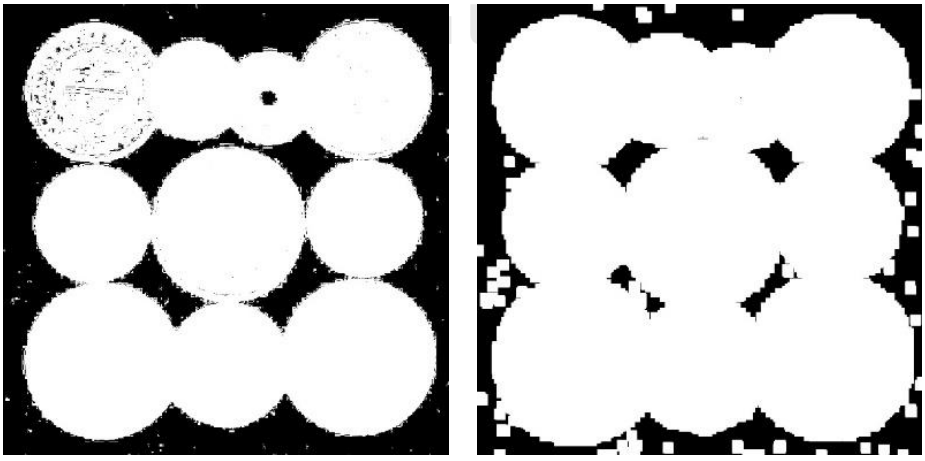


Figure 13 The result of using dilation with a structuring element of size 31×31 whose components are all 1s

2.1.5.2 Opening and Closing

Opening and closing are technique in image processing which use the combination between the erosion and the dilation to process images. Opening commonly used to smooth the contour of the object and remove thin protrusions. Like the opening, closing also used to smooth sections of contours, but closing typically fuses small broken parts, eliminates the gaps in the contour, and fills the small holes inside objects.

The opening of binary image A by structuring element B, denoted by $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B.$$

Hence, the opening of A by B is the erosion of A by B and then followed by the dilation of the result by B.

Similarly, the closing of binary image A by structuring element B, denoted by $\mathrm{A} \bullet \mathrm{B}$, is defined as

$$A \bullet B = (A \oplus B) \ominus B.$$

The closing of A by B is the dilation of A by B and then followed by the erosion of the result by B.

Figure 14 shows the results of using the morphological opening and closing operation with a square structuring element of size 17×17 whose components are all 1s. We can observe that the morphological opening can be used to remove noise around an interesting instance, and the morphological opening can be used to fulfill holes inside an interesting instance.

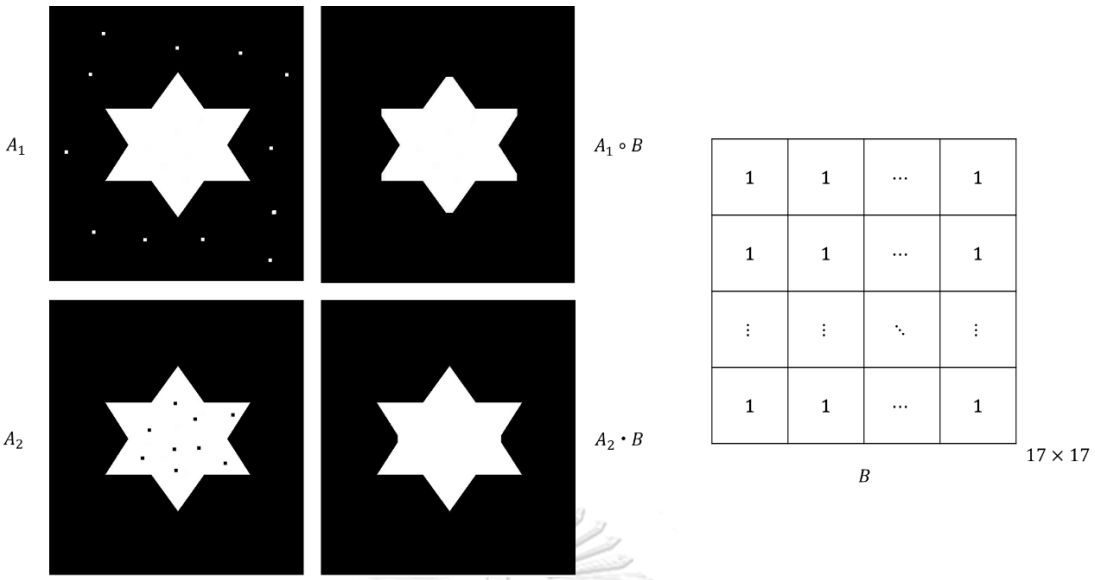


Figure 14 The results of using the morphological opening and closing operation with a square structuring element of size 17×17 whose components are all 1s

2.1.6 Thresholding

Thresholding is one of the segmentation methods that aim to partition an image into a foreground and background. The primary way to separate the objects from the background is to select threshold T to divide each pixel into one of two levels. For any point (x, y) in the image, the segmented image, denoted by $g(x, y)$, is defined as

$$g(x,y) = \begin{cases} 1 & if\ f(x,y) > T \\ 0 & if\ f(x,y) \leq T \end{cases},$$

where $f(x, y)$ is the intensity value of the image f at coordinates (x, y) and T is a constant suitable over an entire image.

2.1.6.1 Otsu's Method

Otsu thresholding method is an essential technique in image processing, which uses to perform automatic image thresholding. This method returns an

optimal threshold that divides pixels into two classes: background and foreground. This threshold is assigned by minimizing intra-class intensity variance.

Suppose a grayscale image f has L distinct integer intensity levels: $0,1,\dots,L-1$. The weighted intra-class variance $\sigma_w^2(t)$, defined as

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t),$$

where t is the threshold, which is a value in the range $[0, L-1]$.

The process for calculating $\sigma_w^2(t)$ is explained next. First, a histogram probability for every pixel value is computed. Let $P(i)$ be the histogram probability of the pixel value $i = 0,1,\dots,L-1$, and Let n_i denote as the number of pixels with intensity $i = 0,1,\dots,L-1$. Then $P(i)$ is defined by the following equation:

$$P(i) = \frac{n_i}{\sum_{i=0}^{L-1} n_i}.$$

After that, each pixel value is separated into two classes, c_1 and c_2, by threshold t, using probability function $q_1(t)$ and $q_2(t)$, defined in the following equation:

$$q_1(t) = \sum_{i=0}^{t} P(i),$$

$$q_2(t) = \sum_{i=t+1}^{L-1} P(i).$$

Class c_1 represents the pixels with intensity in the range $[0, t]$, and class c_2 represents the pixels with intensity in the range $[t, L-1]$.

Next, the mean intensity value of the pixels in class c_1, $\mu_1(t)$, and class c_2 and $\mu_2(t)$ are computed by the following equations:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)},$$

$$\mu_2(t) = \sum_{i=t+1}^{L-1} \frac{iP(i)}{q_2(t)}.$$

Then the variances for class c_1, $\sigma_1^2(t)$, and class c_2, $\sigma_2^2(t)$, are

$$\sigma_1^2(t) = \sum_{i=0}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)},$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{L-1} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}.$$

Finally, the optimal threshold is the value, t^*, that minimizes $\sigma_w^2(t)$:

$$\sigma_w^2(t^*) = \min_{0 \le t \le L-1} \sigma_w^2(t).$$

Figure 15 shows the example result of using Otsu's method with a grayscale image.

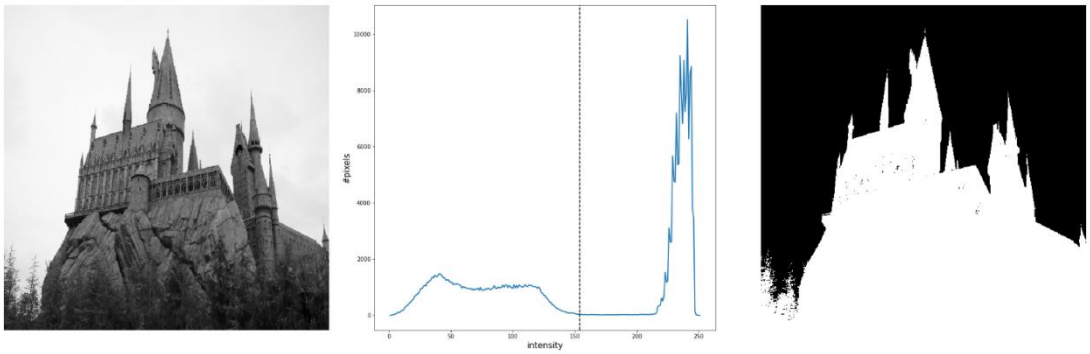


Figure 15 The example result of using Otsu's method

2.2 Machine Learning

Nowadays, Artificial Intelligent (AI) and Machine Learning (ML) play an essential role in many fields of research and practical applications. AI system is applied in people's daily lives, for example, search engines, transportation systems, and fake news filters because it enables human capabilities: reasoning, analysis, and understanding, to be increasing more rapidly, efficiently, and effectively.

AI system will analyze raw data and achieve knowledge by finding insight patterns from that data without explicitly programmed. This capability is called ML [7]. Usually, ML is separated into four categories: association analysis, reinforcement learning, unsupervised learning, and supervised learning [8].

Reinforcement learning is the training of ML models to make a sequence of decisions by giving rewards or punishments for a system. The goal of this learning is to maximize or minimize the total rewards. Unsupervised learning draws inferences and extracts patterns from input data without labeling as a classification. Supervised learning is the process of an algorithm learning from a training dataset that can be taught by feeding it input data and labeled output data. Given $\mathbf{N}$ is the number of samples in training dataset which is a form of input-output vector pairs $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$, where each y_j is defined by an unknown mapping function $y = f(x)$. This learning aims to approximate the mapping function, which can predict the output variables from unseen input data.

2.3 Artificial Neural Networks

An artificial neural network (ANN) is a mathematical model that simulates biological nerve cells' function in the human brain called neurons**.** Figure 16 shows three main parts of a neuron: dendrite, cell body, and axon. A signal will be transmitted from a neuron to other neurons by receiving a signal through the dendrite, transforming a signal through the cell body, and sending the transformed signal via the axon.

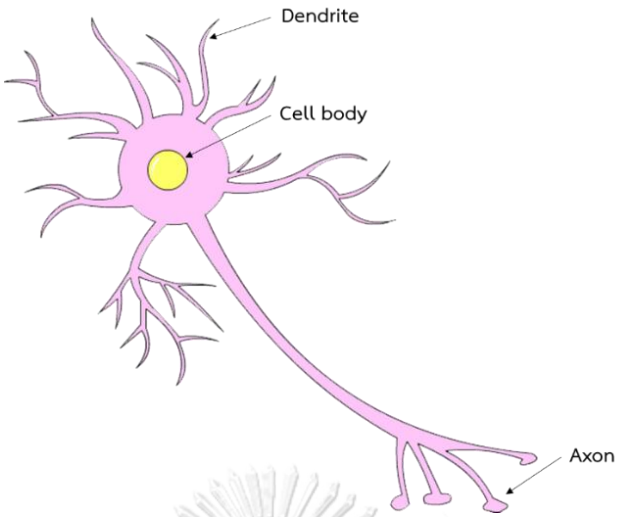


Figure 16 The biological neuron

ANN consists of three main layers, namely the input layer, the hidden layer, and the output layer. Each layer of ANN consists of multiple simple processing units called artificial neurons or nodes, as shown in Figure 17. Artificial neurons try to mimic the structure and behavior of a biological neuron by applying a dot product between input values and weights, then add a bias and use the activation function to transform these values into results, as shown in Figure 18.

Given the number of input N, input $X = (x_1, x_2, \dots, x_N)$, weight $W = (w_1, w_2, \dots, w_N)$, bias b, and activation function φ, the output of an artificial neuron in a hidden layer is defined by the following equation:

$$y = \varphi\left(\sum_{i=1}^{N} w_i x_i + b\right).$$

Generally, the artificial neurons in the output layer do not employ an activation function because the last output layer often uses for representing class scores and uses them to predict the class of input data.

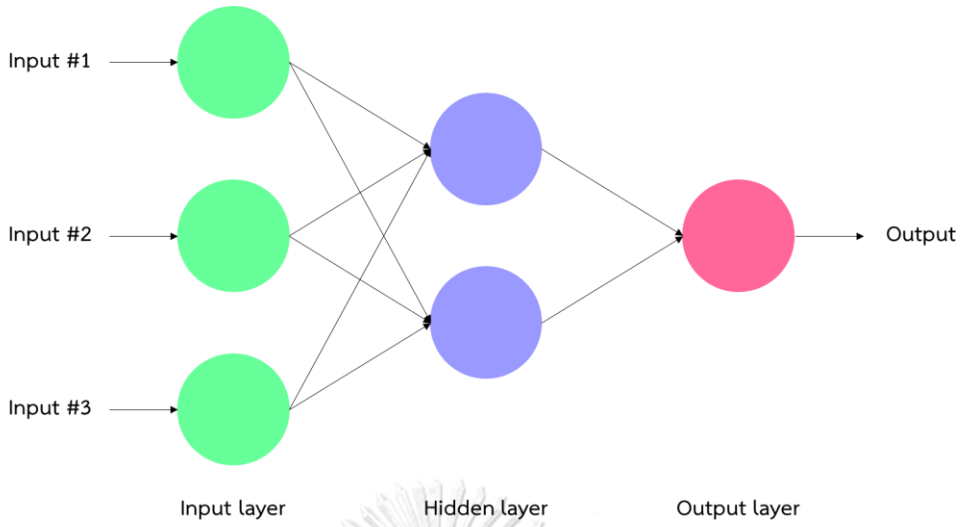


Figure 17 The visualization of an artificial neural network with one hidden layer

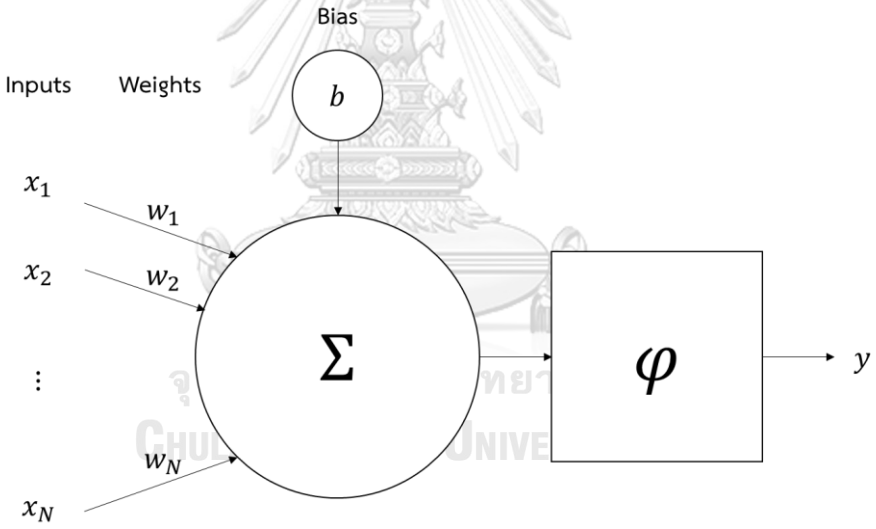


Figure 18 The visualization of an artificial neuron in the hidden layer

2.3.1 Activation Functions

An activation function is used in ANN to bound the output value to some limit. There are numerous activation functions used in ANNs, such as the rectified linear unit function (ReLU), sigmoid function (sigmoid), hyperbolic tangent function (tanh), and exponential linear unit function (ELU), as shown in Figure 19.

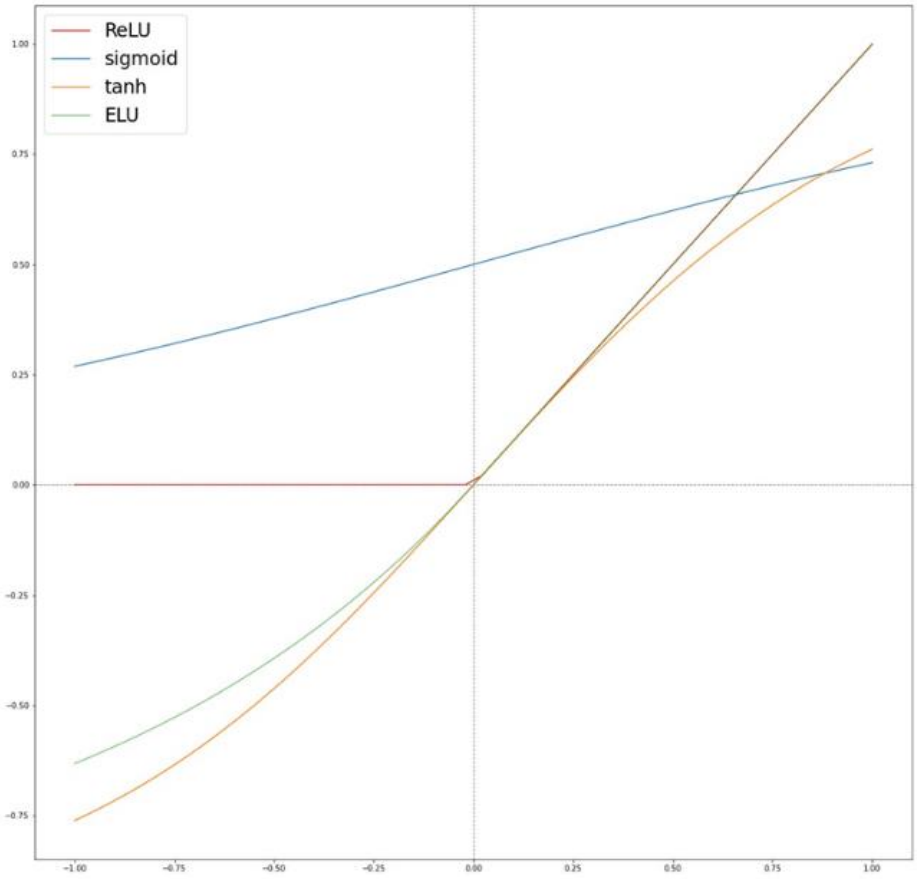


Figure 19 The common activation functions used in ANN

- **Sigmoid –** The *sigmoid* function is a mathematical function which exhibits a S-shaped curve. This function maps real-valued numbers to a continuous range of values between 0 and 1. The sigmoid function is defined by this formula:

$$sigmoid(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}.$$

- **Tanh –** The *hyperbolic tangent* function is like sigmoid, but this function produced the range value of output from -1 to 1. The hyperbolic tangent function is defined by this formula:

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}}.$$

- **ReLU –** The *rectified linear unit* function is the most popular activation function used in deep learning research. It produces the output value above or equal to zero and experiences fewer issues with vanishing gradients than *sigmoid* and *tanh*. The rectified linear unit function is defined by this formula:

$$ReLU(x) = max(0, x).$$

- **ELU –** The exponential linear unit function is a strong alternative to ReLU. ELU can produce negative outputs and converge to zero faster than many activation functions. The exponential linear unit function is defined by this formula:

$$ELU(x) = \begin{cases} x, & x < 0 \\ \alpha \cdot (e^z - 1), & x \geq 0 \end{cases}, \alpha \in \mathbb{R}^+.$$

2.4 Training Artificial Neural Networks

For training an ANN, input data is used to forward pass through the network to produce the predicted output. Then a loss function (or cost function) is selected to evaluate the error between the predicted output and the exact output, also known as the ground truth, and propagate it back through the network to improve trainable network parameters.

2.4.1 Loss Functions

The loss function is used to evaluate the performance of a predictive model by measuring the inconsistency between predicted the output and the ground truth. A value from the loss function is a non-negative real number where decreasing values represent increasing of the model's correctness. It means that to get the best model performance, we have to minimize the loss function.

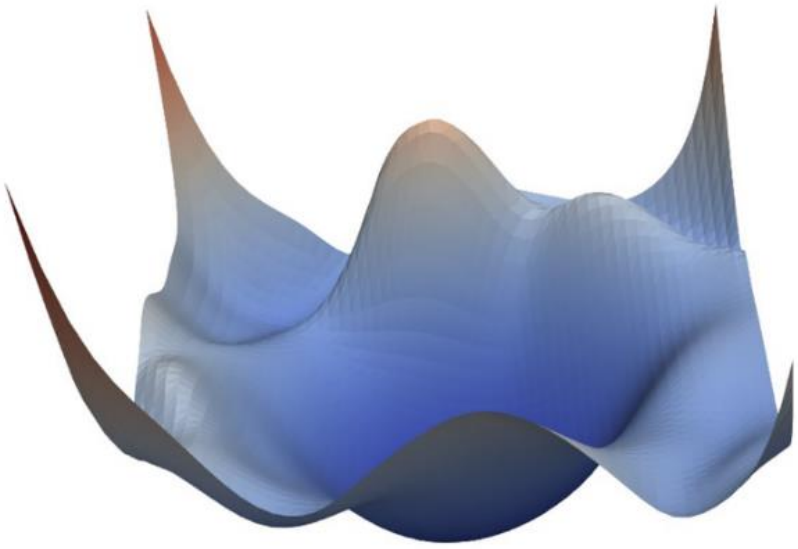


Figure 20 The visualization of the loss function

Source: Adapted from [9]

Given m is the number of training input, $X = (x_1, x_2, \dots, x_m)$ is a training input, $Y = (y_1, y_2, \dots, y_m)$ is a ground truth. The loss function $L_{\hat{\theta}}$, where $\hat{\theta} = (\theta_1, \theta_2, \dots, \theta_n)$ is the vector of trainable parameters, can be written as

$$L_{\hat{\theta}} = k \sum_{i=1}^{m} \ell(f(x_i), y_i),$$

where k is a constant, and $\ell(f(x_i), y_i)$ is a function that compared between the predicted output $f(x_i)$ of the training input x_i and the corresponding ground truth y_i.

Nowadays, there are many loss functions that are usually used in ANNs, such as mean squared error (MSE), mean absolute error (MAE), L1 loss (L1), L2 loss (L2), and cross entropy loss (CE).

- **Mean squared error**

$$L^{MSE} = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2$$

- **Mean absolute error**

$$L^{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - f(x_i)|$$

- **L1 loss**

$$L^{L1} = \sum_{i=1}^{m}|y_i - f(x_i)|$$

- **L2 loss**

$$L^{L2} = \sum_{i=1}^{m}(y_i - f(x_i))^2$$

- **Cross entropy loss**

$$L^{CE} = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(f(x_i)) + (1 - y_i) \cdot log(1 - f(x_i)))$$

2.4.2 Backpropagation

Backpropagation is a standard algorithm of ANNs that use to find the minimum of a loss function in the weight space. The work of this algorithm is to compute the gradient of the loss function L or a vector of all partial derivatives of the loss function L with respect to any trainable parameters θ. Then the gradient descent method (see in section 2.4.3) will use these partial derivatives to improve the parameters θ iteratively. The illustration of the Backpropagation algorithm is shown in Figure 21.

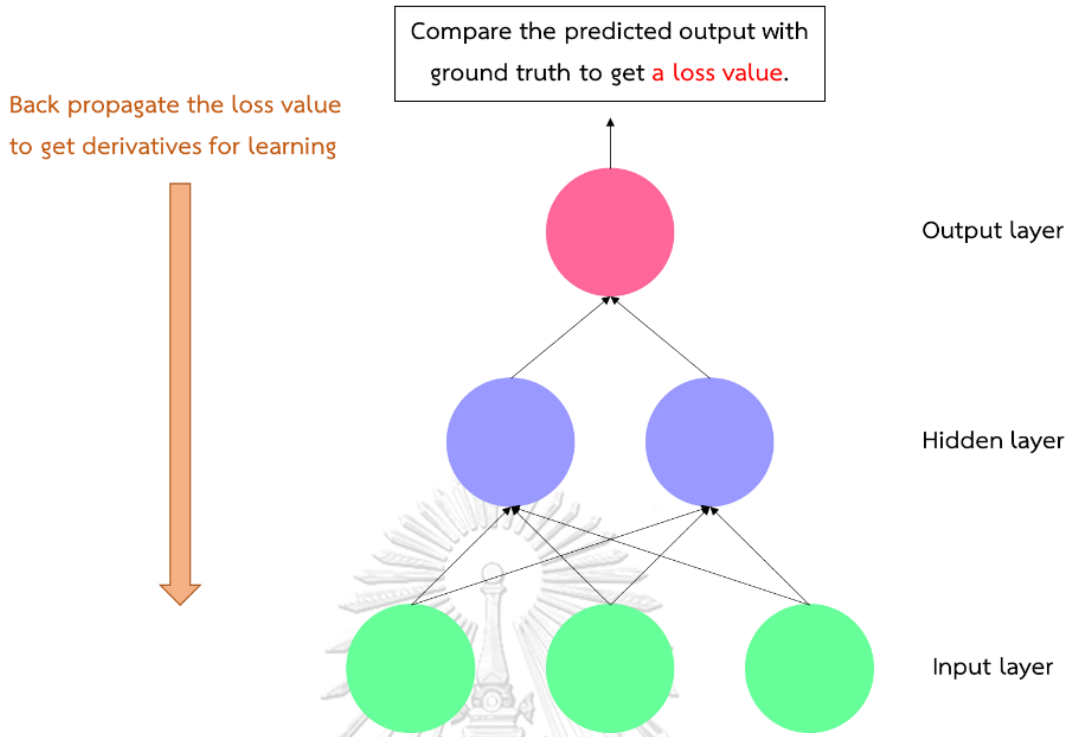


Figure 21 The visualization of the backpropagation method

2.4.3 Optimization Algorithms

A gradient descent algorithm is an optimization method used to minimize the loss function L with respect to any trainable parameters θ. The gradient descent method will update parameters θ in the opposite direction of the gradient of the loss function L:

$$\Delta\theta_j = -\eta \cdot \frac{\partial L}{\partial \theta_j},$$

where $\Delta\theta_j$ is the update change of a trainable parameters θ_j, and η is the learning rate, which is the size of steps we take to achieve the minimum.

2.4.4 Hyperparameters

Hyperparameter, such as a learning rate, the number of an epoch, the loss function, and the batch size, is a parameter that is not learned by a network, and the value is initialized before the learning process begins. The hyperparameter will define a network structure and how the network is trained.

2.4.5 Parameter Initialization

The initial values of the trainable parameters, such as weights w_i and biases b_i, play an essential role in the training phase. For example, using too high initial values may cause harshly huge gradients and make the training process unstable. In contrast, using too low initial values may cause the small gradients and make the training process take a long time. There are many methods proposed for initialization, and one of the methods that researchers commonly use is called Xavier initialization.

2.4.5.1 Xavier Initialization

Xavier initialization is the method which is widely used for parameter initialization. Xavier initialization assigns layer's weights to values chosen from a random uniform distribution between minus one and one and scaled by a normalization factor of $\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}$:

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\right],$$

where n_{in} and n_{out} are the numbers of neurons in the previous layer and next layer, respectively.

2.5 Deep Neural Networks

A deep neural network (DNN) is an ANN which consists of two or more hidden layers. Because DNN has a higher number of hidden layers from the regular ANN, DNN can analyze or solve the more complex data problems. For example, DNN can recognize a sound, recognize graphics, and perform many other tasks related to prediction, creativity, and analysis.

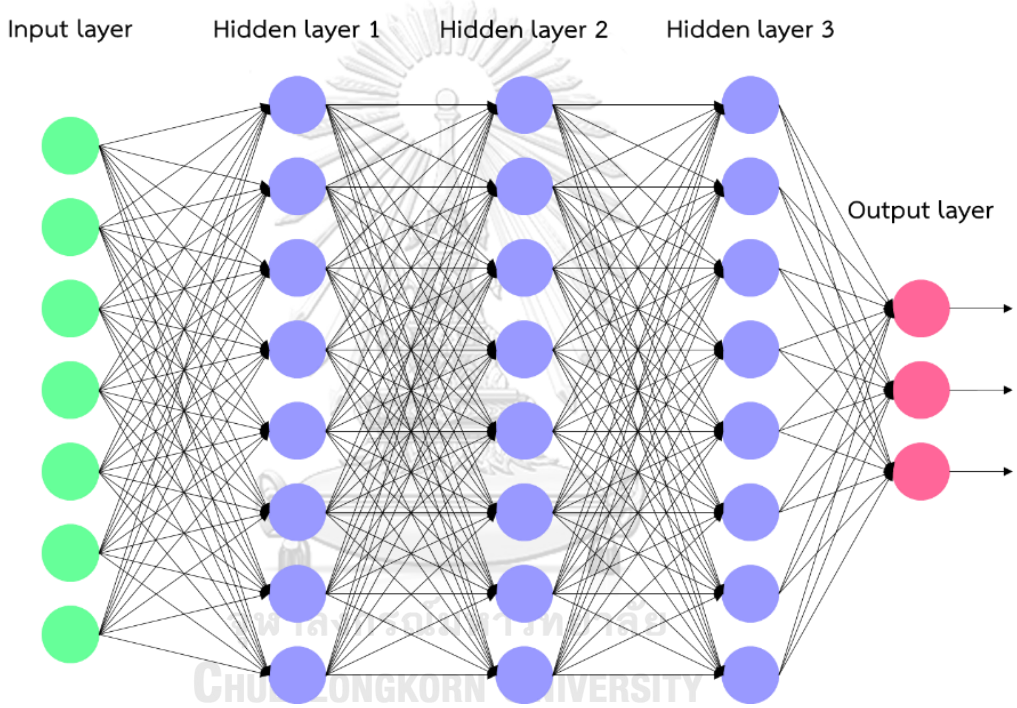


Figure 22 The visualization of a DNN with three hidden layers

2.6 Convolutional Neural Networks

Convolutional neural network (CNN) is a specific type of DNN which is particularly useful in computer vision tasks such as image recognition, object segmentation, and object detection. The process and structure of CNN are like DNN but differ from DNN in two ways: convolutional layers and pooling layers.

2.6.1 Convolutional Layers

A convolutional layer defines a window called a kernel (also known as a mask, template, and filter), an array whose size and coefficients define the neighborhood of operation and the feature of the kernel. For a kernel w size $m \times n$, assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are non-negative integers. Given $f(x, y)$ is the value of the array at point (x, y), the convolution of w and f, denoted as $(w * f)(x, y)$, is defined by the following equation:

$$(w * f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) \cdot f(x + s, y + t).$$

Figure 23 show an example of convolution result between 3×3 kernel and 3×3 input array.

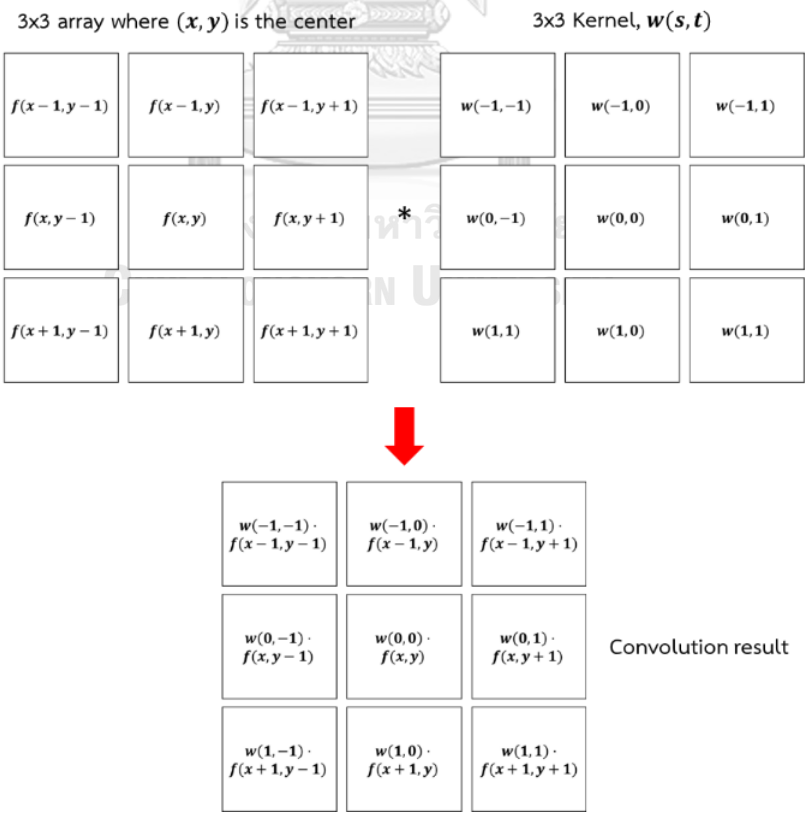


Figure 23 The example of a convolutional operation with kernel of size 3x3

The kernel moves from the top left to the right with a stride value until it parses the complete width. Then it hops down to the left with the same stride value and repeats the process until the entire input array is traversed. The movement of the kernel is shown in Figure 24.

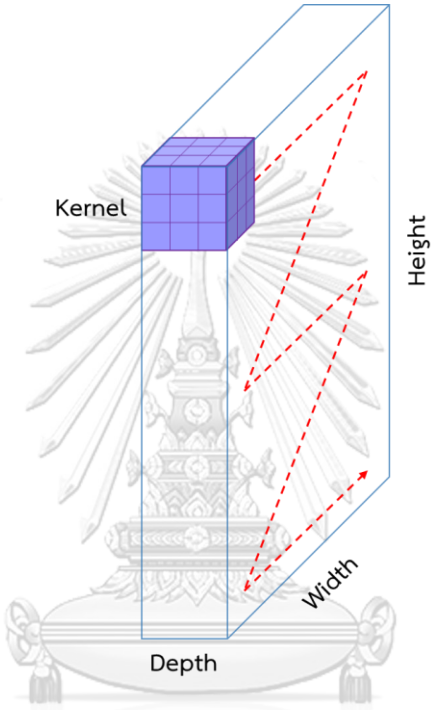


Figure 24 The movement of kernel

2.6.2 Pooling Layers

A pooling layer is a type of downsampling function used to compress a spatial transform of the feature map. Like convolutional layers, the pooling layer defines the window to traverse all the arrays but focuses more on compressing the information. The most common pooling is max pooling.

Max pooling returns the maximum value from a portion of the array, which covers by the window. Max pooling performs as a noise reduction. It ignores all the noisy activations and also de-noising along with dimensionality reduction. Figure 25 shows an example of max pooling result with 2×2 filters and stride 2.

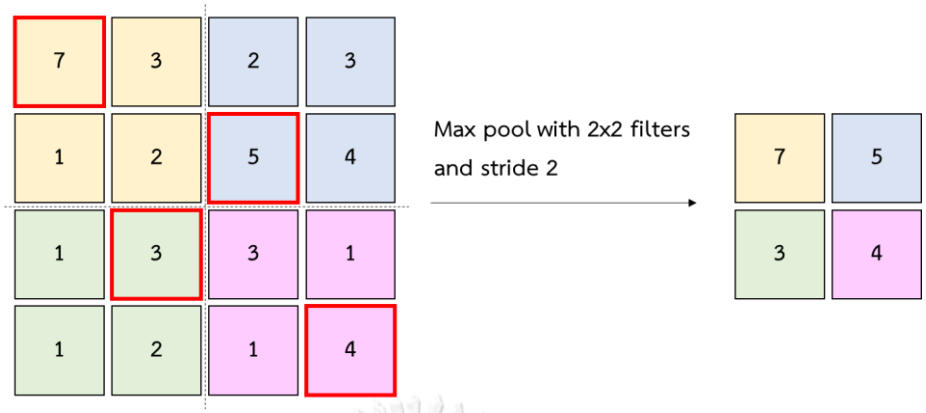


Figure 25 The example result using max pooling with 2x2

2.6.3 CNN Architecture

CNN architecture is designed to mimic the connectivity pattern of neurons of the human brain. A standard classification CNN will process each input by feeding them through a collection of convolutional layers with kernels, pooling layers, fully connected layers, and a loss function to predict the input class. Figure 26 show the visualization of the famous CNN architecture: Visual Geometry Group 16 (VGG16) [10].

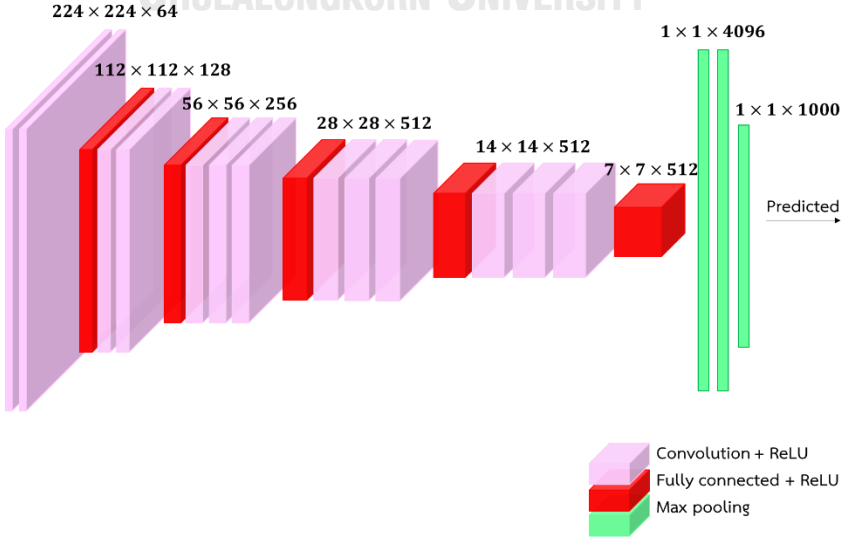


Figure 26 The visualization of the VGG16 architecture

2.7 Object Detection

Object detection is the task which deals with identifying and location instances of objects of a particular class in the image. The object detection networks take images as input and output bounding boxes for all interest objects with a class label, as shown in Figure 27.

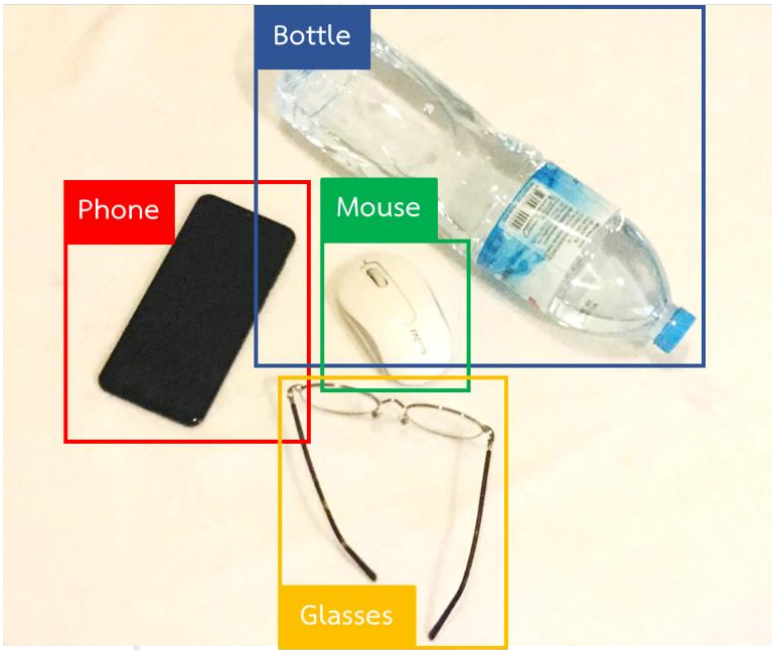


Figure 27 The visualization of an example image with ground truth bounding boxes

2.8 Single Shot Scale-invariant Face Detector

Single Shot Scale-invariant Face Detector (S^3FD) [11] is a real-time face detector that is implemented and trained on the AP2019 dataset in this project. S^3FD is a one-stage face detection network that requires only a single pass through the entire CNN and predicts bounding boxes with confidence scores for objects in the given image.

2.8.1 Multi-scale Feature Maps and Prior Boundary Boxes

To predict object bounding boxes and class scores, S^{3}FD defines a collection of bounding boxes called prior boxes at each cell. For each cell, the network generates a set of predictions composed of a bounding box with confidence scores for each class. Then the prior boxes are matched to the ground truth boxes, and those with the IoU value (see in section 2.9.1) higher than a set of thresholds are selected. Finally, the matched prior boxes are encoded into four offsets with confidence scores and used to compute the loss value during the training phase. Figure 28 shows the visualization of an example prior box in con6_2 and conv7_2.

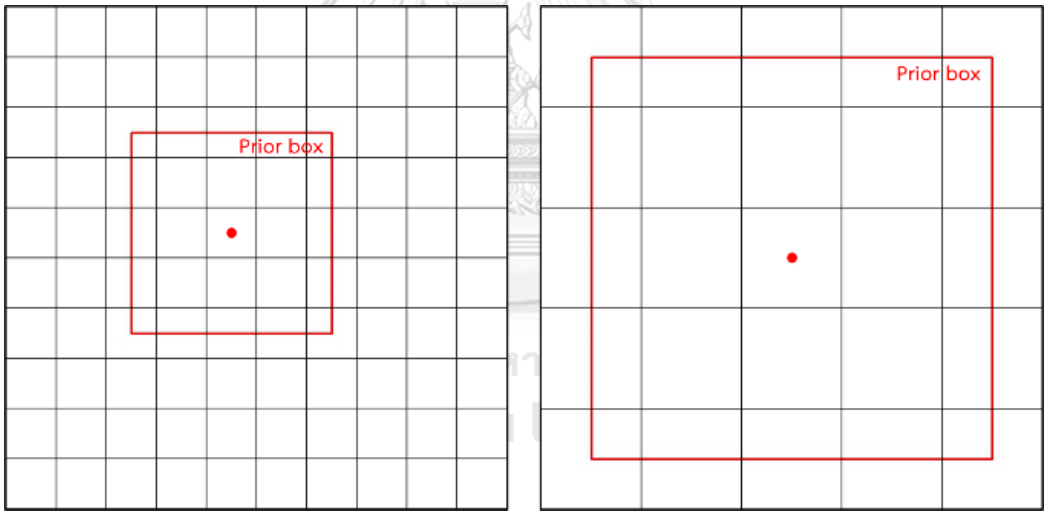


Figure 28 The visualization of an example prior box in con6_2 and conv7_2

Using prior boxes is usually applied by many other object detection methods. However, one thing that distinguishes S^{3}FD from the other is that the network applies it in multiple layers instead of only applying it on the last layer. Applying the technique of using prior boxes with multiple layers makes S^{3}FD use shallow layers to detect smaller faces and more deep layers to detect larger faces.

For each detection layer, the prior box scales are designed as listed in the second column of Table 2, and the prior boxes are 1:1 aspect ratio (width: height) because the bounding box of the face is approximately square.

Table 2 The prior box scales of each detection layer

Position	Scale
conv3_3	16
conv4_3	32
conv5_3	64
conv_fc7	128
conv6_2	256
conv7_2	512

2.8.2 S^3FD Architecture

The architecture of S^3FD, shown in Figure 29, is based on VGG16 network with auxiliary structures:

- **Base convolutional layers:** We keep the layers from conv1_1 to pool5 of VGG16 and discard all others.
- **Extra convolutional layers**: we keep the layers fc6 and fc7 of VGG16 and add extra layers behind them.
- **Detection convolutional layers**: Select conv3_3, conv4_3, conv5_3, conv_fc7, conv6_2 and conv7_2 as the detection layers.
- **Normalization layers**: Due to the difference in feature scales of conv3_3, conv4_3, and conv5_3, therefore we use L2 normalization to rescale their norm to 10, 8, and 5, respectively. After that, the scales are learned during the backpropagation.

- **Predicted convolutional layers**: For each detection layer, the layer will be followed by $p \times 3 \times 3 \times q$ convolutional layers, where 3×3 is the kernel size, and p and q are the numbers of an input channel and output channel. For each prior, we predict four offsets corresponding to its coordinates and N_s class scores for classification, where $N_s = N_m + 1$ (N_m is the max-out background label) for conv3_3 detection layer and $N_s = 2$ for other detection layers.
- **Multi-task loss layer**: we use softmax loss and smooth L1 loss for classification and regression, respectively.

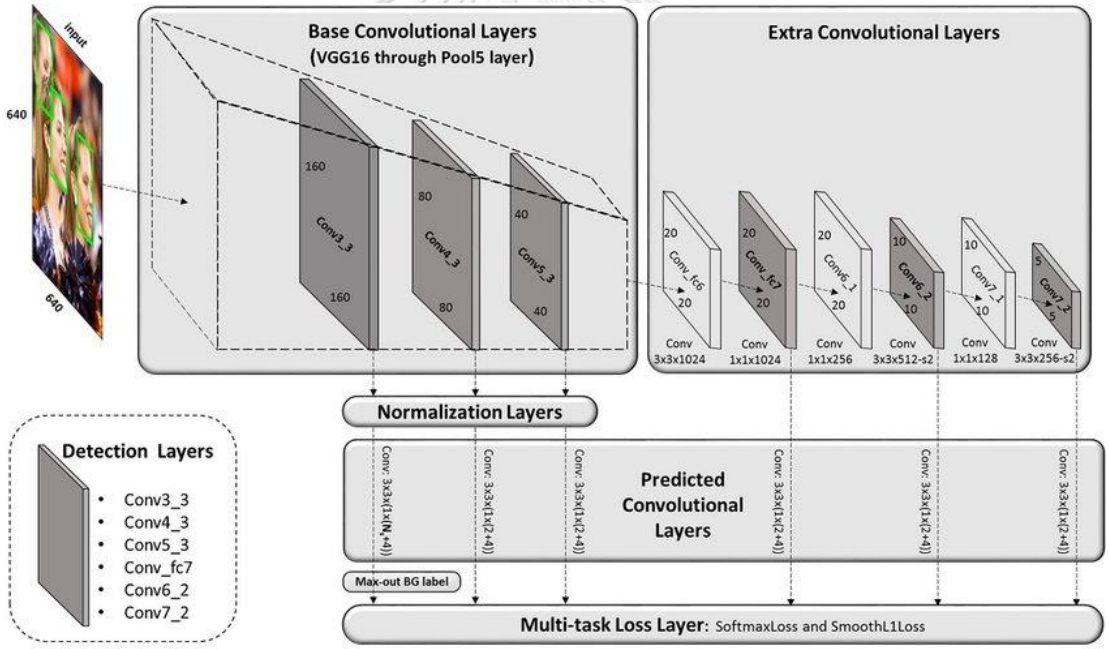


Figure 29 The visualization of the Single Shot Scale-invariant Face Detector (S^3FD) architecture

2.8.3 Max-out Background Label

For a 640×640 input image, it has 34,125 priors, while about 75.02% of them come from the conv3_3 detection layer (see in the third column of Table 3), which

related to the smallest prior (16×16). These smallest priors contribute most to the incorrect predictions (false positives).

Table 3 The detailed information about priors in a 640×640 image

Position	Number	Percentage (%)
conv3_3	25600	75.02
conv4_3	6400	18.76
conv5_3	1600	4.69
conv_fc7	400	1.17
conv6_2	100	0.29
conv7_2	25	0.07

To solve this issue, a max-out background label, which is the method that can reduce false positive rate, is applied for the conv3_3 detection layer by predicting N_m class scores for background labels and selecting the highest value as a final score, as display in Figure 30.

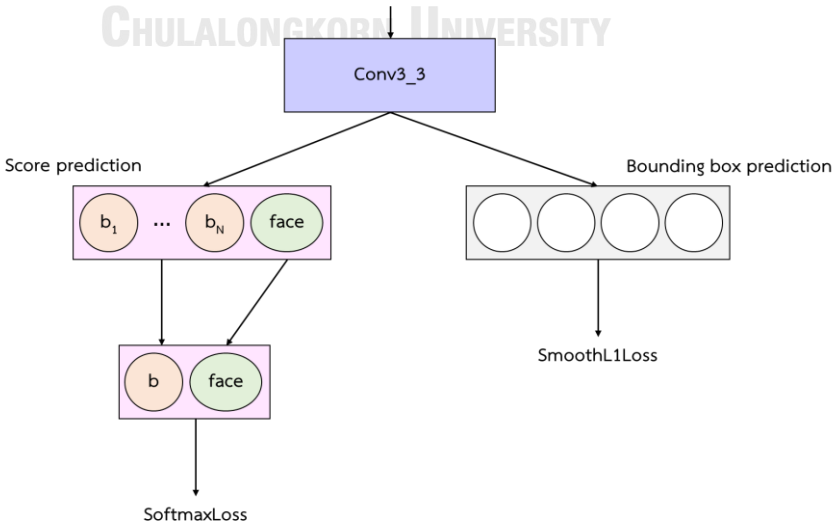


Figure 30 The visualization of the max-out background label

2.8.4 Loss Function

The loss function used by S^{3}FD is the multi-task loss defined in RPN [12]. The multi-task loss function is a sum between weighted confidence loss (classification) and weighted localization loss (regression):

$$L(\{p_i\},\{t_i\}) = \frac{\lambda}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*),$$

where i is the index of a prior, and p_i is the predicted probability that prior i is an aircraft. The ground truth label p_i^* is 1 if the prior is positive, 0 otherwise. t_i is a vector representing four parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground truth box associated with a positive prior. The classification loss $L_{cls}(p_i, p_i^*)$ is softmax loss over two classes (non-aircraft vs. aircraft), the regression loss $L_{reg}(t_i, t_i^*)$ is the smooth L1 loss defined as

$$L_{reg}(t_i, t_i^*) = L1^{smooth}(t_i - t_i^*),$$

where

$$L1^{smooth}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}.$$

The term $p_i^* L_{reg}$ means the regression loss is activated only for positive priors and ignored otherwise. These two terms are normalized by N_{cls} and N_{reg}, and weighted by a balancing parameter λ ($\lambda = 1$ in S^{3}FD paper). In this implementation, the cls term is normalized by the number of positive and negative priors, and the reg term is normalized by the number of positive priors.

2.8.5 Hard Negative Mining

Hard negative mining is the method used for making the training process is stable and faster in optimization. Instead of using all negatives, the negatives are

sorted by their confidence loss values and pick the negatives with the top loss and keep the ratio between the negatives and positives is at most 3:1.

2.8.6 Non-maximum suppression

Non-Maximum suppression (NMS) is a post-processing technique used to make sure that the prediction bounding box identifies a particular object in the given image only once. First, select the top-k confidence scores (k=200 in S^3FD paper) from the set of prediction bounding boxes and discard all others. Next, choose the box with the highest confidence score and calculate the IoU value with the remaining boxes. If there is a box that has the IoU value higher than the threshold, remove it. Finally, repeat the process until there are no more prediction boxes left.

2.9 Performance Evaluation

Performance evaluation is defined as a procedure to measure the results. The two most common metrics used for evaluating an object detection system are Intersection over Union, Precision and Recall.

2.9.1 Intersection over Union

Intersection over Union (IoU) is an evaluation metric used to measure an object detector's accuracy on a dataset. IoU calculates the value of the area of overlap between a predicted bounding B_{Pred} and the corresponding ground truth box B_{GT}:

$$IoU = \frac{|B_{Pred} \cap \mathrm{B_{GT}}|}{|B_{Pred} \cup \mathrm{B_{GT}}|} = \frac{|B_{Pred} \cap \mathrm{B_{GT}}|}{|B_{Pred}| + |B_{GT}| - |B_{Pred} \cap \mathrm{B_{GT}}|}.$$

Figure 31 shows the three IoU examples. A score that closes to 1 means that the predicted bounding box accurately matches the ground truth bounding box. The score that closes to 0 means that the predicted bounding box and the ground truth bounding box are not overlapping on each other.

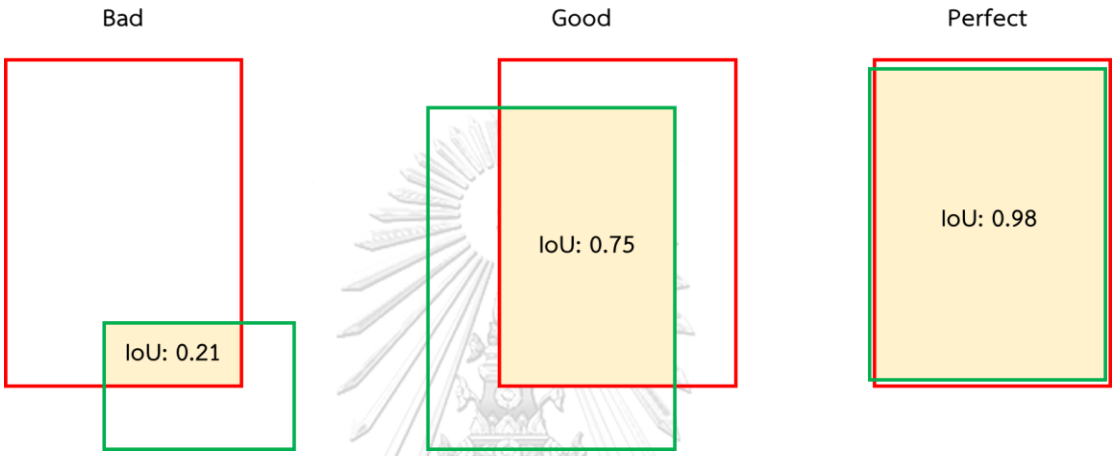


Figure 31 The examples of the Intersection over Union (IoU)

2.9.2 Precision and Recall

Precision and recall are mathematical functions used to measure the accuracy of data retrieval, classification, and identification.

Precision (P), known as the positive predictive value, is defined as the fraction of the number of correct predictions (true positives), T_P, over the sum between the number of incorrect predictions (false positives), F_P, and the number of true positives:

$$P = \frac{T_P}{F_P + T_P}.$$

Recall (R), known as sensitivity, is defined as the fraction of the number of true positives over the sum between the number of missed predictions (false negatives), F_N, and the number of true positives:

$$R = \frac{T_P}{F_N + T_P}.$$

CHAPTER III

Proposed Method

In this work, we propose the method that combines a Viridis saliency map with the Single Shot Scale-invariant Face Detector (S^3FD), see in section 2.8. We create a one-dimensional image called Viridis saliency map, which can reduce the background's complexity well and combine it with the three-dimensional RGB input image into a four-dimensional input image of size 640×640 before feeding through the S^3FD network. The architecture of our approach is shown in Figure 32.

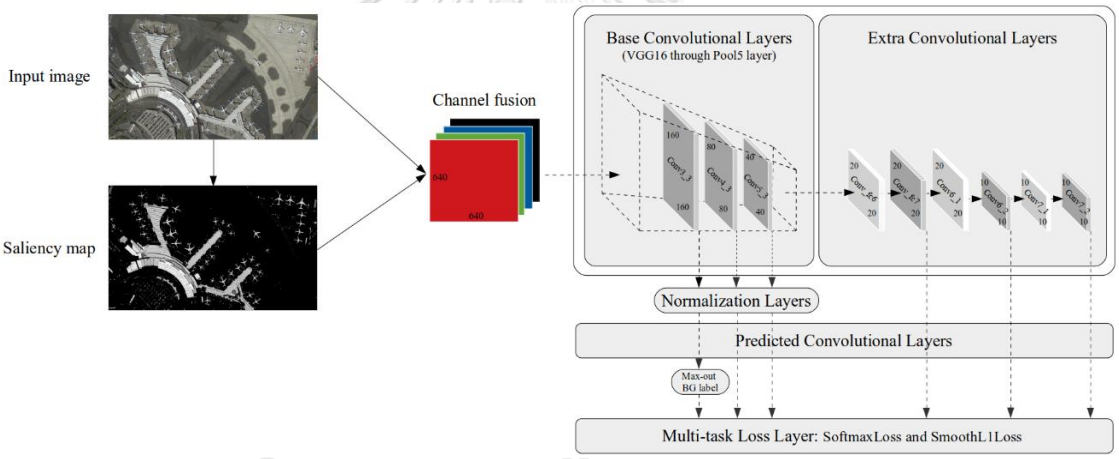


Figure 32 The visualization of our proposed method architecture

3.1 Viridis Saliency Map

The process for creating a Viridis saliency map, shown in Figure 33, follows these five steps:

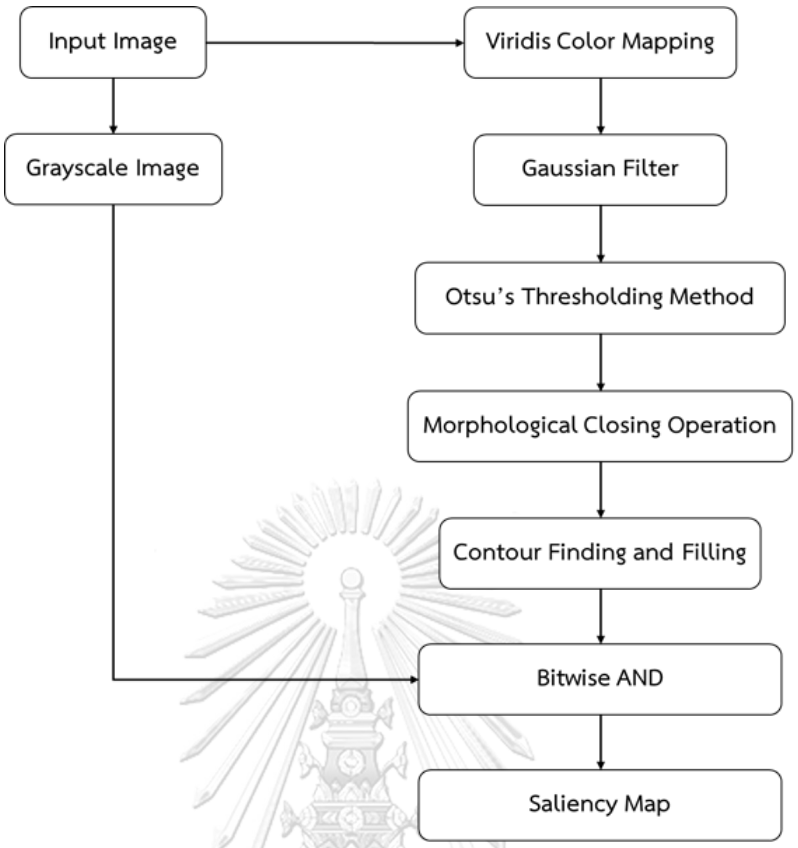


Figure 33 The process of the Viridis saliency map creation

1. We transform an RGB color input image into Viridis color image and grayscale image. Then we choose the last channel of the Viridis color image, which can perform the shape of the aircraft and reduce noise from the background well as shown in Figure 34(e), to apply in the next step.

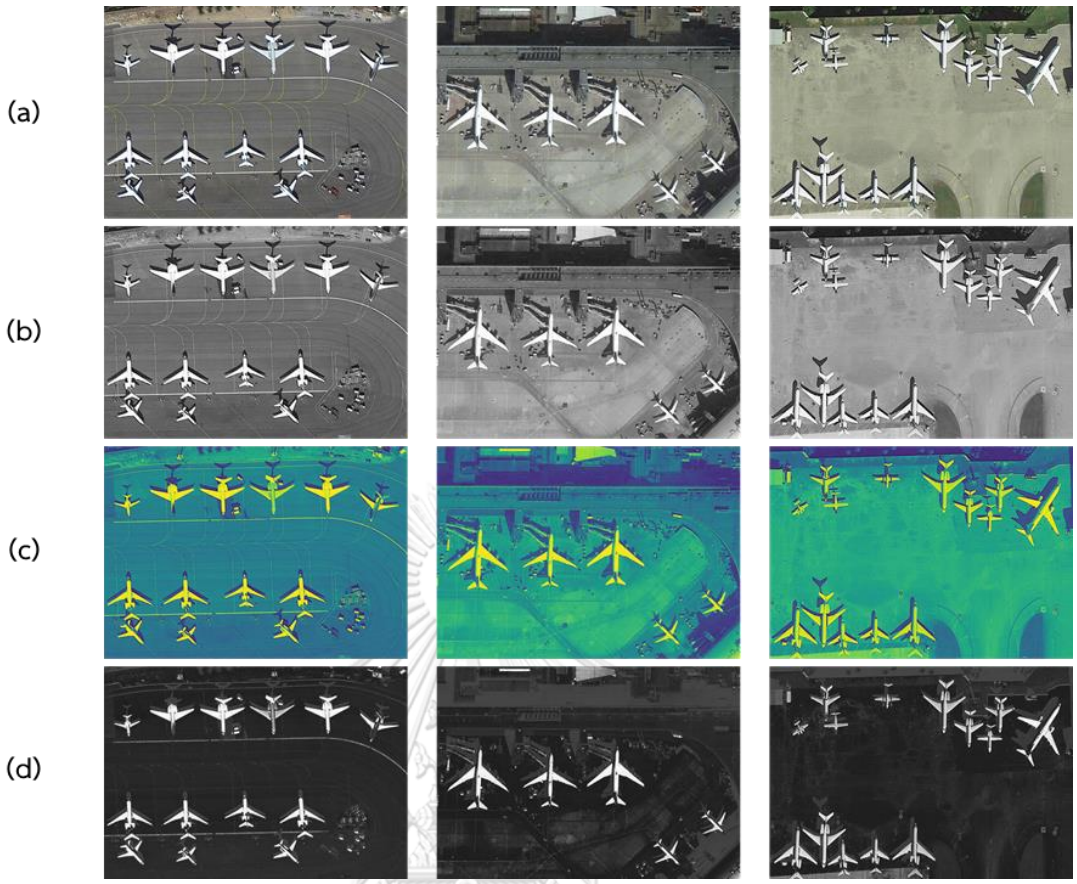


Figure 34 (a) Input images, (b) grayscale images of (a), (c) Viridis color images of (a), (d) the last channel of the Viridis color images of (c)

2. We use the Gaussian filter with a kernel size 5×5, whose components are all 1s, to blur the image. The example results of using Gaussian filtering are shown in Figure 35(b).

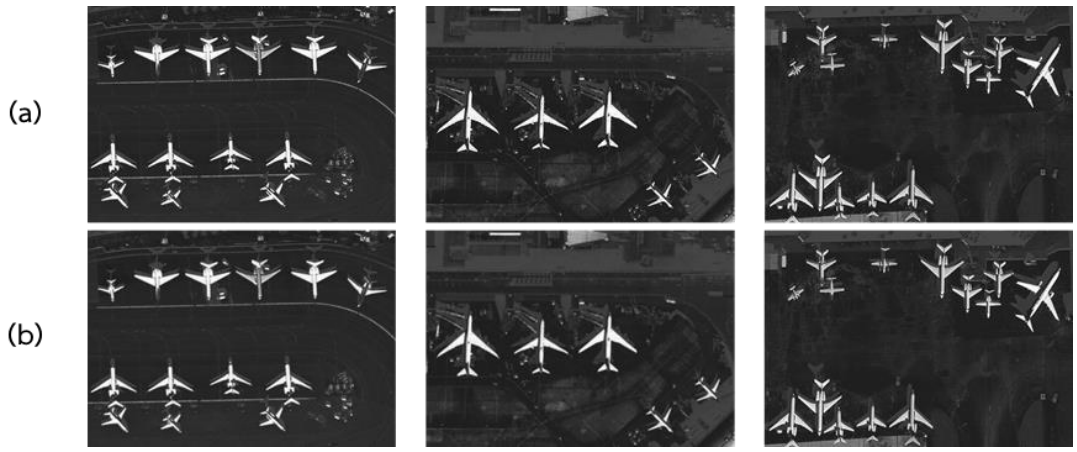


Figure 35 (a) Input images (the last channel of Viridis color images), (b) the results of using Gaussian filtering of (a)

3. We use the Otsu's thresholding method to convert the blurred image into a binary image, as shown in Figure 36.

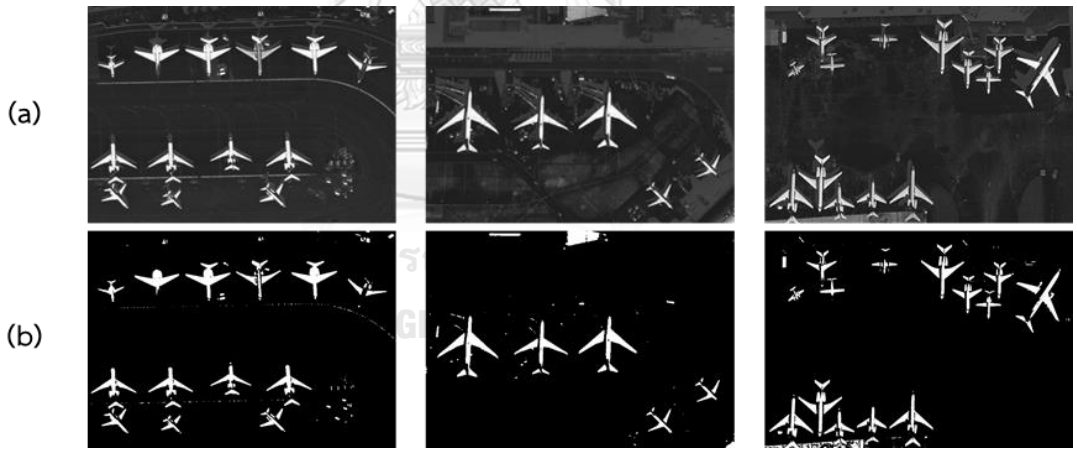


Figure 36 (a) Input images (blurred images), (b) the results of using the Otsu's method of (a)

4. We apply the morphological closing operation with a square structuring element of size 13×13 whose components are all 1s to connect some separated parts of the aircraft body.

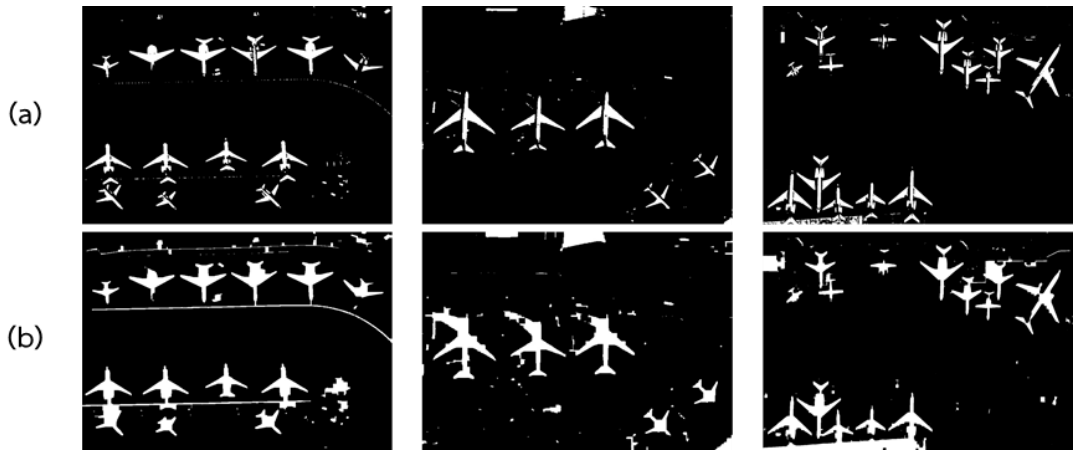


Figure 37 (a) Input images (binary images), (b) the result of using the morphological closing operation of (a)

5. We create a binary mask image using the CHAIN_APPROX_SIMPLE [13] algorithm, a kind of contour approximation method in the OpenCV library, to find the contour of all objects in the image. Then fulfill the area inside the contours.

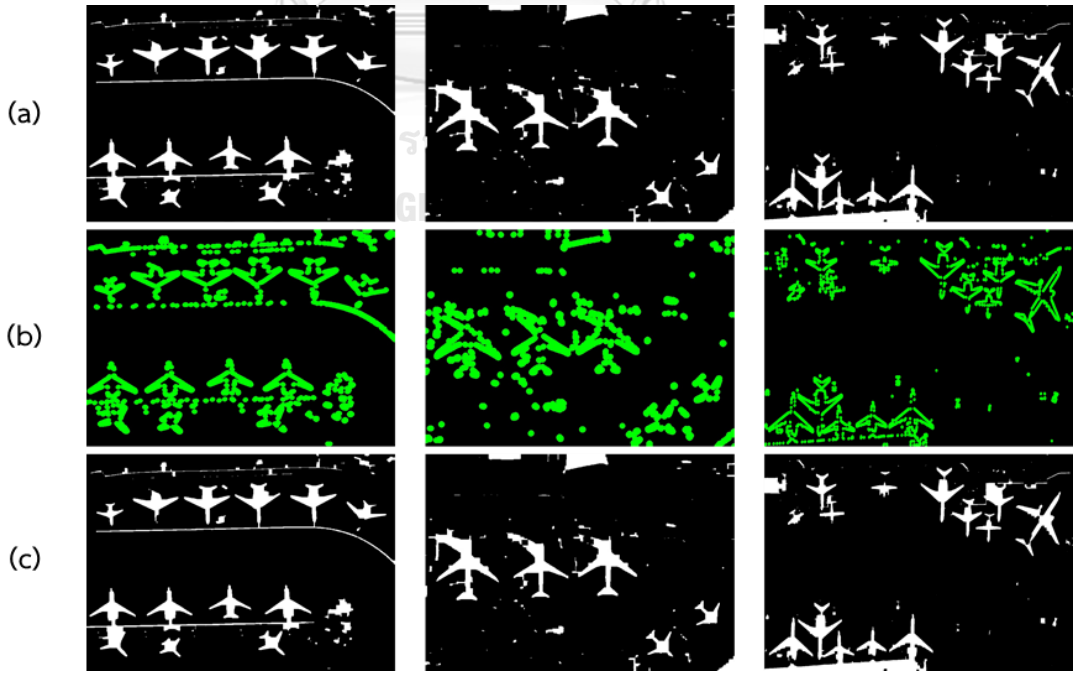


Figure 38 (a) Input images, (b) the contour points of all object in (a), (c) the result of the filling inside all objects in contour points (b)

6. Due to the better detail from the grayscale image, we use bitwise AND operation to combine the binary mask image with a grayscale image to make the Viridis saliency map instead of combining it with the last channel of the Viridis color image.

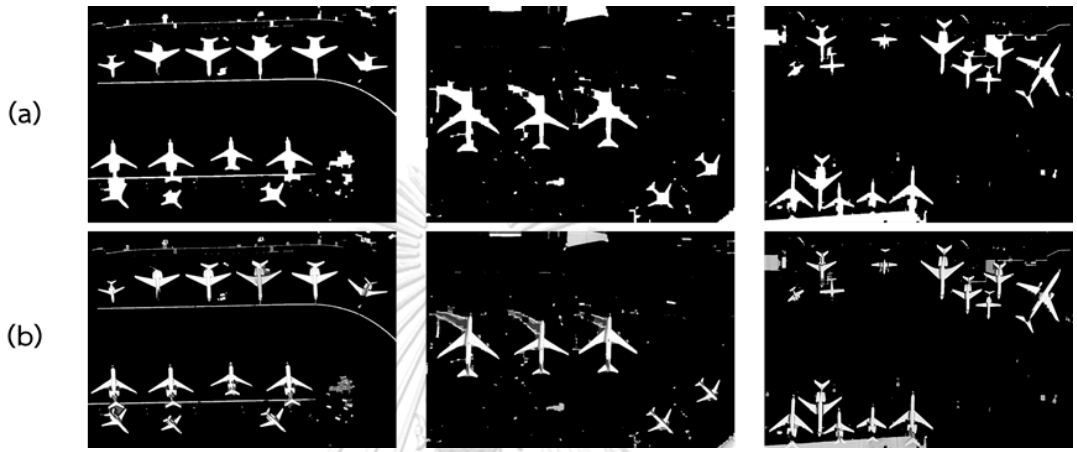


Figure 39 (a) Input images (filled images), (b) the Viridis saliency maps

3.2 Applying Viridis Saliency Map with S^{3}FD

To combine the Viridis saliency map with the S^{3}FD network, we join an RGB input image with its Viridis saliency map as a 4-dimensional input array. Then feed it into the S^{3}FD network for detecting the aircraft. The S^{3}FD network parameters that we used in this work are the same in the S^{3}FD paper. In addition, we use precision and recall (see in section 2.9.2) for evaluating the performance of our proposed method. The result will be described in Chapter IV.

CHAPTER IV

Results and Discussion

In this work, the proposed method is built using PyTorch, an open-source machine learning library, with self-programmed python code and preexisting python code. The computer used to train this model has a single NVIDIA GTX 1050ti GPU. In addition, we create our dataset for training and testing called AP2019, which collects remote sensing images containing aircraft inside from Google Earth. This dataset composes of 206 remote sensing images with bounding box annotations for all 4,828 aircraft. We separate our dataset into a training set consisting of 160 images (3,347 aircraft), a validation set consisting of 23 images (483 aircraft), and a test set consisting of 23 images (998 aircraft). The example images from our dataset are shown in Figure 40.

Figure 40 Example images from the AP2019 dataset

For training, our method's hyperparameter settings are identical to the hyperparameters used in the S^3FD paper. Some examples of these hyperparameters are the interval between prior box center points, kernel sizes, threshold values, and loss function parameters.

Before feeding the input image into the model, the image must be resized to 640x640 pixels. The examples of resized image (640x640) from our dataset are shown in Figure 41. Moreover, to increase the input data diversity, we use the same data augmentation techniques in the S^3FD paper. Examples include random image cropping, random image expansion, random hue, random contrast, random saturation, and random brightness.

Figure 41 The examples of resized image (640x640) from the AP2019 dataset

Our method was trained on AP2019, which contained only remote sensing images and annotations for aircraft, using a batch size of 2 and a learning rate of 10^{-3}. The iteration for this training is limited to 23,000 repetitions because the computation resources, for example, GPU and RAM, are limited. Our training and validation loss of each iteration is shown in Figure 42.

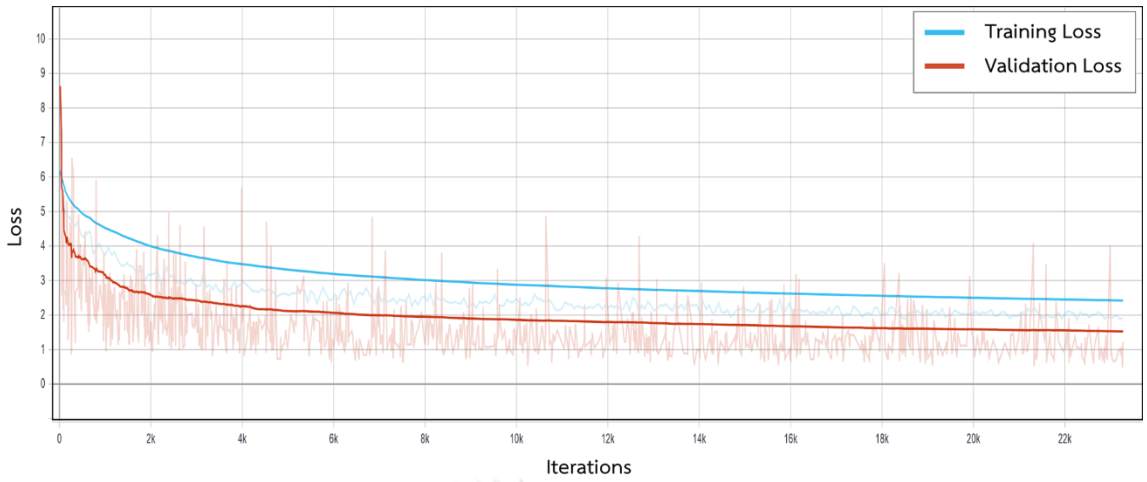


Figure 42 Training and validation losses for 23,000 iterations on AP2019

For evaluating our method's performance, we used the IoU threshold of 0.6 and compared it with the other two methods: original S^3FD and SSD [14]. The result of our method is shown in Table 4.

Table 4 The comparative results of our proposed method

Method	Precision (%)	Recall (%)
SSD	63.74	46.59
S^3FD	**84.95**	94.99
Ours	77.62	**98.00**

In Table 4, the results show that our method performed exceptionally well in the term of recall rate compared to the original S^3FD and the SSD. However, our method provided a lower precision rate than the original S^3FD, as our method gives a very high number of incorrect predictions.

Figure 43-48 shows examples of detections and missed detection in six example images from 23 test images in AP2019 made by our method trained on the complete AP2019 dataset. In these examples,

- **Green boxes** represent correct predictions,
- **Red boxes** represent missed predictions,
- **Yellow boxes** represent incorrect predictions.

We can observe that our method can detect aircraft in remote sensing images well, but our method is still confused by the object that has a shape like an aircraft, and the aircraft that park very close to each other cannot be detected.

Figure 43 The example result of aircraft detection by our proposed method

Figure 44 The example result of aircraft detection by our proposed method

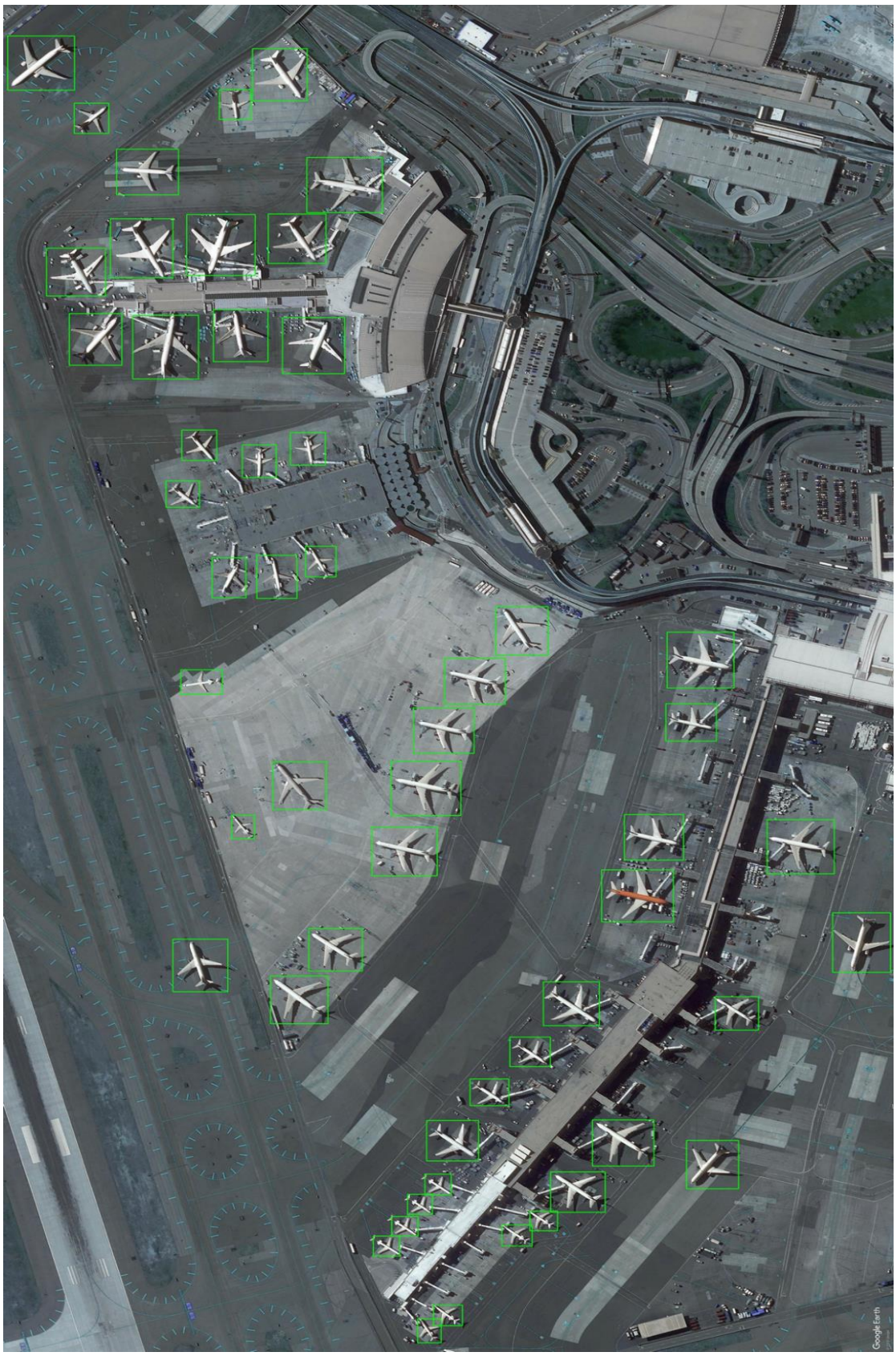


Figure 45 The example result of aircraft detection by our proposed method

Figure 46 The example result of aircraft detection by our proposed method

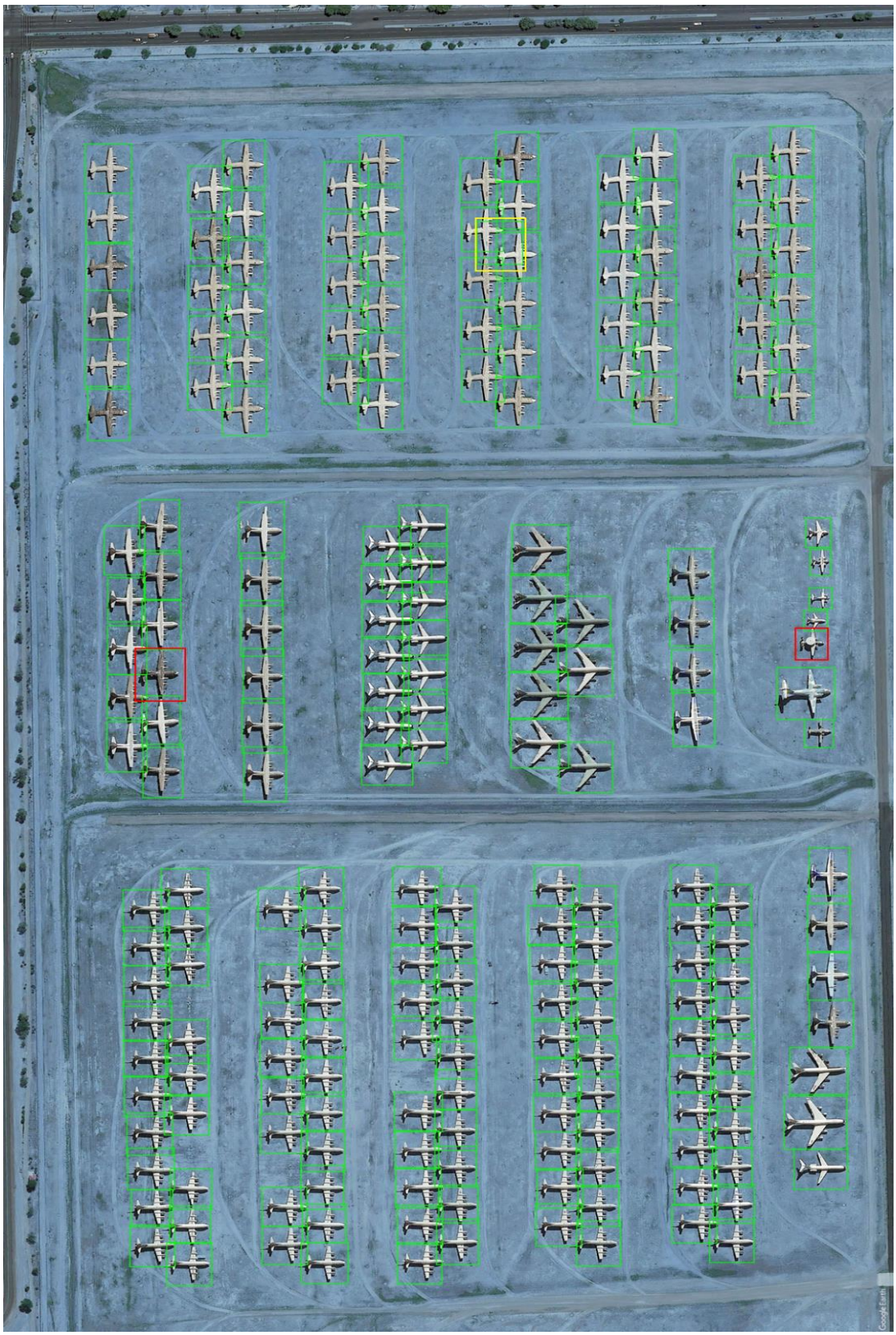


Figure 47 The example result of aircraft detection by our proposed method

Figure 48 The example result of aircraft detection by our proposed method

CHAPTER V

Conclusions

In this thesis, we present a method to detect an aircraft from remote sensing images trained on the AP2019 dataset. The proposed method combines a Viridis saliency map, which can remove the complexity of background in remote sensing images well, with the S^3FD network to handle the various scales of the aircraft in the images. The proposed method was evaluated on 24 remote sensing images that contain 998 aircraft inside. The results in Table 4 show that our proposed method's recall is very high because almost aircraft from the 24 remote sensing images are detected, but the precision of our proposed method decreases because the network was produced many incorrect predictions (false positive) compared to the original S^3FD.

5.1 Future Work

In this work, the proposed method aims to detect aircraft from remote sensing images as much as possible. However, our proposed method is confused by the object that looks similar to the aircraft and caused many incorrect predictions. Therefore, we will find some way to decrease this proposed method's false positive rate, and we will try to improve the Viridis saliency map to reduce noise better to make the detector more accurate and more efficient. Besides, we will do more experiments about the change of model's performance when the model's parameters are adjusted, and we will try the other performance evaluation criteria that appropriate to this work, such as the $F_\beta - score$ [15], which can weight the importance of the recall value.

REFERENCES

1. Wu, H., et al. *Fast aircraft detection in satellite images based on convolutional neural networks*. in *2015 IEEE International Conference on Image Processing (ICIP)*. 2015.
2. Chen, X., et al., *Aircraft Detection by Deep Convolutional Neural Networks.* IPSJ Transactions on Computer Vision and Applications, 2014. **7**: p. 10-17.
3. Han, Z., et al. *Fast aircraft detection based on region locating network in large-scale remote sensing images*. in *2017 IEEE International Conference on Image Processing (ICIP)*. 2017.
4. Gonzalez, R.C. and R.E. Woods, *Digital Image Processing*. 4 ed. 2018: Pearson, Inc.
5. DerSarkissian, C. *Lung X-ray*. 2018; Available from: https://www.webmd.com/lung-cancer/ss/slideshow-lung-cancer-overview.
6. Van der Walt, S.J. and S. Nathaniel. *A Better Default Colormap for Matplotlib*. 2015; Available from: http://bids.github.io/colormap/.
7. Dennis, F.K. and L. Pat, *Machine learning as an experimental science*, in *Proceedings of the 3rd European Conference on European Working Session on Learning*. 1988, Pitman Publishing, Inc.: Glasgow, UK. p. 81–92.
8. Alpaydin, E., *Introduction to Machine Learning*. 4 ed. 2020: The MIT Press.
9. Mohan, A. *Visualizing the Loss Landscape of Neural Nets*. 2019; Available from: https://www.cs.umd.edu/~tomg/projects/landscapes/.
10. Liu, S. and W. Deng. *Very deep convolutional neural network based image classification using small training sample size*. in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. 2015.
11. Zhang, S., et al., *S³FD: Single shot scale-invariant face detector*, in *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017.
12. Ren, S., et al., *Faster R-CNN: Towards real-time object detection with region proposal networks*, in *arXiv e-prints*. 2015. p. arXiv:1506.01497.
13. Suzuki, S. and K. be, *Topological structural analysis of digitized binary images*

by border following. Computer Vision, Graphics, and Image Processing, 1985. **30**(1): p. 32-46.

14. Liu, W., et al., *SSD: Single shot multibox detector*, in *arXiv e-prints*. 2015. p. arXiv:1512.02325.

15. Brown, J. *A Gentle Introduction to the Fbeta-Measure for Machine Learning*. 2020; Available from: https://machinelearningmastery.com/fbeta-measure-for-machine-learning/.

VITA

NAME	Teepakorn Lilek
DATE OF BIRTH	18 December 1994
PLACE OF BIRTH	Ratchaburi
INSTITUTIONS ATTENDED	Kasetsart University
HOME ADDRESS	13/4 Moo 5, Bang Pa, Muang, Ratchaburi, 70000