# การระบุประเภทขององค์ประกอบหลักในสนามบินจากภาพรับรู้ระยะไกลโดยเครือข่ายที่มีประสิทธิภาพ

นางสาวพิมพิศา เจริญจิตตั้ง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

# CLASSIFICATION OF MAIN COMPONENTS IN AIRPORTS FROM REMOTE SENSING IMAGES BY EFFICIENT NETWORK

Miss Pimpisa Charoenchittang

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and

Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2020

| | |
|---|---|
| Thesis Title | CLASSIFICATION OF MAIN COMPONENTS IN AIRPORTS FROM REMOTE SENSING IMAGES BY EFFICIENT NETWORK |
| By | Miss Pimpisa Charoenchittang |
| Field of Study | Applied Mathematics and Computational Science |
| Thesis Advisor | Associate Professor Nagul Cooharojananone, Ph.D. |
| Thesis Co-advisor | Associate Professor Petarpa Boonserm, Ph.D. |

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

............................................... Dean of the Faculty of Science

(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

............................................... Chairman

(Assistant Professor Krung Sinapiromsaran, Ph.D.)

............................................... Thesis Advisor

(Associate Professor Nagul Cooharojananone, Ph.D.)

............................................... Thesis Co-advisor

(Associate Professor Petarpa Boonserm, Ph.D.)

............................................... Examiner

(Thap Panitanarak, Ph.D.)

............................................... External Examiner

(Suriya Natsupakpong, Ph.D.)

พิมพิศา เจริญจิตตั้ง : การระบุประเภทขององค์ประกอบหลักในสนามบินจากภาพรับรู้ระยะไกลโดยเครือข่ายที่มีประสิทธิภาพ. (CLASSIFICATION OF MAIN COMPONENTS IN AIRPORTS FROM REMOTE SENSING IMAGES BY EFFICIENT NETWORK) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร. นกุล คูหะโรจนานนท์, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : รศ.ดร. เพชรอาภา บุญเสริม, **??** หน้า.

ในวิทยานิพนธ์ฉบับนี้ ได้ทำการระบุประเภทขององค์ประกอบหลักในสนามบิน จากภาพรับรู้ระยะไกล ซึ่งเป็นชุดข้อมูลที่มีความน่าสนใจ เนื่องจากองค์ประกอบประเภทเดียวกันในแต่ละสนามบิน อาจมีความแตกต่างกันทั้งในด้านของรูปทรง ขนาด และสี สถาปัตยกรรมเครือข่ายที่มีประสิทธิภาพ (EfficientNet) เป็นสถาปัตยกรรมของการเรียนรู้ลึกที่ใช้ในงานวิจัยนี้ เนื่องจากมีการใช้จำนวนพารามิเตอร์น้อย และใช้เวลาในการคำนวณที่รวดเร็วกว่า เมื่อเทียบกับสถาปัตยกรรมอื่น ๆ ที่มีความแม่นยำใกล้เคียงกัน ในการทดลองจึงใช้ EfficientNet ในรุ่น B0 B1 B2 B3 และ B4 เพื่อระบุประเภทขององค์ประกอบหลักในสนามบินทั้งหมด 4 ประเภท ได้แก่ อาคารผู้โดยสาร หอบังคับการบิน ลานบิน และอาคารจอดรถ โดยการเก็บภาพสี RGB ที่มีความละเอียด 560 × 560 พิกเซล จากสนามบินทั้งหมด 322 แห่งในทวีปเอเชีย และแบ่งชุดข้อมูลสำหรับชุดการเรียนรู้ 70 เปอร์เซ็นต์ ชุดการตรวจสอบ 10 เปอร์เซ็นต์ และชุดการทดสอบ 20 เปอร์เซ็นต์ ผลการทดลองสรุปได้ว่า EfficientNet รุ่น B4 เป็นสถาปัตยกรรมที่เหมาะสมกับงานนี้ที่สุด โดยมีความแม่นยำสูงถึง 90 เปอร์เซ็นต์

| | | |
|---|---|---|
| ภาควิชา | คณิตศาสตร์และ | ลายมือชื่อนิสิต ........................ |
| | วิทยาการคอมพิวเตอร์ | ลายมือชื่อ อ.ที่ปรึกษาหลัก .............. |
| สาขาวิชา | คณิตศาสตร์ประยุกต์ | ลายมือชื่อ อ.ที่ปรึกษาร่วม .............. |
| | และวิทยาการคณนา | |
| ปีการศึกษา | 2563 | |

## 6172023023 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

KEYWORDS : AIRPORT COMPONENTS / CLASSIFICATION / EFFICIENTNET / REMOTE SENSING IMAGES

PIMPISA CHAROENCHITTANG : CLASSIFICATION OF MAIN COMPONENTS IN AIRPORTS FROM REMOTE SENSING IMAGES BY EFFICIENT NETWORK. ADVISOR : ASSOC. PROF. NAGUL COOHAROJANANONE, Ph.D., COADVISOR : ASSOC. PROF. PETARPA BOONSERM, Ph.D., **??** pp.

In this thesis, we classify the type of main components in airports from the remote sensing images. This datasets is considered as an interesting information since the same type of component may have different shape, size, and color. EfficientNet architecture is the deep learning architecture to use in this research due to the small number of parameters and computational time compared to the other architectures with similar accuracy. In our experiment, we apply the EfficientNet in versions B0, B1, B2, B3, and B4 to classify four types of components in the airport; the passenger terminal, the radio tower, the runway, and the car park. We collect the RGB format datasets of 322 airports in Asia with a resolution of 560 × 560 pixels. Then, the datasets is partitioned into 70% for the training set, 10% for the validation set, and 20% for the test set. From the experimental results, we conclude that EfficientNet-B4 is a suitable architecture for our datasets, which provides a high accuracy up to 90%.

Department : Mathematics and Computer Science ............ Student's Signature ......................
Advisor's Signature ......................
Field of Study : Applied Mathematics and Computational Science ............ Co-advisor's Signature ......................
Academic Year : 2020 ............

# ACKNOWLEDGEMENTS

It is difficult to express my gratitude to my thesis advisor, Associate Professor Dr. Nagul Cooharojananone and my co-advisor, Associate Professor Dr. Petarpa Boonserm for their enthusiasm, inspiration and great efforts to explain things clearly and simply. Throughout my thesis writing period comments, they provide sound advice, lots of good ideas and kindness. This thesis would not have been completed without them.

I further would like to thank all of my dissertation committees: Assistant Professor Dr. Krung Sinapiromsaran, Dr. Thap Panitanarak, and Dr. Suriya Natsupakpong, for their insightful comments and suggestions which motivated me to extend my research from various perspectives.

I wish to thank all of my teachers for sharing their knowledge and would like to thank all other lecturers and staff of the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University for their patience, encouragement and impressive teaching.

I am greatly indebted to my beloved parents, my grandmother, my brother and my friends for their love, support, understanding and encouragement.

Finally, I wish to thank Science Achievement Scholarship of Thailand (SAST) that supports a scholarship for my study.

Pimpisa Charoenchittang

September 2, 2021

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER I

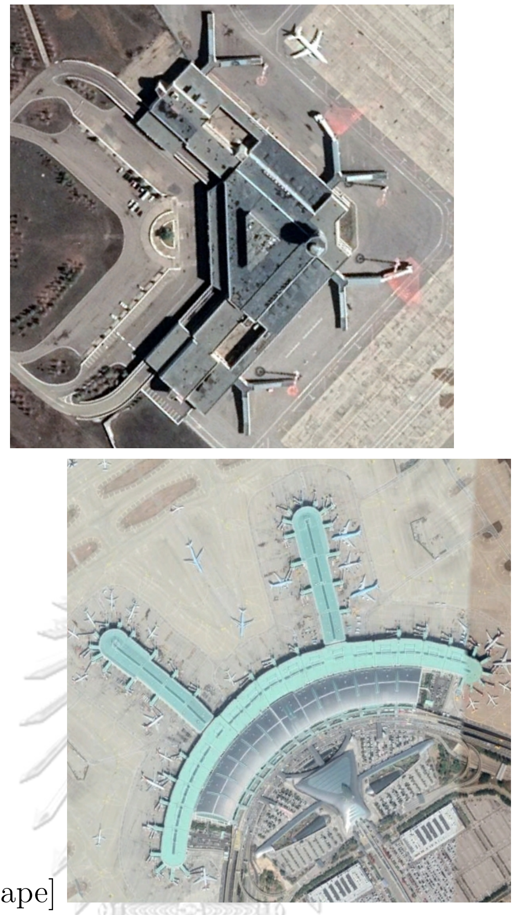
# INTRODUCTION

Deep learning was discovered and favour to solve complicated problems for decades. One problem that is important in digital age and required a lot of labour was image classification. Convolutional neural networks, which is one type of deep learning, was commonly used to solve image classification problems [?]. The airport components datasets were challenging for image classification and useful for urban planning. As aforementioned information, this is the main motivation of this thesis. Also, the objective, scopes and assumptions were described in this chapter.

## 1.1 Motivation

In recently year, there are studied about aircraft recognition [?] and also detection airport [?] and building in airport [?]. However, the images of components in airports classification from remote sensing images has hardly been studied yet. Mostly of building classification research were used 3D image datasets, which the data collection process is complicated, such as IFSAR and LIDAR images [?], high resolution stereo satellite images [?], and SAR images [?]. Meanwhile the remote sensing image datasets were easily to collect and images of main components in airports are an interesting datasets in term of applying with other types of components datasets. However, the same type of component may have a different shape or vice versa, the same shape of component may be classified as a different component type. The challenge of this problem is to classify the same types of components in each airport with the shape, size, and color differences. For instance, two passenger terminals in Figure ?? are expressed in the form of both triangular shape and semicircular shape.

Image classification is a process in computer vision that attempts to classify an image by its visual content. Recently, the number of research in this field of work has rapidly increased. Most of the early studies apply machine learning (ML) to solve this



[The triangular shape] [The semicircular shape]

**Figure 1.1:** Two passenger terminal examples with different shapes [?]

problem. However, the ML system needs to manually extract features from the input data before training a model to learn; therefore, an automatic extraction feature is required.

Deep learning (DL) is a modern learning technique which successfully classify images. The great number of DL approaches were proposed. Convolutional neural networks (CNNs) is considered as a high performance DL model in computer vision. The deep structure of CNNs encourages the model to capture and generalize filtering mechanisms by performing convolutions in the image domain, leading to its high effective features. The form of CNNs can appear in several architectures, such as Inception-v1 [?], Inception-v3 [?], ResNet-50 [?], Xception [?], Inception-v4, Inception-ResNets [?], ResNeXt-50 [?] and EfficientNet [?].

The EfficientNet concept was introduced by Tan and Le [?] in 2019. Comparing with the other architectures, EfficientNet provides higher performance for classification problem despite of its small number of parameters and computational time. With these advantages, the EfficientNet architecture is selected in this thesis.

In this thesis, we manually collect datasets of international airport components from 45 countries in Asia. Our collected images contain four types of main components in each airport including the passenger terminal, the radio tower, the runway, and the car park. Next, we classify those types of components by employing the five versions of this EfficientNet architecture, i.e., B0, B1, B2, B3, and B4. Finally, we validate and analyze all five versions of EfficientNet architecture performance.

## 1.2 Objective

To apply five versions of EfficientNet, including B0, B1, B2, B3, and B4 for classify four types of main components in airports from remote sensing images; the passenger terminal, the radio tower, the runway, and the car park, and compare their performance.

## 1.3 Scopes and Assumptions

1. The datasets are the remote sensing images of 322 international airports from 45 countries in Asia.
2. The whole components must be clearly visible from remote sensing images and contain the surrounding area.
3. The resolution of the image is 560 × 560 pixels.
4. There are four main types of components in the airport including the passenger terminal, the radio tower, the runway, and the car park.

## 1.4 Thesis Overview

The remainder of this thesis consists of the following chapters. Chapter II explains the necessary related background knowledge of this thesis. Chapter III describes the algorithms of EfficientNet architecture and the experimental setup. Chapter IV demonstrates results and discussions. The final chapter consists of the conclusions and the future work.

# CHAPTER II

# BACKGROUND KNOWLEDGE

The purpose of this chapter is to explain the necessary background knowledge containing the basic of digital image processing, the remote sensing image, the image classification problem, the convolutional neural networks, and the MobileNet architecture.

## 2.1 Basic of Digital Image Processing

Digital image processing is the technique using the computer process for dealing with digital images such as acquisition, enhancement, restoration, etc. [?]. Moreover, it also applies for object detection, object recognition, including image classification.

The image in computer vision is called a digital image. Digital image is a two dimensional (width × height) image which is partitioned the whole image into a lot of small units called pixels. It can be defined by function $f(x, y)$, where $x$ and $y$ are discrete coordinates, and the value of $f(x, y)$ is called gray level or intensity for each pixel.

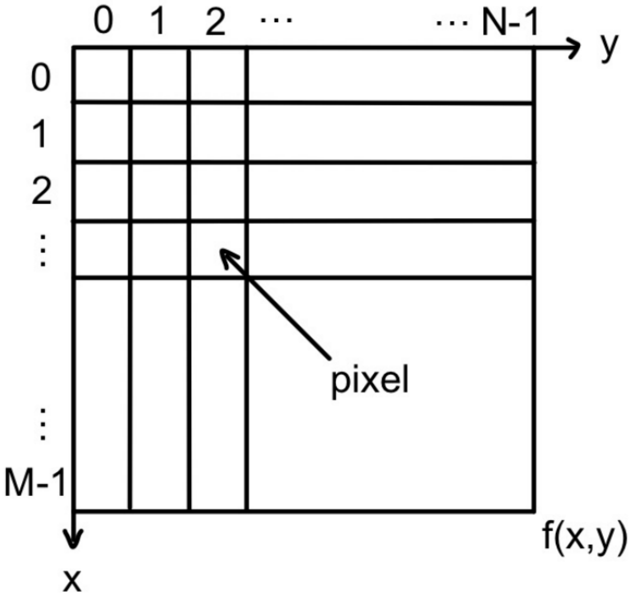
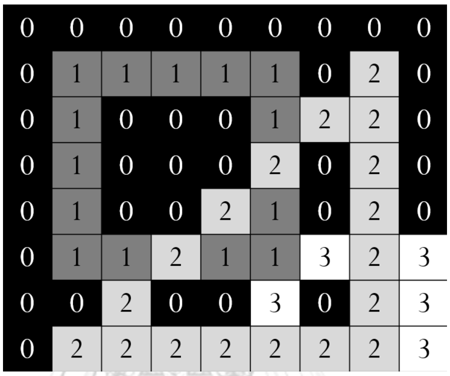


**Figure 2.1:** The two dimension image of a digital image

From Figure **??**, the matrix contains $M$ rows and $N$ columns, where $M$ and $N$ are not necessarily equal, starting with $(0,0)$ from the top left of an image. The matrix of the image is

$$F_{M,N} = \begin{pmatrix} f_{0,0} & f_{1,2} & \cdots & f_{1,N-1} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M-1,1} & f_{M-1,2} & \cdots & f_{M-1,N-1} \end{pmatrix}.$$



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 2 | 2 | 0 |
| 0 | 1 | 0 | 0 | 0 | 2 | 0 | 2 | 0 |
| 0 | 1 | 0 | 0 | 2 | 1 | 0 | 2 | 0 |
| 0 | 1 | 1 | 2 | 1 | 1 | 3 | 2 | 3 |
| 0 | 0 | 2 | 0 | 0 | 3 | 0 | 2 | 3 |
| 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 |

**Figure 2.2:** The intensity of 8 × 9 pixels

For example, Figure **??** displays the intensity of each pixel in 8 × 9 image and transformation into the matrix of image as

$$F_{8,9} = \begin{pmatrix} f_{0,0} & f_{1,2} & \cdots & f_{1,8} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,8} \\ \vdots & \vdots & \ddots & \vdots \\ f_{7,1} & f_{7,2} & \cdots & f_{7,8} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 2 & 2 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 2 & 1 & 0 & 2 & 0 \\ 0 & 1 & 1 & 2 & 1 & 1 & 3 & 2 & 3 \\ 0 & 0 & 2 & 0 & 0 & 3 & 0 & 2 & 3 \\ 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 \end{pmatrix}.$$

### 2.1.1 Grayscale Image

A grayscale image has only colors that are shades of gray. The intensity can be 1–bit, 2–bit, 4–bit, or 8–bit, etc., as shown in Figure ??. In general, every pixel consists of an 8–bit integer. The range value of intensity is 0–255.

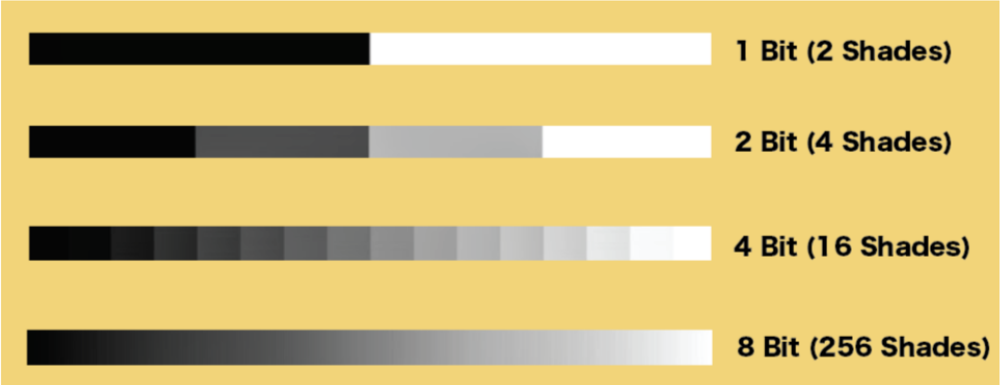
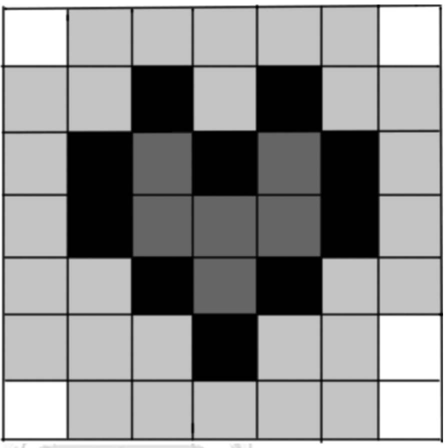


**Figure 2.3:** Color depth: bits and shades [?]

Figure ?? presents the example of the 2–bit grayscale image. The value of intensity for each pixel is in the range of 0–3 as shown in Figure ?? and Figure ?? shows the corresponding grayscale image.

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |
| 2 | 2 | 0 | 2 | 0 | 2 | 2 |
| 2 | 0 | 1 | 0 | 1 | 0 | 2 |
| 2 | 0 | 1 | 1 | 1 | 0 | 2 |
| 2 | 2 | 0 | 1 | 0 | 2 | 2 |
| 2 | 2 | 2 | 0 | 2 | 2 | 2 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |

[The intensity in each pixel]



[The grayscale image]

**Figure 2.4:** The example of the 2–bit grayscale image

Comparing the different number of bits in the same image, the results of higher number of bit shown the better sharp and more details of the image as seen in Figure **??**.

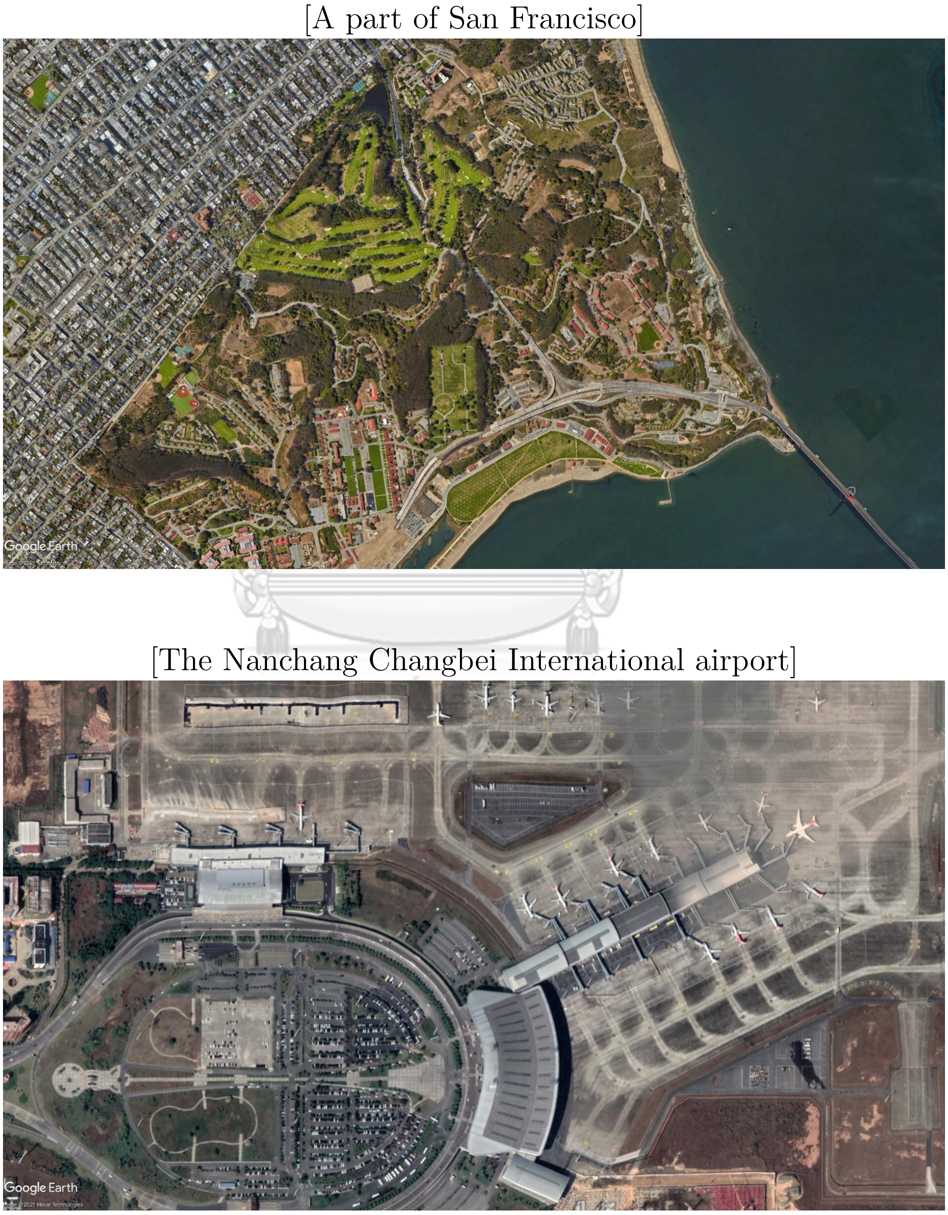
### 2.1.2 RGB Color Image

In the same number of bit, color image is more commonly used than grayscale image because colors give more details and it is used to be an important feature in image processing process. Each pixel of the color image consists of a $1 \times 3$–matrix, representing the three dimensions of color channels Red (R), Green (G), and Blue (B). For an 24–bit color image, the intensity refers to the brightness of each color channel, with 0 and 255 being the darkest and the brightest version of the primary color, respectively.

Figure **??** displays the RGB channels of the color image. From the original image in Figure **??**, it can be separated into three channels. Color representation of the red, green, and blue color channels are illustrated in Figures **??**, **??**, and **??**, respectively. The areas that have dominant tone of one color will give a high intensity of that color. For example, in Figure **??**, we can notice the high intensity of red color at the upper right of the red channel image thanks to this area from the original image has a red tone.

### 2.1.3 Remote Sensing Images

Remote sensing images are representations of a part of the earth's surface which are normally collected from a satellite or an aircraft by the process of detecting and monitoring the physical characteristics of the surface area. The following figures illustrate examples of a remote sensing image, Figure **??** is a part of San Francisco and Figure **??** is the Nanchang Changbei International Airport in China.


[A part of San Francisco]

[The Nanchang Changbei International airport]

**Figure 2.7:** The examples of remote sensing images [**?**]

## 2.2 Image Classification

Image classification is a process that tries to classify the type of an object in the input image according to its visual content. Ordinarily, the input datasets in image classification must appear only one type. For instance, an image classification process may be designed to state whether an image contains a cat as shown in Figure **??**.

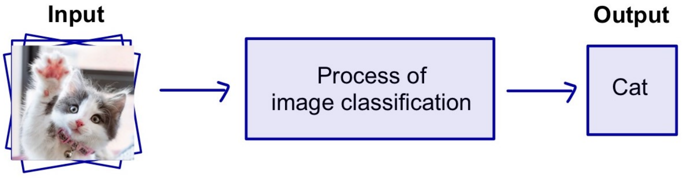


**Figure 2.8:** The process of an image classification
source: Adapted from [**?**]

While it is trivial for humans to classify an image, robust image classification is still a challenge problem in computer vision applications. However, training is a key to the success of classification [**?**].

Traditionally, ML is used to solve the image classification problem. An important process of ML is the feature extraction from an input image. The examples of notable features are the dominant pixel, color histograms, textures, and shapes. However, for new datasets the new key feature may be changed, we have to re-extract the new key feature ourselves. Because the key feature in each datasets may be different.

Thus, we need a way to automatically extract the important feature. A breakthrough in Deep Learning (DL) leading to the discovery of a Convolutional Neural Networks (CNNs) which is the automatically feature extraction.

## 2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is a form of feedforward neural networks. The design of CNNs has shown in Figure **??**. First, the convolutional and max-pooling layers extract the feature from an input image. The next part performs non-linear transformations. Finally, the fully connected layers classify images.

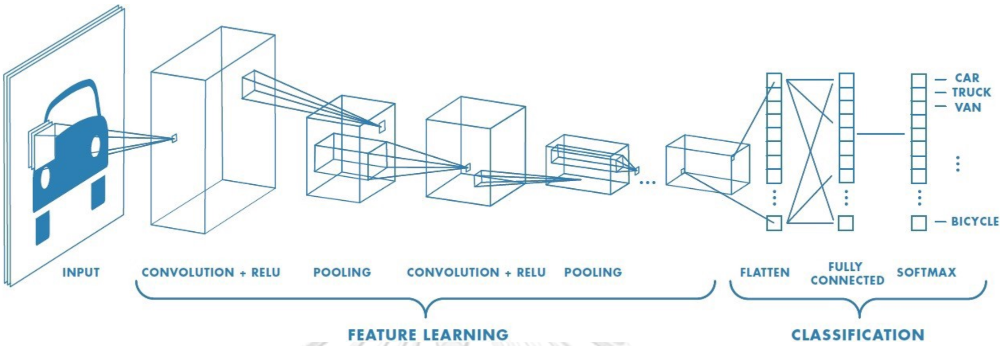
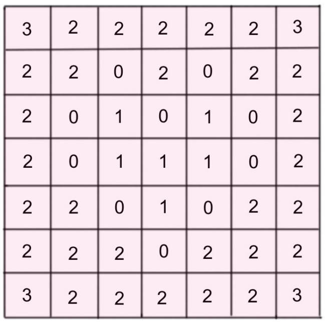


**Figure 2.9:** The process of Convolutional Neural Networks [**?**]

### 2.3.1 The Convolutional Layer

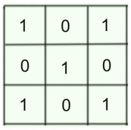
The convolutional layer (Conv layer) is the main concept of CNNs. A convolution filter, or kernel, is applied to produce a feature map from the input image. The size of the filter is depend on each architecture, typically $3 \times 3$ or $5 \times 5$ pixels.

[Input]



| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |
| 2 | 2 | 0 | 2 | 0 | 2 | 2 |
| 2 | 0 | 1 | 0 | 1 | 0 | 2 |
| 2 | 0 | 1 | 1 | 1 | 0 | 2 |
| 2 | 2 | 0 | 1 | 0 | 2 | 2 |
| 2 | 2 | 2 | 0 | 2 | 2 | 2 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |

[Filter/Kernel]



| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

**Figure 2.10:** An example of the input image and the filter

Figure **??** shows an example of the input image and the 3 × 3 filter. The matrix on the left is the input image and the matrix on the right is the convolution filter. To perform the convolution operation, the filter will slide over the input image and perform element-wise matrix multiplication. Then, the total sum will be placed in the feature map, as demonstrate in Figure **??**.

The activation function defines the output given input for the next layers. Then, the output of the convolution layer is summed with a bias term and passed through the activation function. Mostly Rectified Linear Unit (ReLU) is used because of efficient computational and non-vanishing gradient issue. The MobileNet architecture was utilised the ReLu6 activation function, which encourages the model to learn sparse features earlier than regular ReLU with low computational precision [**?**]. The graph of ReLU and RuLU6 are represent in blue and yellow colors in Figure **??**, respectively.

### 2.3.2 The Pooling Layers

The pooling layers are mostly used process for reducing the spatial size; width and height. This process can be reduced the number of parameters, hence computation requirement was decreased and this benefit to avoid overfitting.

The most common form of pooling layer is the max-pooling $2 \times 2$ in figure **??**. The stride, which mean the number of pixels shifts over the input matrix from convolutional layer at a time, of $2 \times 2$ pixels were used to filter and selected the maximum intensity in each submatrix as representative [?]. For instance, the $2 \times 2$ pixels in the upper left of original image have the value 7, 2, 4, and 5, here, the maximum value is "7" which will be selected to the value in the upper left of the output image.

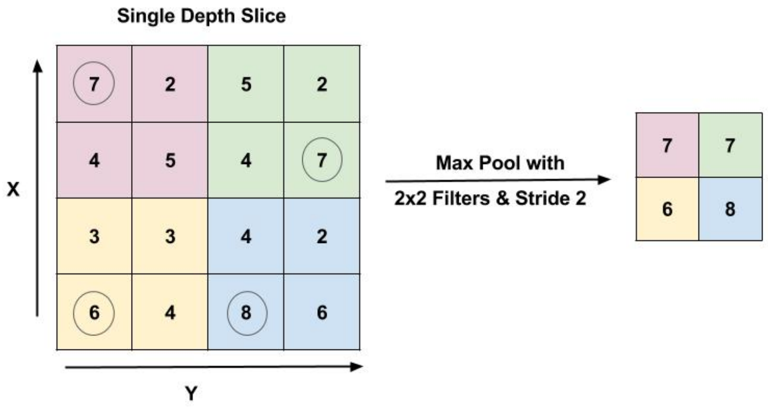


**Figure 2.13:** Max pooling layer with the filter size 2×2 and stride 2 [?]

### 2.3.3 Fully Connected Layers

After the convolution and pooling layers, we add fully connected layers to classify the class of input images. Since the output of both convolution and pooling layers are 3D structure, but a fully connected layer expects a 1D vector of numbers. Therefore, we flatten the output of the final pooling layer to a vector and that becomes the input to the fully connected layer, as shown in the following figure [?].

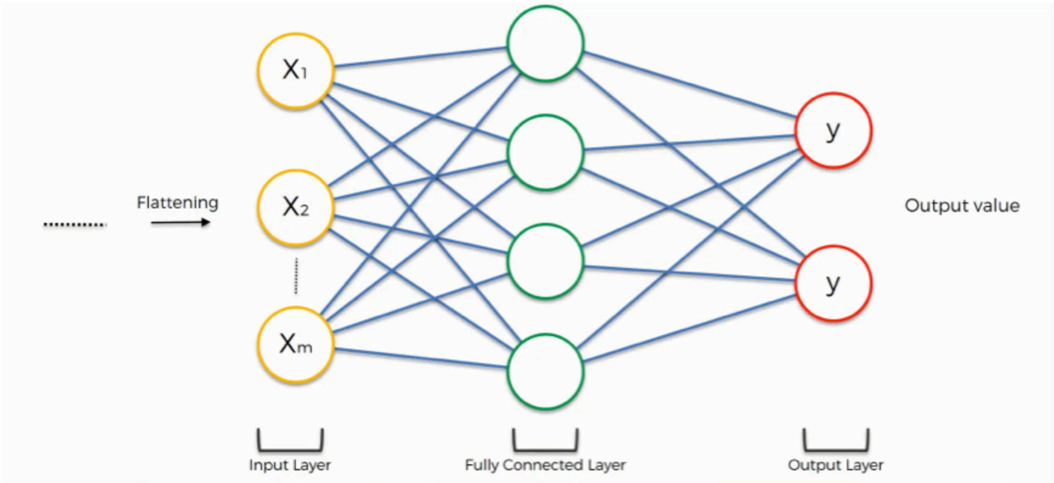


**Figure 2.14:** The process of fully connected layers [?]

The timeline of CNNs architectures are illustrated in Figure **??**. The following section will explain the MobileNet architecture, which is the main building block of the EfficientNet architecture that used in this thesis. The algorithm of EfficientNet will be described in Chapter III.

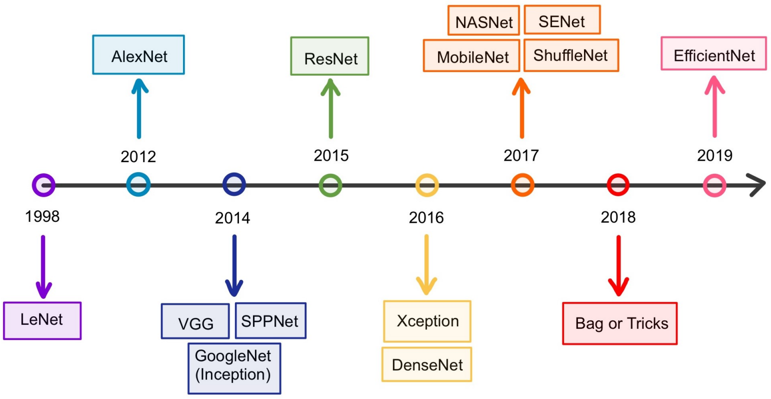


**Figure 2.15:** The timeline of CNNs architectures
source: Adapted from [?]

## 2.4 MobileNet Architecture

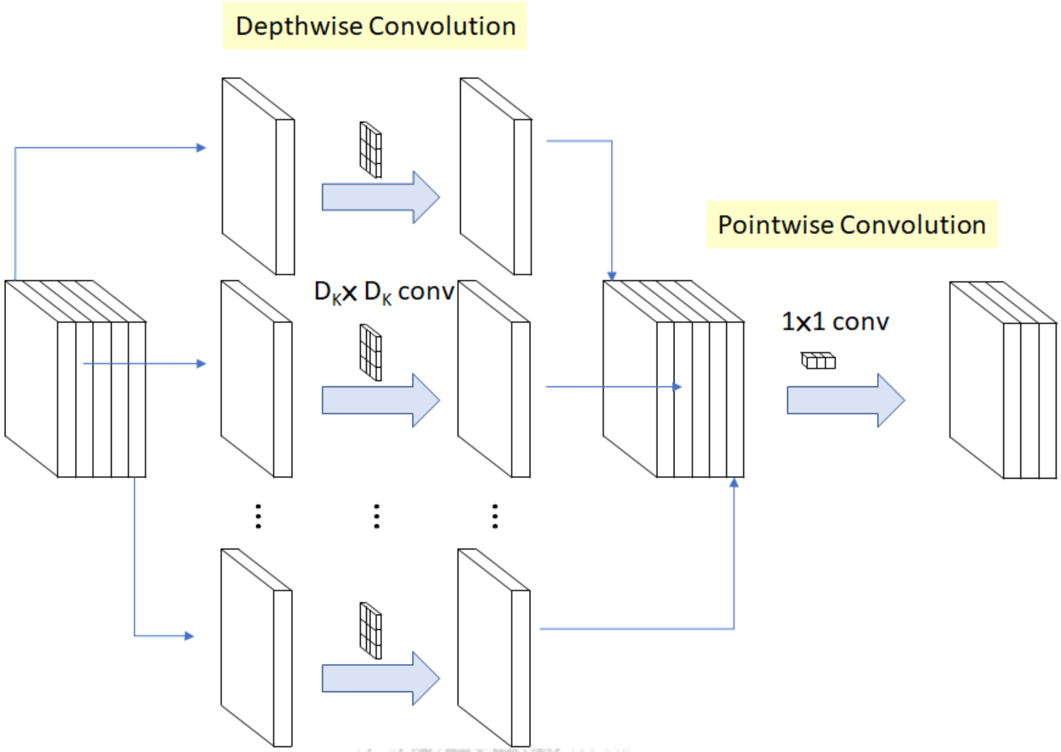


**Figure 2.16:** The Depthwise Separable Convolution [?]

The MobileNet architecture creates for using in mobile phone applications since they have the lightweight network. The main concept is applying the depthwise separable convolution to reduce the number of parameters. The depthwise separable convolution is the combination of depthwise (Dwise) and pointwise convolution as shown in Figure ??. The Dwise is the channels $D_K \times D_K$ spatial convolution which is a map of a single convolution on each input channel separately. Therefore, the number of output and input channels are similar. Then, the pointwise convolution combines the features created by the Dwise by using Conv1×1 [?].

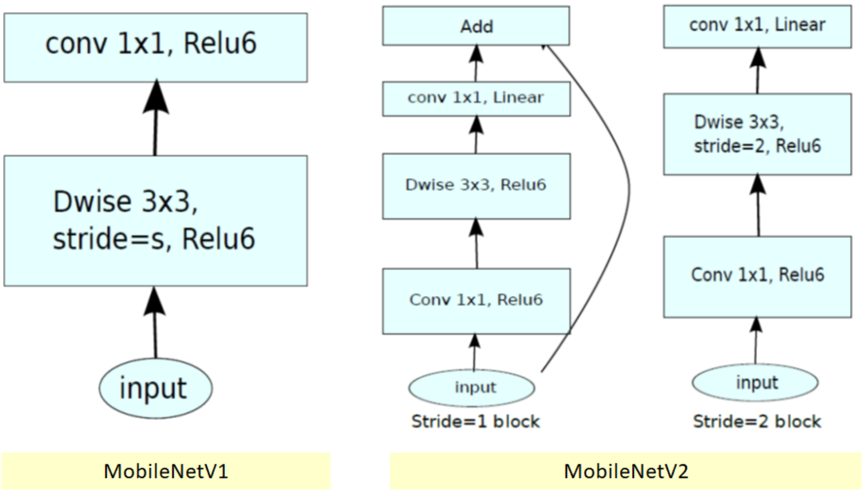
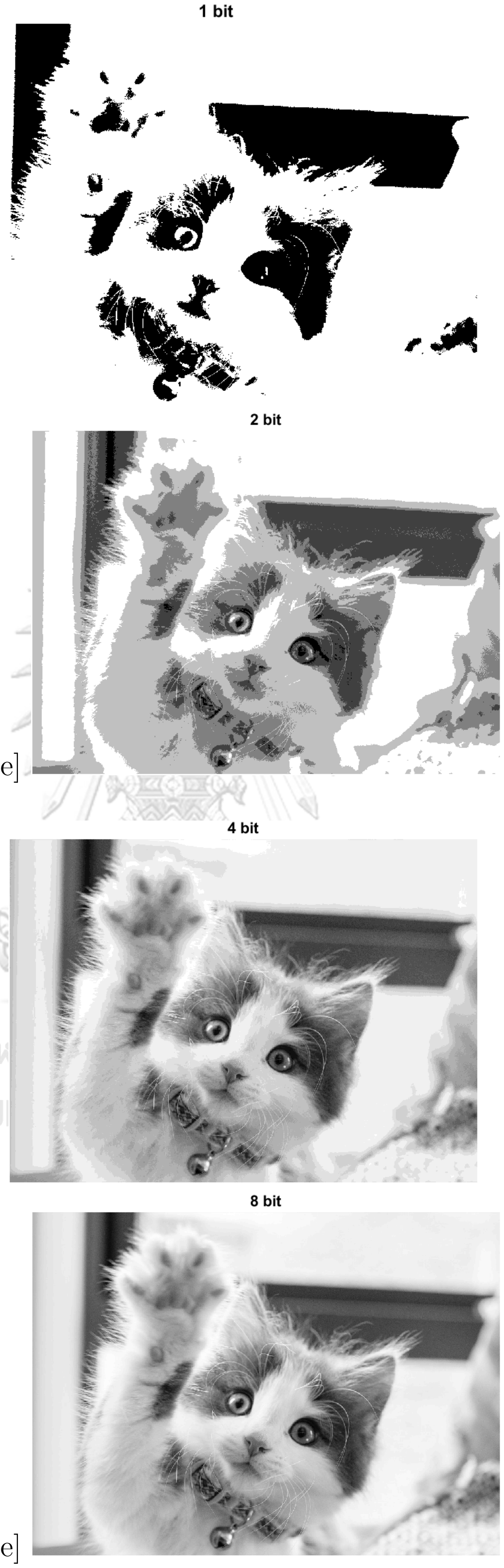


**Figure 2.17:** The algorithm of MobileNet V1–V2 [?]

Figure ?? has shown the difference between two versions of MobileNet, V1 and V2. MobileNet–V1 have a depthwise convolution in the first layer. Then, the second layer is the pointwise convolution with Conv1×1. Finally, ReLU6 is used due to its robustness when used with low-precision computation. On the other hand, there are two types of blocks in MobileNet–V2 as the residual block with stride of 1 and 2, which we called the stride of 1 as inverted residual block (MBConv). There are three layers for both blocks, namely Conv1×1 with ReLU6, depthwise convolution, and Conv1×1.

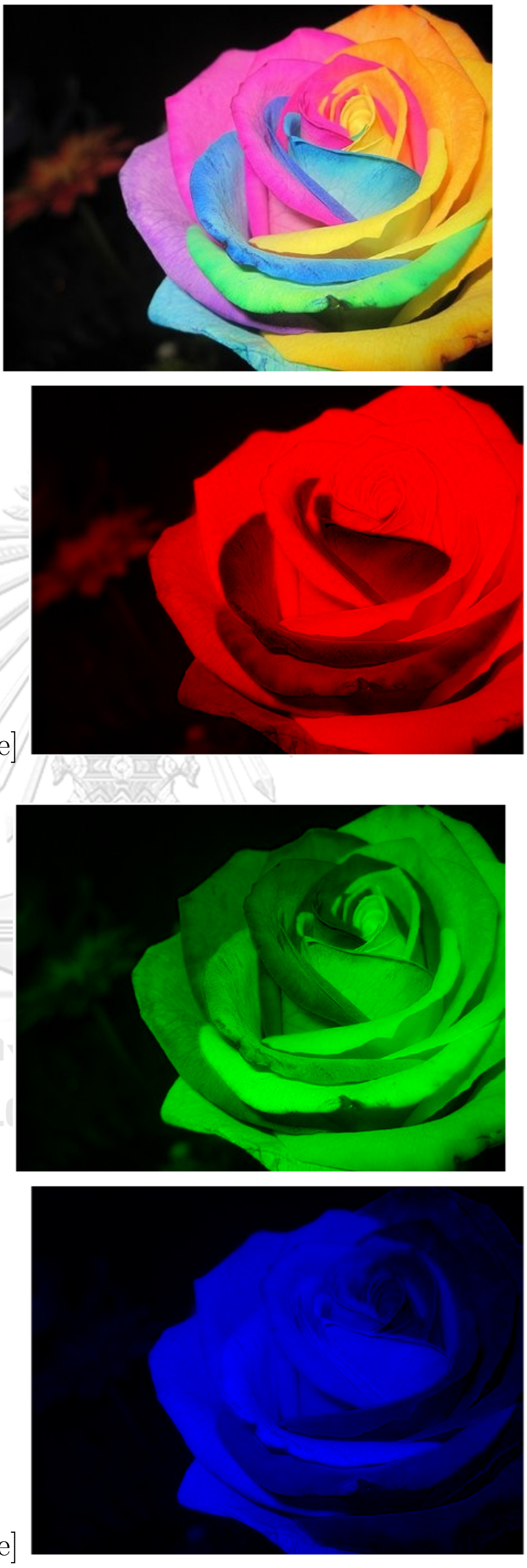
According to aforementioned knowledge, the EffcientNet was the lastest CNNs architecture published in 2019 and seem to be great efficiency for the image classification [?]. Along with complication of airport components datasets form remote sensing in RGB color image with 24–bit, the EfficientNet was apply to classify airport components.



[1–bit grayscale image] [2–bit grayscale image]

[4–bit grayscale image] [8–bit grayscale image]

**Figure 2.5:** Grayscale images in 1, 2, 4, and 8 bits
source: Adapted from [**?**]



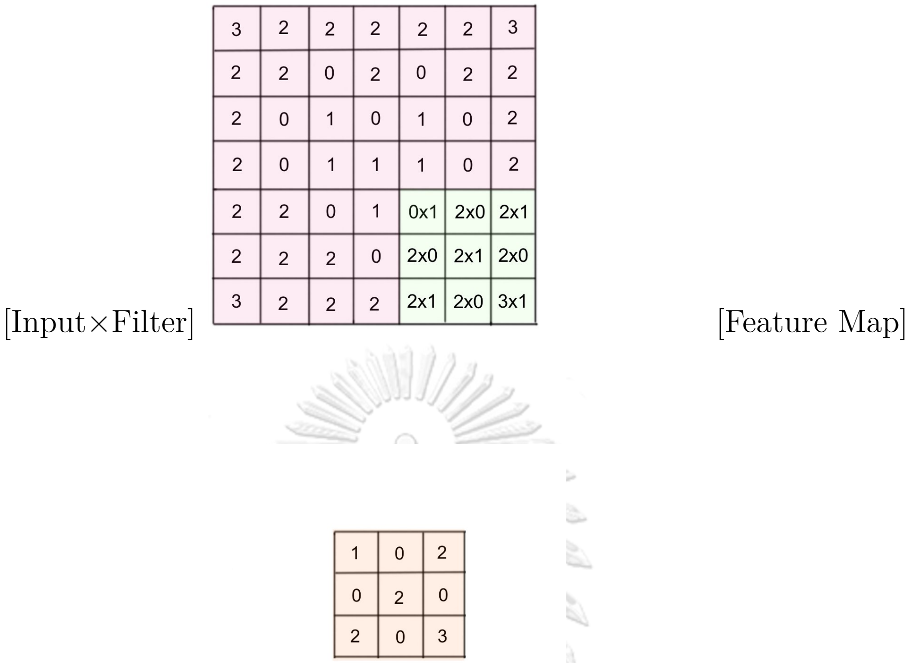
[The original image [?]]

[The red channel image]

[The green channel image]

[The blue channel image]

**Figure 2.6:** The RGB channels of the color image



|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |
| 2 | 2 | 0 | 2 | 0 | 2 | 2 |
| 2 | 0 | 1 | 0 | 1 | 0 | 2 |
| 2 | 0 | 1 | 1 | 1 | 0 | 2 |
| 2 | 2 | 0 | 1 | 0x1 | 2x0 | 2x1 |
| 2 | 2 | 2 | 0 | 2x0 | 2x1 | 2x0 |
| 3 | 2 | 2 | 2 | 2x1 | 2x0 | 3x1 |

|  |  |  |
|---|---|---|
| 1 | 0 | 2 |
| 0 | 2 | 0 |
| 2 | 0 | 3 |

**Figure 2.11:** An example of the feature map generation

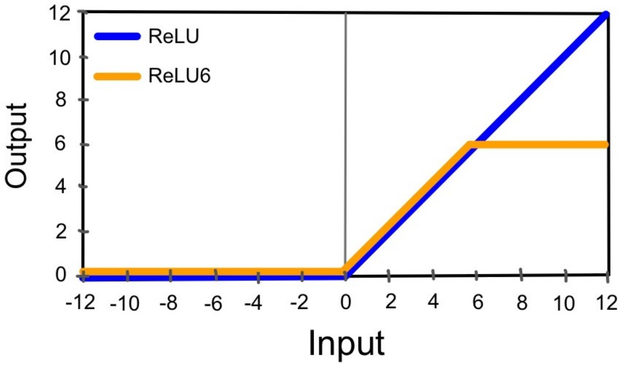


**Figure 2.12:** ReLU and ReLU6 activation function
source: Adapted from [?]

# CHAPTER III

# METHODOLOGY AND EXPERIMENTS

This chapter explains the details of the datasets and the algorithms of the EfficientNet architecture in five versions B0, B1, B2, B3, and B4. Moreover, the experimental setup, and the evaluation metrics are described.

## 3.1 Datasets

Our datasets is gathered by Google Earth Pro in the type of remote sensing image, from 45 countries in Asia (322 international airports). All images, which are RGB color format with a resolution of $560 \times 560$ pixels, were manually collected. They can be divided into four types, consisting image of the passenger terminal, the radio tower, the runway, and the car park. Some examples were displayed in Figure **??**.

## 3.2 EfficientNet Architecture

After the process of data preparation, we will apply the EfficientNet architecture. The EfficientNet is a CNN architecture to solve the image classification problem.

Most of the previous CNN architecture use only single factor of three basic factors to improve the accuracy and efficiency. The first factor is width scaling, when the width is the number of channels in the fully connected layer. The next factor is depth scaling, when the depth is the number of convolutional layers. And the last one is resolution scaling. Figure **??** (a) is a baseline network example; (b)–(d) are conventional scaling that only increases one dimension of the network width, depth, or resolution. The main idea of the EfficientNet architecture is to use all of three factors; width, depth, and resolution with a fixed ratio, namely compound scaling method as shown in Figure **??** (e).

Conventionally, MBConv of MobileNet–V2 was utilized to generate the algorithm of EfficientNet–B0, as mentioned in Chapter II. The work flow of EfficientNet–B0 was shown in Figure **??**. It consisting of the input layer, the Conv3×3 layer, the MBConv layer, the Conv1×1 layer, the pooling layer and the fully connected layer. Lastly, the predicted type of the input image is obtained as the output.

**Table 3.1:** The details of EfficientNet B1–B4 [**?**]

| Operator | EfficientNet-B1 | | | EfficientNet-B2 | | | EfficientNet-B3 | | | EfficientNet-B4 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Resolution | Width | Depth | Resolution | Width | Depth | Resolution | Width | Depth | Resolution | Width | Depth |
| Conv3×3 | 240×240 | 32 | 1 | 260×260 | 32 | 1 | 300×300 | 40 | 1 | 380×380 | 48 | 1 |
| MBConv1, k3×3 | 120×120 | 16 | 2 | 130×130 | 16 | 2 | 150×150 | 24 | 2 | 190×190 | 24 | 2 |
| MBConv6, k3×3 | 60×60 | 24 | 3 | 130×130 | 24 | 3 | 150×150 | 32 | 3 | 190×190 | 32 | 4 |
| MBConv6, k5×5 | 30×30 | 40 | 3 | 65×65 | 48 | 3 | 75×75 | 48 | 3 | 95×95 | 56 | 4 |
| MBConv6, k3×3 | 15×15 | 80 | 4 | 33×33 | 88 | 4 | 38×38 | 96 | 5 | 48×48 | 112 | 6 |
| MBConv6, k5×5 | 15×15 | 112 | 4 | 17×17 | 120 | 4 | 19×19 | 136 | 5 | 24×24 | 160 | 6 |
| MBConv6, k5×5 | 8×8 | 192 | 5 | 17×17 | 208 | 5 | 19×19 | 232 | 6 | 24×24 | 272 | 8 |
| MBConv6, k3×3 | 8×8 | 320 | 2 | 9×9 | 352 | 2 | 10×10 | 384 | 2 | 12×12 | 448 | 2 |

EfficientNet–B0 was scaling up by the compound scaling method to get better performance. In general, there are four versions of EfficientNet B1–B4. The comparison of each version was illustrated in Table **??**. The steps of each architecture are clearly shown in Figure **??**. The EfficientNet computational tool is based on the tensorflow library with python3, the code is provided by P. Yakubovskiy [**?**].

## 3.3 Experimental setup

The numbers of batch size and epoch were set up to 12 and 50, respectively. Then, the operation was run on Google Colaboratory with GPU. First, the input datasets were divided into three parts following 70% of training, 10% of validation, and 20% of test set. To generate the best performance model of EfficientNet B0–B4 the training set was used to train each model. After training the model, the validation set used to determine whether the model is able to perform on unknown images. Finally, we measured the performance of the model obtained from both of two previous datasets by test set as shown in Figure **??**.

## 3.4 Evaluation Metrics

The performance metrics consist of Accuracy, Precision, Recall, and F1–Score are used to evaluate the models. In Table **??**, positive type is the type of considered component and negative type includes all other components. For example, if we consider the passenger terminal image, positive type is all images of the passenger terminal and negative type includes all images of the radio towers, the runways and the car parks.

**Table 3.2:** Confusion matrix

| | | **Actual** | **type** |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Predicted** | **Positive** | True Positive<br>(TP) | False Positive<br>(FP) |
| **type** | **Negative** | False Negative<br>(FN) | True Negative<br>(TN) |

The performance metrics are defined as follows. First, Accuracy is a ratio of the correctly predicted samples for each type to the total samples which is the summation of diagonal in the confusion matrix, defined by

$$\text{Accuracy} \;=\; \frac{\sum \text{TP for each type}}{\text{Total Samples}}.$$

Next, Precision is the ratio of the unmistakable predicted types to the total predicted positive types, computed by

$$\text{Precision} \;=\; \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall is the ratio of the correctly predicted positive types to all the samples in the actual positive type

$$\text{Recall} \;=\; \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Lastly, F1–Score is the harmonic mean of Precision and Recall which calculated by

$$\text{F1–Score} \;=\; 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

To clarify, we demonstrate a calculation of Accuracy, Precision, Recall, and F1–Score, by using the information from the confusion matrix that summarizes the results of classifier as shown in Table **??**. The following table shows the number of predicted and the actual type for each component type.

**Table 3.3:** The example of predicted result

| | | **Actual type** | | | |
|---|---|---|---|---|---|
| | | **Terminal** | **Radio tower** | **Runway** | **Car park** |
| **Predicted type** | **Terminal** | 132 | 11 | 1 | 15 |
| | **Radio tower** | 4 | 81 | 1 | 4 |
| | **Runway** | 8 | 7 | 122 | 6 |
| | **Car park** | 1 | 2 | 1 | 150 |

If we consider the passenger terminal type to be a positive type, then the others are negative types. We can calculate the value of true positive (TP), false positive (FP), false negative (FN), and true negative (TN) as follows; TP = 132, FP = $11+1+15 = 27$, FN = $4+8+1 = 13$, and TN = $81+122+150 = 374$, which are displayed in Table **??**.

**Table 3.4:** Confusion matrix for passenger terminal type

| | | **Actual** | **type** |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Predicted** | **Positive** | 132 | 13 |
| **type** | **Negative** | 27 | 374 |

Next, we can evaluate all of the performance metrics with the values from Table **??** as follow;

$$\begin{aligned}
\text{Accuracy} &= \frac{132+81+122+374}{546} &= \frac{485}{546} &= 0.89,\\
\text{Precision} &= \frac{132}{132+27} &= \frac{132}{159} &= 0.83,\\
\text{Recall} &= \frac{132}{132+13} &= \frac{132}{145} &= 0.91,\\
\text{F1–Score} &= 2\times\frac{0.83\times 0.91}{0.83+0.91} &= 2\times\frac{0.7553}{1.74} &= 0.87.
\end{aligned}$$

From all of performance measures, Accuracy indicates the correctness of model and F1–Score is the effective tool which is weighting between Precision and Recall. Then, Accuracy and F1–Score are selected as the significant metrics to decide which is the most suitable model for our datasets. In conclusion, we have already explained the datasets, methodology and the performance metrics which will be applied in our experiment and the result will be shown in the next section.

[The passenger terminal]

[The radio tower]

[The runway]

[The car park]

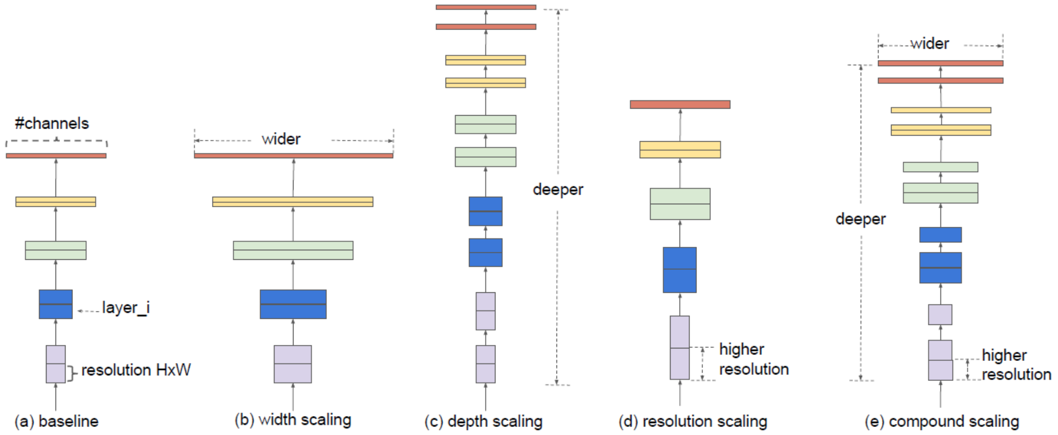


**Figure 3.2:** The difference between EfficientNet scaling and conventional methods [?]

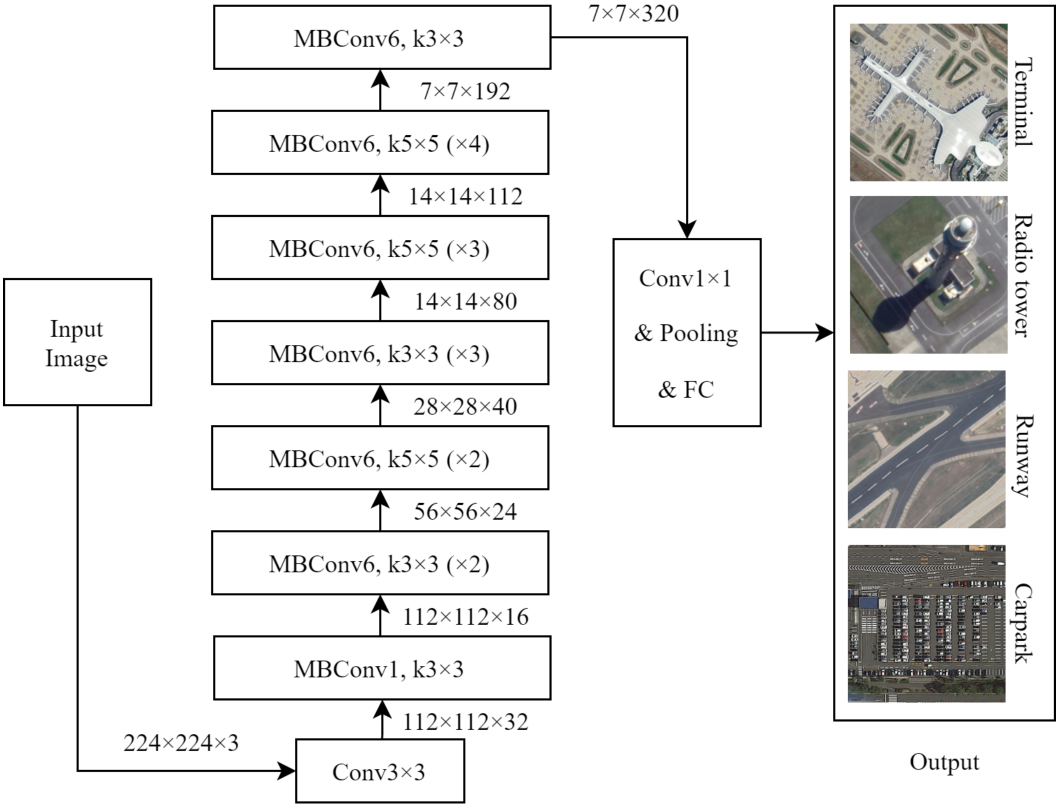


**Figure 3.3:** The work flow of EfficientNet–B0 [?]

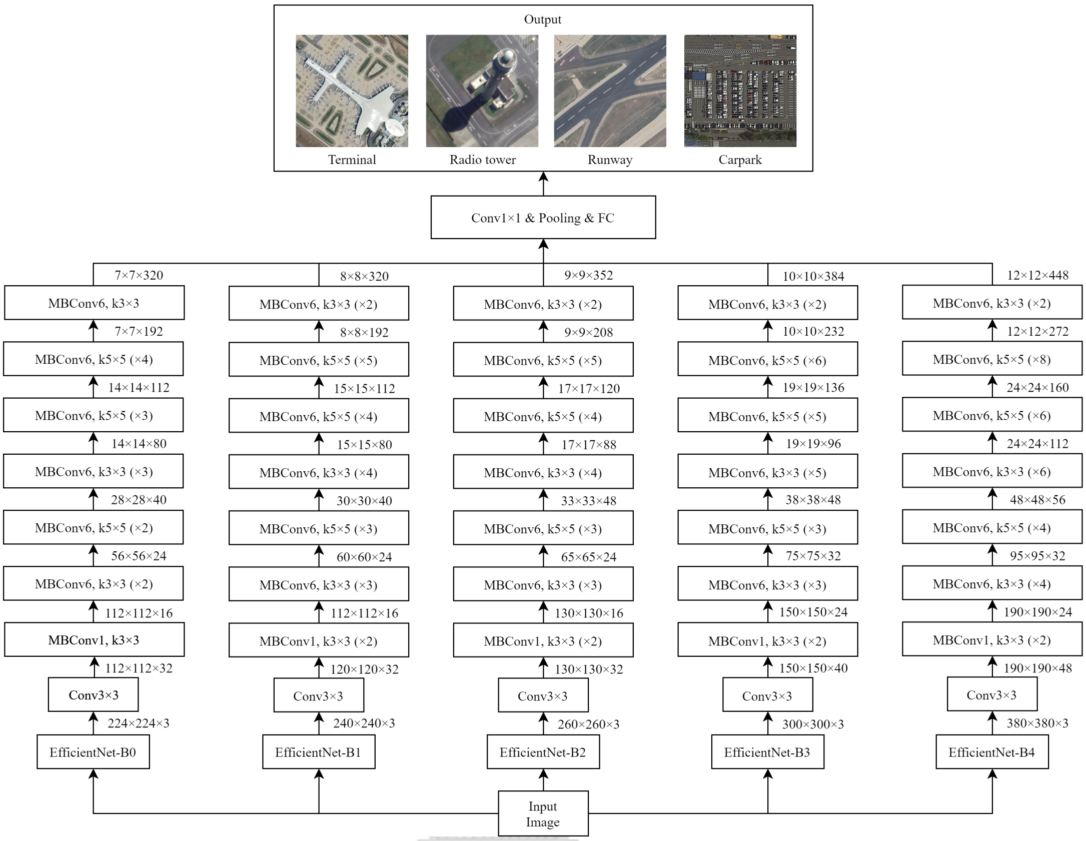


**Figure 3.4:** The work flow of EfficientNet B0–B4
source: Adapted from [?]

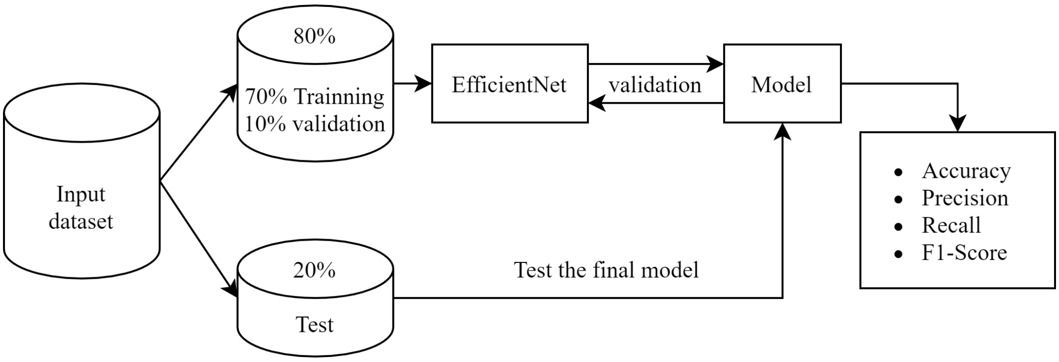


**Figure 3.5:** Schematic of the training, validation and test process [?]

# CHAPTER IV

# RESULTS AND DISCUSSION

In this chapter, we present the result of our collected datasets, the classification performance of EfficientNet B0–B4 and discuss the outcome.

## 4.1 Collected Datasets

The collected datasets has 2, 720 images including four types of main components, which details of each type were provided in Table **??**. We randomly split the datasets into 70%, 10%, and 20% percentages for training, validation, and test sets, respectively.

**Table 4.1:** The details of our datasets [**?**]

| Class | Terminal | Radio tower | Runway | Car park | Total |
|---|---|---|---|---|---|
| Training set | 487 | 340 | 421 | 586 | 1,834 |
| Validation set | 90 | 63 | 78 | 109 | 340 |
| Test set | 145 | 101 | 125 | 175 | 546 |
| Total | 722 | 504 | 624 | 870 | 2,720 |

## 4.2 Result

First of all, the training and validation set were considered. For each epoch, the training and validation sets are applied in the EfficientNet B0–B4. Then, the accuracy and loss value are evaluated as shown in Figures **??** and **??**. The x-axis is the number of epoch, and the y-axis in Figures **??** and **??** are the accuracy and loss, respectively. The blue line represents the accuracy of the training set and the pink line represents the accuracy of the validation set.

We can observe from Figure ?? that in the higher version of EfficientNet, the blue and pink lines are closer to each other. Thus, the approach of the training and validation accuracy indicates that the model with the EfficentNet–B4 architecture has a tendency to perform on unknown images as well as a test set.

Furthermore, from the graph of training and validation loss in Figure ??, the differences of the training loss between each epoch in EffientNetB0–B4 are not deceasing after training more than 40 epochs. Thus, the large number of training epochs is not necessary.

Next, we compare among EfficientNet B0–B4 using the test set. The evaluated results are illustrated in Table **??**. The statistic results, consist of Precision, Recall, F1–Score, and Accuracy, were calculated for four component types and five model versions as follows;

**Table 4.2:** The statistic results from the test set [**?**]

| **Components** | **Indicators** | **EfficientNet** | | | | |
|---|---|---|---|---|---|---|
| | | B0 | B1 | B2 | B3 | B4 |
| Terminal | Precision | 0.92 | **0.96** | 0.76 | 0.88 | 0.92 |
| | Recall | 0.63 | 0.89 | **0.92** | 0.79 | 0.90 |
| | F1-Score | 0.75 | 0.78 | 0.83 | 0.84 | **0.91** |
| Radio tower | Precision | 0.81 | 0.71 | **0.92** | 0.73 | **0.92** |
| | Recall | 0.46 | 0.81 | 0.60 | **0.87** | 0.72 |
| | F1-Score | 0.58 | 0.76 | 0.73 | 0.79 | **0.81** |
| Runway | Precision | **0.99** | 0.91 | 0.86 | 0.98 | 0.81 |
| | Recall | 0.83 | 0.79 | 0.86 | 0.86 | **0.98** |
| | F1-Score | 0.90 | 0.85 | 0.86 | **0.92** | 0.89 |
| Car park | Precision | 0.61 | **0.96** | 0.86 | 0.88 | 0.95 |
| | Recall | **0.98** | 0.73 | 0.90 | 0.93 | 0.94 |
| | F1-Score | 0.75 | 0.83 | 0.88 | 0.91 | **0.94** |
| **Accuracy** | | 0.76 | 0.80 | 0.84 | 0.87 | **0.90** |
| **Time (s/epoch)** | | 22.93 | 34.79 | 44.79 | 50.06 | 93.89 |
| **Total parameters** | | 4,054,695 | 6,580,363 | 7,774,205 | 10,789,683 | 17,680,995 |

**Passenger terminal:** The highest Recall was occurred EfficientNet–B2 at 92%, while the highest Precision was up to 96% in EfficientNet–B1. However, EfficientNet–B4 acquired the highest F1–Score at 91%.

**Radio tower:** The Precision of EfficientNet B2 and B4 provided the highest value at 92%. Besides, EfficientNet–B3 showed the highest Recall at 87%, and the 81% highest F1–Score was obtained by EfficientNet–B4.

**Runway:** This component type has the best classification result, which performed the highest Precision at 99%, Recall at 98%, and F1–Score at 92%, from EfficientNet B0, B4, and B3, respectively.

**Car park:** The Recall score reach to 98% in EfficientNet–B0, while the highest Precision was happened in EfficientNet–B1 at 96%. However, EfficientNet–B4 gave the highest F1–Score at 94%.

Additionally, the Accuracy of EfficientNet–B4 overcomes the other lower versions EfficientNet with the value of 90%. However, almost twice of computational time and total parameters of EfficientNet–B4 per epoch were increased from EfficientNet–B3. Similarly, the computational time and total parameters of other versions tend to increase when the accuracy and version are higher.

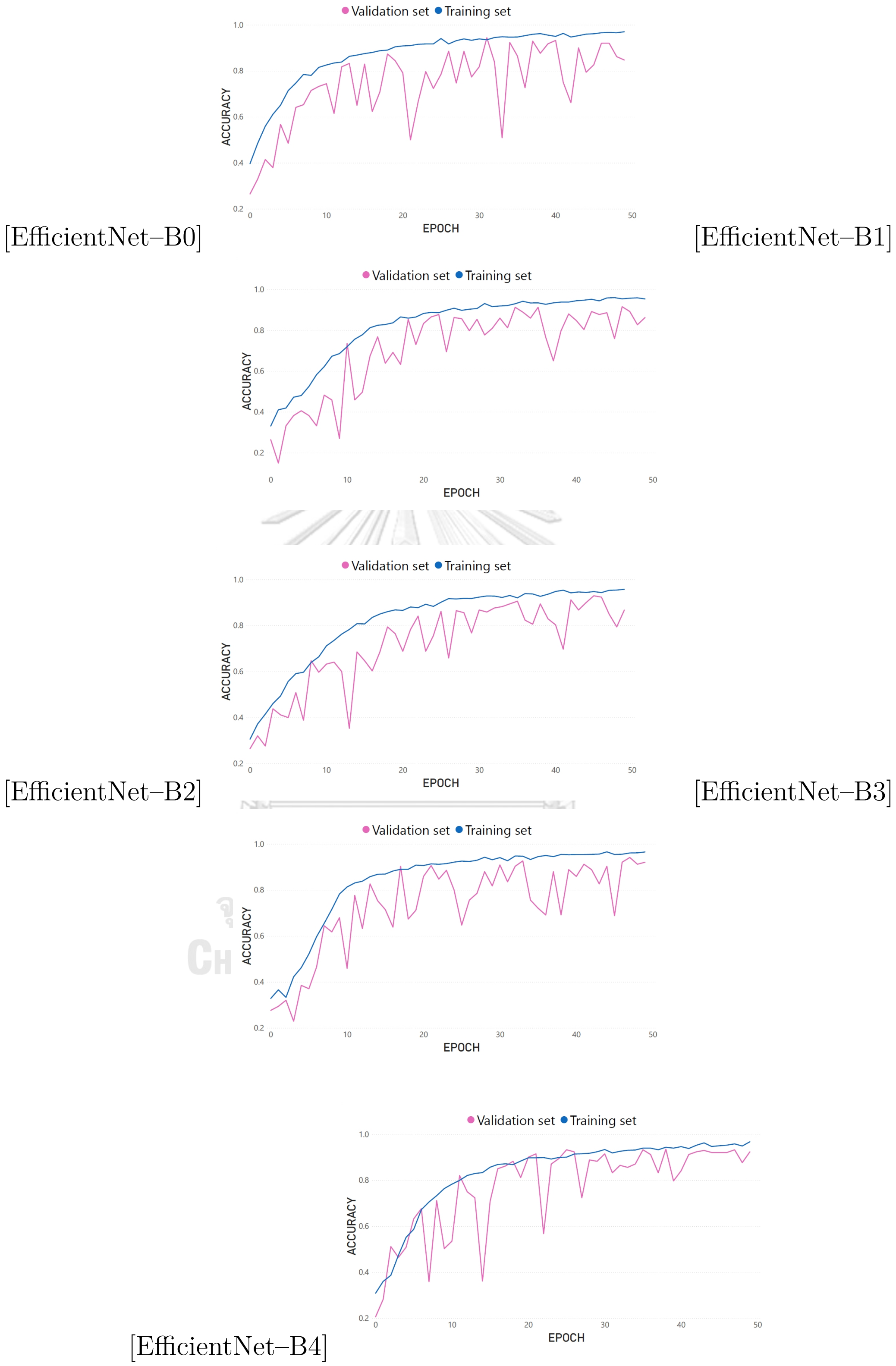
## 4.3 Discussion

From Table **??**, Precision, Recall and F1–Score are performed. Most of the models with higher Precision have lower Recall. F1–Score is the most effective measurement to balance between the number of Precision and Recall. Comparing among EfficientNet B0–B4, EfficientNet–B4 is the best architecture for classifying the passenger terminal, the radio tower, and the car park, while EfficientNet–B3 is the best architecture for the runway classification. In addition, we can also confirm from the Accuracy that EfficientNet–B4 is an appropriate architecture for our datasets.

In fact, the EfficientNet architecture has been developed continuously to the higher versions for more accuracy, i.e., B5–B7, see [**?**], but they required immensely computational requirement. Furthermore, only 3.5 percentage of accuracy in EfficientNet–B4 was increased from EfficientNet–B3, it extremely consumes the average computational time and total parameters. Accordingly, EfficientNet–B4 seems to be the most suitable architecture for component classifications in this thesis.

To focus on the datasets, EfficientNet B0–B4 displayed some limitations. The inaccurate classifications appear when the components have similar shapes, or when the background color or objects are similar to the color of the target components. For example, the radio tower displayed in Figure **??** has been classified as the passenger terminal. The reason might be due to the complexity of the components. To fix this problem, the datasets may be specified for some type of component or only some parts of that component.

Another concern is that the model cannot detect and classify the specific characteristics of each component type as shown as example in Figure **??**. The jet bridge, which is the main characteristic of the passenger terminal, are clearly visible but model has been wrongly classified as the radio tower. The cause is that the background and the component colors are relatively similar. This issue may be discussed and developed in the future work.



[EfficientNet–B0]

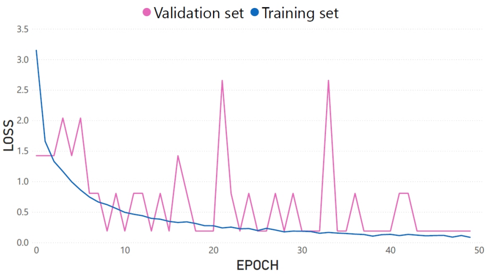
[EfficientNet–B1]

[EfficientNet–B2]

[EfficientNet–B3]

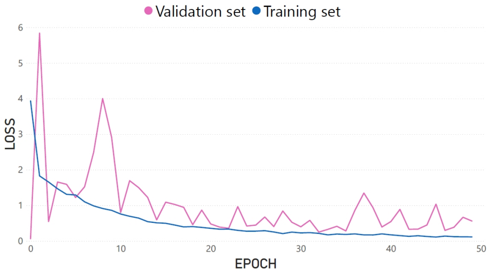
[EfficientNet–B4]

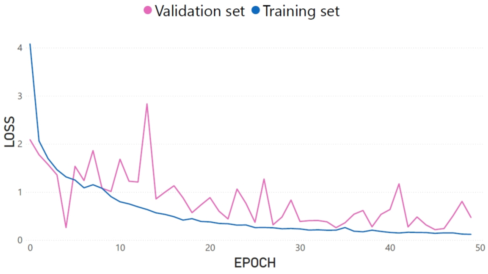
**Figure 4.1:** The graph of accuracy in each version of EfficientNet B0–B4



[EfficientNet–B0]

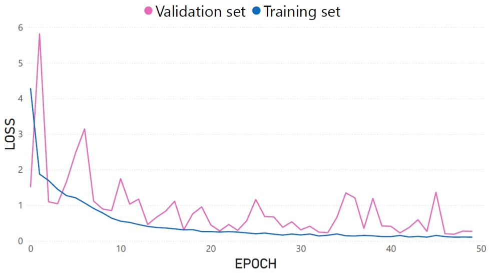
[EfficientNet–B1]

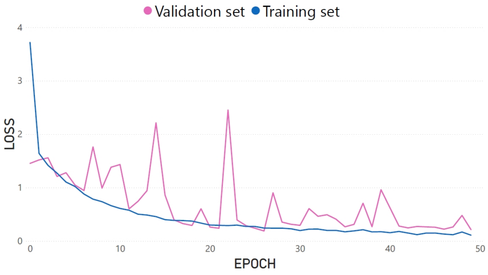




[EfficientNet–B2]

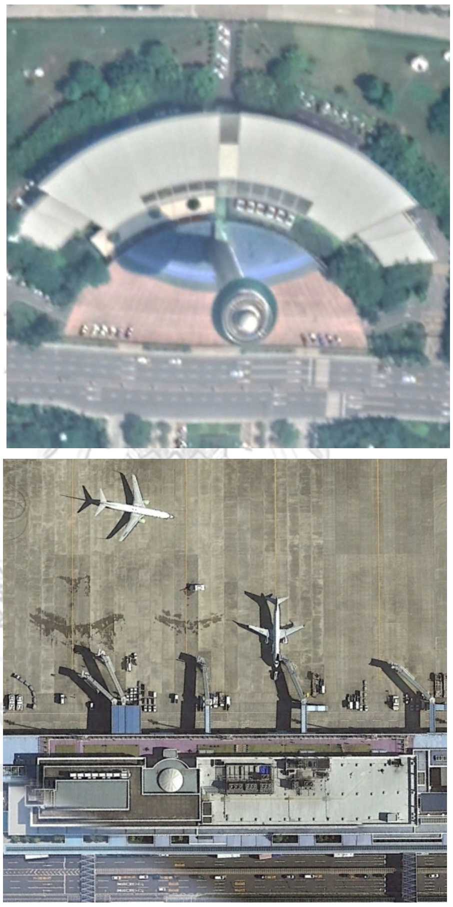
[EfficientNet–B3]





[EfficientNet–B4]

**Figure 4.2:** The graph of loss in each version of EfficientNet B0–B4



[The radio tower [?]]

[The terminal [?]]

**Figure 4.3:** The examples of the failure classification

# CHAPTER V

# CONCLUSION AND FUTURE WORK

In the last chapter, we will conclude over all of this thesis and introduce some idea to solve the weakness of the model for future work.

## 5.1 Conclusion

In Chapter I, we introduced the strategies to solve the image classification problem, i.e., ML and DL. With its effectiveness, we are interested in convolutional neural networks (CNNs) which is a type of DL. In this thesis, the goal is to classify four types of main components in airports from remote sensing images; the passenger terminal, the radio tower, the runway, and the car park using the EfficientNet architecture versions B0, B1, B2, B3, and B4 and compare their performance. In Chapter II, we mentioned the background knowledge of the digital image processing, the remote sensing image, the image classification problem, the convolutional neural networks, and the MobileNet architecture. The MobileNet–V2 contains two necessary blocks, convolutional layer (Conv) and inverted residual block (MBConv), which are the initiate of EfficientNet architecture. Next, the details of EfficientNet are described in Chapter III. EfficientNet has been purposed for the compound scaling method with the great efficiency to balance network in term of depth, width, and resolution. Moreover, we also explained our datasets, the experimental setup and the evaluation metrics in this chapter. The datasets is manually collected from Google Earth Pro in the form of $560 \times 560$ pixels RGB color remote sensing image, which cover 322 international airports from 45 countries in Asia. Then, 70 percent of all images in datasets were performed as the training set and another 20 percent for testing through EfficienctNet B0–B4. Moreover, for validation of each version of EfficientNet were performed through a partition of 10 percent of the datasets. In addition, we evaluate the performance metrics of each model consisting of Accuracy, Precision, Recall, and F1–Score. In Chapter IV, we exhibit the collected result of our $2,720$ images datasets including four types of main components, which are 722 passenger terminals, 504 radio towers, 624 runways, and 870 car parks. Then, the accuracy and loss value of the training and validation sets are evaluated for each epoch of EfficientNet B0–B4. The highest F1–Score demonstrates that EfficientNet–B4 is suitable for classifying the passenger terminal, the radio tower, and the car park, while EfficientNet–B3 is suitable for the runway classification. Furthermore, the Accuracy of EfficientNet–B4 overcomes the other lower versions EfficientNet with the value of 90%. Lastly, we suggested EfficientNet–B4 as the most appropriate architecture for classifying these four types of main components in airports. However, based on this airport components datasets, the original EfficientNet B0–B4 architecture may be inaccurate classification especially when component had similarly shapes, background colors or objects to interested component. In addition, it is hard to detect and classify the peculiar characteristics of each components such as the jet bridge of the passenger terminal.

## 5.2 Future work

In fact, EfficientNet still has some limitation. First, EfficientNet may have inaccurately classification when we have the same type with different shape. Secondly, it cannot detect and classify the peculiar characteristic of each components. Due to the weakness of method, the EfficientNet architecture could be improved. Moreover, the surrounding area of component containing in our datasets could bias the prediction. The process of data preparation could be adjusted.

# APPENDICES

**APPENDIX A :** Code for classification with EfficientNet-B0

```
#Import Library
import os
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tqdm import tqdm
from sklearn.metrics import confusion_matrix,accuracy_score,
    precision_score,recall_score,f1_score,classification_report
#Import Model
from keras.applications.imagenet_utils import decode_predictions
from efficientnet.tfkeras import EfficientNetB0
from efficientnet.tfkeras import center_crop_and_resize,
    preprocess_input
model_B0 = EfficientNetB0(weights=None,classes=4,augmentation=True,
    pre_process='normalize')
#Split Datasetss
image_size = model_B0.input_shape[1:3]
batch_size = 32
#Train Set
train_ds = tf.keras.preprocessing.image_dataset_from_directory("/
    content/drive/MyDrive/ABdataset_split/train", labels='inferred',
    image_size=image_size, batch_size=batch_size)
#Validation Set
val_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/
    drive/MyDrive/ABdataset_split/val", labels='inferred', image_size=
    image_size, batch_size=batch_size)
#Test Set
test_ds = tf.keras.preprocessing.image_dataset_from_directory("/content
    /drive/MyDrive/ABdataset_split/test", labels='inferred', image_size
    =image_size, batch_size=batch_size)
train_ds = train_ds.prefetch(buffer_size=batch_size)
val_ds = val_ds.prefetch(buffer_size=batch_size)
test_ds = test_ds.prefetch(buffer_size=batch_size)
#Training Model
epochs = 50
model_B0.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='
    sparse_categorical_crossentropy', metrics=['
    sparse_categorical_accuracy'])
history = model_B0.fit(train_ds, epochs=epochs, validation_data=val_ds)
#Test Model
test_result = model_B0.evaluate(test_ds)
test_img = []
test_class = []
for X,y in test_ds:
  test_img = test_img + list(X)
  test_class = test_class + list(y)
test_img = np.array(test_img)
test_pred = np.argmax(model_B0.predict(test_img),axis=1)
print(classification_report(test_class,test_pred))
confusion_matrix(test_class,test_pred)
```

## **APPENDIX B :** Code for classification with EfficientNet-B1

```
#Import Library
import os
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tqdm import tqdm
from sklearn.metrics import confusion_matrix,accuracy_score,
    precision_score,recall_score,f1_score,classification_report
#Import Model
from keras.applications.imagenet_utils import decode_predictions
from efficientnet.tfkeras import EfficientNetB1
from efficientnet.tfkeras import center_crop_and_resize,
    preprocess_input
model_B1 = EfficientNetB1(weights=None,classes=4,augmentation=True,
    pre_process='normalize')
#Split Datasets
image_size = model_B1.input_shape[1:3]
batch_size = 32
#Train Set
train_ds = tf.keras.preprocessing.image_dataset_from_directory("/
    content/drive/MyDrive/ABdataset_split/train", labels='inferred',
    image_size=image_size, batch_size=batch_size)
#Validation Set
val_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/
    drive/MyDrive/ABdataset_split/val", labels='inferred', image_size=
    image_size, batch_size=batch_size)
#Test Set
test_ds = tf.keras.preprocessing.image_dataset_from_directory("/content
    /drive/MyDrive/ABdataset_split/test", labels='inferred', image_size
    =image_size, batch_size=batch_size)
train_ds = train_ds.prefetch(buffer_size=batch_size)
val_ds = val_ds.prefetch(buffer_size=batch_size)
test_ds = test_ds.prefetch(buffer_size=batch_size)
#Training Model
epochs = 50
model_B1.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='
    sparse_categorical_crossentropy', metrics=['
    sparse_categorical_accuracy'])
history = model_B1.fit(train_ds, epochs=epochs, validation_data=val_ds)
#Test Model
test_result = model_B1.evaluate(test_ds)
test_img = []
test_class = []
for X,y in test_ds:
  test_img = test_img + list(X)
  test_class = test_class + list(y)
test_img = np.array(test_img)
test_pred = np.argmax(model_B1.predict(test_img),axis=1)
print(classification_report(test_class,test_pred))
confusion_matrix(test_class,test_pred)
```

**APPENDIX C :** Code for classification with EfficientNet-B2

```
#Import Library
import os
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tqdm import tqdm
from sklearn.metrics import confusion_matrix,accuracy_score,
    precision_score,recall_score,f1_score,classification_report
#Import Model
from keras.applications.imagenet_utils import decode_predictions
from efficientnet.tfkeras import EfficientNetB2
from efficientnet.tfkeras import center_crop_and_resize,
    preprocess_input
model_B2 = EfficientNetB2(weights=None,classes=4,augmentation=True,
    pre_process='normalize')
#Split Datasets
image_size = model_B2.input_shape[1:3]
batch_size = 32
#Train Set
train_ds = tf.keras.preprocessing.image_dataset_from_directory("/
    content/drive/MyDrive/ABdataset_split/train", labels='inferred',
    image_size=image_size, batch_size=batch_size)
#Validation Set
val_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/
    drive/MyDrive/ABdataset_split/val", labels='inferred', image_size=
    image_size, batch_size=batch_size)
#Test Set
test_ds = tf.keras.preprocessing.image_dataset_from_directory("/content
    /drive/MyDrive/ABdataset_split/test", labels='inferred', image_size
    =image_size, batch_size=batch_size)
train_ds = train_ds.prefetch(buffer_size=batch_size)
val_ds = val_ds.prefetch(buffer_size=batch_size)
test_ds = test_ds.prefetch(buffer_size=batch_size)
#Training Model
epochs = 50
model_B2.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='
    sparse_categorical_crossentropy', metrics=['
    sparse_categorical_accuracy'])
history = model_B2.fit(train_ds, epochs=epochs, validation_data=val_ds)
#Test Model
test_result = model_B2.evaluate(test_ds)
test_img = []
test_class = []
for X,y in test_ds:
  test_img = test_img + list(X)
  test_class = test_class + list(y)
test_img = np.array(test_img)
test_pred = np.argmax(model_B2.predict(test_img),axis=1)
print(classification_report(test_class,test_pred))
confusion_matrix(test_class,test_pred)
```

**APPENDIX D :** Code for classification with EfficientNet-B3

```
#Import Library
import os
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tqdm import tqdm
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_score,f1_score,classification_report
#Import Model
from keras.applications.imagenet_utils import decode_predictions
from efficientnet.tfkeras import EfficientNetB3
from efficientnet.tfkeras import center_crop_and_resize,preprocess_input
model_B3 = EfficientNetB3(weights=None,classes=4,augmentation=True,pre_process='normalize')
#Split Datasets
image_size = model_B3.input_shape[1:3]
batch_size = 32
#Train Set
train_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/drive/MyDrive/ABdataset_split/train", labels='inferred', image_size=image_size, batch_size=batch_size)
#Validation Set
val_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/drive/MyDrive/ABdataset_split/val", labels='inferred', image_size=image_size, batch_size=batch_size)
#Test Set
test_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/drive/MyDrive/ABdataset_split/test", labels='inferred', image_size
    =image_size, batch_size=batch_size)
train_ds = train_ds.prefetch(buffer_size=batch_size)
val_ds = val_ds.prefetch(buffer_size=batch_size)
test_ds = test_ds.prefetch(buffer_size=batch_size)
#Training Model
epochs = 50
model_B3.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='
    sparse_categorical_crossentropy', metrics=['
    sparse_categorical_accuracy'])
history = model_B3.fit(train_ds, epochs=epochs, validation_data=val_ds)
#Test Model
test_result = model_B3.evaluate(test_ds)
test_img = []
test_class = []
for X,y in test_ds:
  test_img = test_img + list(X)
  test_class = test_class + list(y)
test_img = np.array(test_img)
test_pred = np.argmax(model_B3.predict(test_img),axis=1)
print(classification_report(test_class,test_pred))
confusion_matrix(test_class,test_pred)
```

## **APPENDIX E :** Code for classification with EfficientNet-B4

```
#Import Library
import os
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tqdm import tqdm
from sklearn.metrics import confusion_matrix,accuracy_score,
    precision_score,recall_score,f1_score,classification_report
#Import Model
from keras.applications.imagenet_utils import decode_predictions
from efficientnet.tfkeras import EfficientNetB4
from efficientnet.tfkeras import center_crop_and_resize,
    preprocess_input
model_B4 = EfficientNetB4(weights=None,classes=4,augmentation=True,
    pre_process='normalize')
#Split Datasets
image_size = model_B4.input_shape[1:3]
batch_size = 32
#Train Set
train_ds = tf.keras.preprocessing.image_dataset_from_directory("/
    content/drive/MyDrive/ABdataset_split/train", labels='inferred',
    image_size=image_size, batch_size=batch_size)
#Validation Set
val_ds = tf.keras.preprocessing.image_dataset_from_directory("/content/
    drive/MyDrive/ABdataset_split/val", labels='inferred', image_size=
    image_size, batch_size=batch_size)
#Test Set
test_ds = tf.keras.preprocessing.image_dataset_from_directory("/content
    /drive/MyDrive/ABdataset_split/test", labels='inferred', image_size
    =image_size, batch_size=batch_size)
train_ds = train_ds.prefetch(buffer_size=batch_size)
val_ds = val_ds.prefetch(buffer_size=batch_size)
test_ds = test_ds.prefetch(buffer_size=batch_size)
#Training Model
epochs = 50
model_B4.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='
    sparse_categorical_crossentropy', metrics=['
    sparse_categorical_accuracy'])
history = model_B4.fit(train_ds, epochs=epochs, validation_data=val_ds)
#Test Model
test_result = model_B4.evaluate(test_ds)
test_img = []
test_class = []
for X,y in test_ds:
  test_img = test_img + list(X)
  test_class = test_class + list(y)
test_img = np.array(test_img)
test_pred = np.argmax(model_B4.predict(test_img),axis=1)
print(classification_report(test_class,test_pred))
confusion_matrix(test_class,test_pred)
```

# BIOGRAPHY

| | |
|---|---|
| **Name** | Miss Pimpisa Charoenchittang |
| **Date of Birth** | August 29, 1995 |
| **Place of Birth** | Bangkok, Thailand |
| **Educations** | B.Sc. (Mathematics) (Second Class Honours), Kasetsart University, 2018 |
| **Scholarships** | Science Achievement Scholarship of Thailand (SAST) |

**Publications**

- P. Charoenchittang, P. Boonserm, K. Kobayashi, and N. Cooharojananone, “Airport-buildings classification through remote sensing images using efficientnet,*”2021 18thInternational Conference on Electrical Engineering/Electronics, Computer, Telecom-munications and Information Technology (ECTI-CON)*, pp. 127–130, 2021.