A DATA ENGINEERING SOLUTION FOR MICROMETEOROLOGY FORECASTING USING CLOUD RECOGNITION

BY

CHAI AI JIA

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2024

UNIVERSITI TUNKU ABDUL RAHMAN

D

REPORT STATUS DECLARATION FORM

Title: A DATA ENGINEERING SOLUTION FOR MICROMETEOROLOGY FORECASTING USING CLOUD RECOGNITION

Academic Session: JAN 2024

I CHAI AI JIA

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible]

(Author's signature)

Alex

(Supervisor's signature)

Address:
20, JALAN SUNGAI KERAMAT 17,
TAMAN KLANG UTAMA,
42100, KLANG, SELANGOR.

Dr. Ooi Boon Yaik
Supervisor's name

Date: 26/4/2024

Date: 26/4/2024

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 26/4/2024

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that ***CHAI AI JIA*** (ID No: ***20ACB03286***) has completed this final year project/ dissertation/ thesis* entitled "A Data Engineering Solution for Micrometeorology Forecasting using Cloud Recognition" under the supervision of Dr. Ooi Boon Yaik (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

(CHAI AI JIA)

DECLARATION OF ORIGINALITY

I declare that this report entitled "**A DATA ENGINEERING SOLUTION FOR MICROMETEOROLOGY FORECASTING USING CLOUD RECOGNITION**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	CHAI AI JIA
Date	:	26/4/2024

ACKNOWLEDGEMENTS

I would like to express sincere thanks and appreciation to my supervisor, Dr. Ooi Boon Yaik and my moderator, Dr. Kh’ng Xin Yi, for their invaluable guidance, unwavering support and insightful feedback throughout the entire duration of this project. Their expertise and encouragement have been instrumental in shaping the development of this research.

I would also like to express my heartfelt thanks to my parents and family for their continuous encouragement, understanding and patience throughout this academic journey. Their unwavering support has been my pillar of strength, providing me with the motivation and resilience needed to overcome challenges.

To my supervisor, moderator and my family, I am profoundly thankful for the roles you have played in my academic and personal growth. Your support has made a significant impact on this endeavour, and I am grateful for the privilege of having you by my side.

ABSTRACT

Micrometeorology forecasting stands at the forefront of accurate localized weather predictions, providing invaluable insights for individuals, communities and industries alike. Traditional weather forecast systems predominantly operate on mesoscale and synoptic meteorology, providing generalized information for large geographic regions. However, the micrometeorological conditions within smaller areas are often overlooked, leading to inaccuracies in local weather predictions. Apart from that, most of the existing methods rely on costly data collection and complex modelling. Thus, this has limited their accessibility and usability. Therefore, this project aims to address these pivotal challenges in weather forecasting.

This project falls within the domain of micrometeorology and weather forecasting, focusing on developing a data engineering solution for the micrometeorology forecasting system and the recognition of cloud patterns to predict localized rainfall events. The proposed system employs a multi-faceted approach. It begins with the automatic collection of cloud images by capturing it with a camera module. After capturing, the images are automatically uploaded to AWS cloud storage platform. By leveraging machine learning techniques, the images will undergo a labelling process to categorize them as either rainy or not rainy. Manual labelling method will then be used to validate the accuracy of the previous annotation to create a ground truth dataset. Next, classification method will be utilized to classify the dataset into different classes. This approach empowers the system to label the new images automatically in future. Lastly, a predictive model will be trained to recognize the cloud formation patterns and predict the probability of the occurrence of the rainfall events. The novelties of this project encompass several groundbreaking aspects including the automated cloud recognition, localized and real-time predictions and the integration of data engineering methodologies to ensure the adaptability of the system to various geographical areas by facilitating the efficient collection, storage and labelling of cloud images. This streamlined process enables the generation of labeled datasets essential for training accurate predictive models, thus enhancing the system's effectiveness across diverse environments. Moreover, this project also aims to offer a cost-effective solution by using only a camera module to predict the rainfall events, making weather predictions more accessible to a broader audience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NWP	Numerical Weather Prediction
SeLa	Self-Labelling
LBP	Local Binary Patterns
LTP	Local Ternary Patterns
ICS	Intra-Class Similarity
CNN	Convolutional Neural Network
WEAPD	Weather Phenomenon Database
LSTM	Long Short-Term Memory
FC-LSTM	Fully Connected Long Short-Term Memory
ConvLSTM	Convolutional Long Short-Term Memory
HRRR	High-Resolution Rapid Refresh
CLI	Command Line Interface
AWS	Amazon Web Services

Chapter 1
Introduction

Micrometeorology is a specialized field of meteorology that studies small-scale atmospheric processes and interactions occurring close to the Earth's surface, focusing on phenomena ranging from a few centimetres to a few kilometres with short lifespans. Weather forecasting relies on various sources such as weather stations, balloons and satellites, making it challenging for individuals interested in personal weather forecasting. To address this issue, this paper introduces the concept of utilizing cloud images for weather forecasting as a solution.

Modern weather forecasting primarily utilizes Numerical Weather Prediction (NWP), which involves solving dynamic and physics equations to predict future atmospheric conditions based on current data. NWP model is designed to predict atmosphere conditions over a range of scales and emphasize on global and synoptic scales. Conversely, this paper centers attention on microscale meteorology, which pertains to distances ranging from a few centimetres to kilometres and has a temporal scope of no more than 24 hours.

Micrometeorology forecasting offers high precision with detailed spatial resolution, benefiting sectors like construction and agriculture. It focuses on a shorter timeframe which is crucial for decision-making in outdoor events, construction and emergency responses. This paper also integrates data engineering methodologies to automate the collection, storage and labelling of cloud images. Therefore, it can enhance the effectiveness of localized weather predictions.

The paper's focus is on predicting rainfall events in Malaysia, given the country's two primary wet and dry seasons. Heavy rainfall during the monsoon season can lead to significant and far-reaching consequences for the country such as floods, landslides, public health concerns and infrastructure damage. As a consequence, the economy can experience adverse effects due to disrupted business operations, reduced productivity and substantial financial losses. Therefore, micrometeorology forecasting of rainfall events in Malaysia carries vital significance for several factors, including effective flood and disaster management, urban development planning and economic loss mitigation. The focus of this paper will be on developing a data engineering solution for micrometeorology forecasting system and to predict rainfall event, thus deliver advantages to individuals.

.

1.1 Problem Statement

Micrometeorological data collection and analysis is currently both costly, time-consuming and relying on complex tools such as satellites, radiosondes and radar systems. These instruments incur significant expenses, and manual data interpretation by meteorologists further adds to the complexity and time burden of weather forecasting.

Moreover, the absence of an automated weather type classification process poses a significant challenge in the field of meteorology. The process of identifying weather types is important as it provides meaningful insights on current and future atmospheric conditions. However, large effort is often needed to classify the data manually by experts which is time-consuming and prone to human error and subjectivity.

Additionally, current weather forecast systems primarily focus on predicting mesoscale meteorology which range from a few kilometres to approximately 1000 kilometres. However, the spatial resolution used are normally range from 5km to 20km. Spatial resolution refers to the degree of detail at which forecasting model can represent Earth's surface and its atmospheric conditions. Therefore, it may not capture fine-scale weather feathers that occur within a single grid box. This limitation results in less accurate and detailed predictions, especially for smaller geographical areas.

1.2 Motivation

The first motivation of this proposed system aims to help users in their decision making and schedule planning. This is important for those who involved in businesses reliant on weather conditions such as tourism and retail. This technological solution empowers users by providing them with actionable insights, enabling them to make well-informed decisions and optimize available resources. For instance, the retailers especially those who conduct night market businesses might face financial losses since their business might not be well on rainy day. In addition, their time will also be wasted for setting up their stall. Thus, by obtaining the rainfall predictions information, the vendors can reschedule their activities or determine whether it is an appropriate choice to open their stalls. This can lead to the conservation of manpower, time and also money. Additionally, the project targets the development of an affordable micrometeorology forecasting system to address financial challenges in small sectors like night market businesses, making it accessible through a camera module.

Additionally, the proposed system also aims to develop a data engineering solution that facilitates more convenient and effective cloud data collection. This solution is designed to streamline the process of cloud data collection, offering researchers a convenient and efficient means to gather cloud images. By leveraging this system, researchers interested in collecting cloud data can easily access and acquire the necessary images for their studies.

Furthermore, the system can enhance individual safety by offering localized rainfall event insights. This allows for the coordination of emergency responses and resource allocation, minimizing the impact of extreme rainfall events. Industries, particularly those operating during the day like construction, can adjust work schedules based on the forecasted result. This may reduce the risk of accidents associated with hazardous weather conditions.

1.3 Project Scope and Direction

The project's deliverable is a forecast system specialized in predicting localized rainfall events during daytime using cloud recognition techniques. While the forecasting is a key aspect, the primary emphasis lies on automating data engineering processes, particularly in collecting and labelling cloud images for predictive analysis. Due to time constraints, the project prioritizes the development of efficient mechanisms for automated data collection and the implementation of cloud recognition algorithms. This encompasses establishing automated data collection mechanisms, implementing cloud recognition algorithms, and constructing a predictive model to predict rainfall events. The forecast system offers a unique aspect of user adaptability. It empowers users to refine the predictive model specific to their geographical area as different regions may have distinct weather conditions and patterns. Upon deployment, this system enables users to predict the rainfall events catering to their unique meteorological circumstances.

Practical implementation of the system involves placing a camera module in the desired geographical area of interest. Through consistent data collection and analysis, the system gains an in-depth understanding of local weather patterns. Subsequently, it leverages the knowledge to predict future localized rainfall events which enable users to proactively plan and respond to potential weather-related challenges. Thus, users are given the ability to customize and contribute to the predictive model. The system aims to empower individuals with actionable insights for making informed decisions in the face of changing weather conditions.

1.4 Research Objectives

1. **To develop a system that automates the process of capturing cloud pictures and creating a ground truth dataset.**

   To establish a micrometeorology forecasting system, the initial step entails gathering data to train a classification and prediction model. This involves strategically placing a camera module on the geographical area of interest for weather predictions to capture cloud images. The positioning is crucial to comprehensively record diverse cloud patterns specific to that location. For automated data collection, a trigger mechanism will prompt the camera to capture images at predefined 5-minute intervals from 7AM to 7:30PM. Upon capture, the images will be automatically uploaded to AWS cloud storage for further processing and analysis. This has eased the retrieval of data for subsequent model training and evaluation. To create a ground truth dataset, a combination of rule-based and manual labelling approaches will be implemented on the dataset. This hybrid approach ensures both the effectiveness and accuracy of the dataset.

2. **To develop a model to recognize and classify rainfall images.**

   This objective aims to develop a classification model capable of recognizing and classifying rainfall images using machine learning techniques. To achieve this, multiple classification models including Convolutional Neural Networks (CNNs) and Recurrent Neural Network (RNN) will be trained and compared to determine the most effective approach. By training the model on ground truth dataset and fine-tuning its performance, it enables the system to fully automate the labelling process for cloud images in the future. This automation streamlines data labelling, enhances efficiency and scales the system for faster and more accurate predictions of rainfall events. Therefore, the use of classification techniques serves as a crucial step towards achieving full automation of the labelling process.

3. **To test whether the system can be used to predict or forecast rainfall events accurately.**

   This step aims to assess the system’s capability to generate accurate predictions of localized rainfall events within a designated time span based on the cloud images. To achieve this, a few predictive models will be trained by employing classification techniques using the labelled dataset. This is to identify cloud patterns indicative of impending rainfall. Subsequently, the models will generate probabilistic forecasts regarding the likelihood of rainfall events occurring within the specified geographic area. An evaluation will be conducted to compare the prediction accuracy of different models in forecasting rainfall events within this specific region. This evaluation serves to validate the system's practical utility for micrometeorology forecasting and demonstrate its effectiveness in leveraging cloud image for rainfall predictions.

1.5 Contribution

The proposed forecast system focuses on microscale meteorology, delivering users predictions of localized rainfall events tailored to their exact locations. Thus, this precision enhanced the system's value by providing users with insights specifically relevant to their immediate surroundings. Moreover, the system enables informed decision-making for industries like construction, outdoor retailer and tourism. Users can adjust their work schedules based on localized rainfall predictions and reduce economic losses by anticipating and mitigating disruptions caused by unexpected rainfall events.

Moreover, the system makes a significant contribution to the field of meteorological research by addressing the inherent challenges associated with cloud data collection. By introducing a streamlined data engineering solution, researchers are empowered with a convenient and efficient method to collect cloud data images.

Additionally, the system plays a crucial role in risk reduction and safety enhancement, particularly during the monsoon season. By utilizing forecast results, individuals can take proactive measures to safeguard lives and properties against extreme events such as floods and landslides associated with rainfall.

1.6 Background Information

Prediction algorithms, the outcome of predictive modelling is currently at the forefront of technological advancements. They are designed to make predictions or forecasts based on input data. This is achieved by using machine learning techniques to evaluate past data, identify patterns, analyse trends within datasets and ultimately generate prediction on future or unseen trend [1]. Predictive modelling finds applications across various domains such as fraud detection, customer relationship management and churn prediction. In the face of the ever-evolving climate, weather forecasting grapples with substantial challenges that reverberate across people's lives. Hence, there is a pressing need to adapt strategies to address the increasingly complex and unpredictable weather patterns. Traditional micrometeorology forecasts were mainly reliant on complex meteorological instruments and manual observations, making it prone to human error.

Before 1950, there were no notable advancements in micrometeorology forecasting.
The inception of dynamic forecasting represented a conceptual change by employing fundamental principles of hydrodynamics and thermodynamics to characterize the present condition of the atmosphere and predict its future state. By the 1950s, numerical weather prediction (NWP) techniques were introduced and implemented for short-range weather prediction. It was first utilized for research studies on air-pollutant transport and became instrumental in predicting the weather through the continued improvement of numerical forecasts [2]. The NWP model has emerged as the current state-of-the-art technology in this field. One example of the widely used NWP model is the High-Resolution Rapid Refresh (HRRR) model. This type of models can be used to predict weather phenomena by solving numerical equations governing changes in atmospheric conditions, yielding forecasts for parameters such as temperature, wind speed, dew point and more.

In this project, cloud images are used as the input data to be processed to predict the occurrence of rainfall event. The current state-of-the-art method for prediction involving time-series images often leverage a combination of convolutional neural networks (CNNs) to process the images. An example for this prediction model Convolutional LSTM (ConvLSTM) which integrates CNNs with and Long Stort-Term Memory Networks (LSTMs). Time series forecasting entails using a model to analyse past values at evenly spaced time points to predict future values along the same time axis [3].

On the other hand, image automatic annotation approach may be required to label a large amount of data to create a ground truth dataset. This is necessary as it is essential to differentiate between images that portray rainfall and those depicting a clear sky. Automate image labelling plays a pivotal role as it increases the efficiency of data collection process since manual labelling of images is both time-consuming and labour-intensive. The state-of-the-art methods for automated image annotation often involve the use of deep learning techniques, particularly CNNs. VGG-16 and ResNet50 are the examples of CNN models that originally designated for image classification, but they can be adapted for image annotation by modifying the last layer to output multiple labels and train the network on a dataset [4]. However, automated image annotation can also be done by using discriminative models like rule-based labelling. This can be done by defining a set of rules based on the extracted features to assign labels to the images [5]. For example, if an image has a dominant green colour, it could be labelled as “tree”.

Furthermore, image classification is crucial for categorizing cloud images into distinct classes and facilitate the automation of the image annotation process. It is a task of assigning label to an input image from a predefined set of categories. This process entails analysing the input image and return a label of the category to which the image belongs [6]. Image classification serves as a foundational task in computer vision and has numerous applications such as satellite image analysis, facial recognition and medical image analysis. Consequently, it proves valuable for labelling extensive datasets by training a classification model using a ground truth dataset. The prevailing state-of-the-art methods for image classification are CNN-based models, including AlexNet, VGGNet, GoogleNet and ResNet [7]. These CNN models work by extracting features from the images through convolutional, max-pooling and fully connected layers. In the training process, sample data containing input features and corresponding outputs are fed into the model. Through this iterative process, the model learns intricate features and patterns within the images, ultimately enabling it to effectively classify new images [8].

1.7 Report Organization

The specifics of this research are outlined in the subsequent chapters. Chapter 2 comprises a literature review and discussion of the existing research. Following this, Chapter 3 provides a detailed explanation and visualization of the proposed methods for the micrometeorology forecasting system. Moving forward, Chapter 4 details how the system is set up, including both software and hardware aspects, and provides a thorough overview of how the system operates. Chapter 5 delves into a detailed evaluation and discussion on the findings and outcomes outlined in Chapter 4. Lastly, Chapter 6 concludes the research, summarizing the work done and suggesting future research directions.

Chapter 2
Literature Review

2.1 Unsupervised Learning for Self-Labelling

Unsupervised learning for self-labelling is a process where a machine learning model identifies and categorizes data without prior labelled examples or human intervention [9]. Instead, the algorithm identifies patterns within the data and label it based on the similarities in the data points. This approach is useful when dealing with large number of datasets where manual labelling would be impractical or costly.

2.1.1 WCATN: Unsupervised Deep Learning to Classify Weather Conditions from Outdoor Images

Weather classification is a significant step to perform weather forecast. Most of the existing methods is view as supervised task where the model is trained on a labelled dataset. However, these methods are not applicable in real-world scenarios as huge human-annotated weather images are needed to train the model. To address this limitation, Xie et al. 2022 [10] introduced an innovative approach by casting weather classification as an unsupervised task. The proposed methodology consists of three step to group images into weather clusters using unsupervised learning techniques. This involves weather feature learning, employing a self-supervised task with online triplet mining for enhanced feature extraction. Subsequently, learnable clustering is introduced through the utilization of nearest neighbours and consistency enforcement. Furthermore, a self-labelling method is introduced to mitigate uncertainty caused by noisy neighbours and for further refinement. This review sets the stage for an exploration of unsupervised weather classification techniques, contributing to the advancement of weather classification in computer vision. Figure 2.1 shows the overall framework of the unsupervised approach for weather clustering.

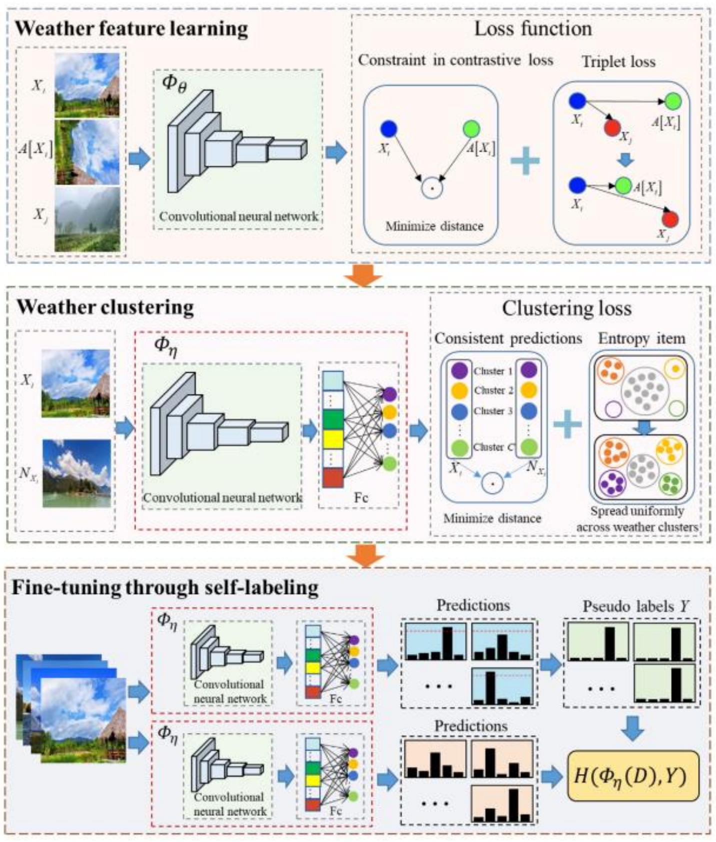


Figure 2.1 Overall framework of the unsupervised approach for weather clustering.

The strength of this paper lies in its innovative approach to weather classification from outdoor images without the use of annotations. By venturing into the domain of unsupervised learning for this task, this paper addresses a significant gap in the past research and opens up new possibilities for real-world applications on classification. Apart from that, the images are processed assuming that the outdoor images depict clear weather conditions, this led to the ignorance of complex weather condition such as haze and rain. Thus, the innovative approach helps in the save of costs on weather image annotation as there is large demand for annotated data.

The proposed three-step unsupervised approach is a well-structured strategy for automatically clustering images based on different weather conditions. This approach solved the struggle to effectively capture the distinctive high-level semantic features related to weather conditions. In the first step, the integration of self-supervised approach and online triplet mining has been introduced to acquire distinctive weather representations using a pretext task. In supervised classification approach, a ground-truth labels of weather classes needed to be defined in the dataset as the prior step. It enables the mapping of images to weather classes. Therefore, this self-supervised approach can solve the difficulties to capture high-level weather features early in the model training process as the existing method are sensitive to network initialization for clustering. Furthermore, the design of a learnable weather clustering by utilizing the nearest neighbours of each sample as the prior has prevent the domination of weather clusters to the others. In the third step, a self-labelling method has been employed to exploit accurately classified weather images, and effectively fine-tune errors introduced by the noisy nearest neighbours.

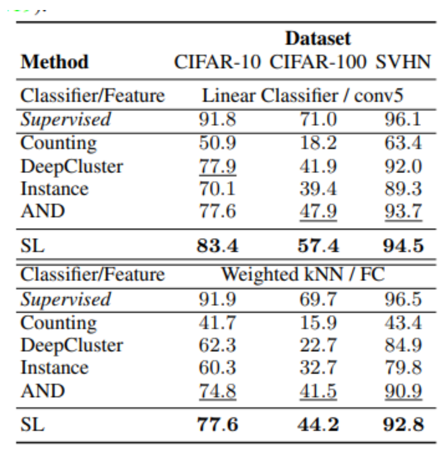
The potential weakness of this paper is the uncertainty in fine-tuning. The introduction of fine-tuning via self-labelling is intended to improve the clustering network's certainty. However, the paper does not discuss the potential trade-offs or uncertainties that might arise during the fine-tuning process. One of the uncertainties that could be associated with this process is labelling accuracy. As self-labelling relies on the assumption that the initially well-classified samples can provide accuracy labels for the rest of the data. However, this assumption might not hold true for all instances, leading to inaccuracies in the fine-tuning process. This uncertainty lies in how reliable the self-labelled samples are in correcting the clustering network's predictions.

The solution recommended to address the weakness in this paper is to employ the iterative approach to the fine-tuning process. This can be done by fine-tune the model multiple times using different subsets of self-labelled samples. By doing so, the uncertainty can be reduced by averaging out the effects of individual samples This iterative refinement approach can helps to contribute to a more stable and reliable fine-tuned model in the context of weather classification.

2.1.2 Self-Labelling via Simultaneous Clustering and Representation Learning

Asano et al. 2019 [11] proposed an innovative approach that integrate both clustering and representation learning for unsupervised learning of deep neural networks. The purpose of this approach is to overcome the challenges associated with degenerate solutions that often arise when integrating these two techniques. Thus, the authors proposed a principled learning formulation that centres around maximizing details between labels and indices of input data. This paper also leverages a variant of the Sinkhorn-Knopp algorithm to handle massive input datasets. The resulting method offers a solution for self-labelling visual data and train competitive image representations without the need for manual labels.

This paper has introduced the formulation of a unified objective that simultaneously optimizes both feature learning and clustering. This integration ensures the learned features align well with the clustering objectives and result in more informative representations. Other than that, the utilization of a modified Sinkhorn-Knopp algorithm enforce the weak assumption which states that the number of dataset samples should be equal across clusters. Therefore, the method is allowed to scale effectively to large datasets and making it more applicable to real-world scenarios. In this paper, an experiment has been carried out to evaluate the performance of the proposed method – SeLa and compare with the state of the art in self-supervised representation learning. The results show that the proposed method achieved the best among those other techniques. Figure 2.2 shows the results of linear probing evaluation.



	Dataset		
Method	CIFAR-10	CIFAR-100	SVHN
Classifier/Feature	Linear Classifier / conv5		
Supervised	91.8	71.0	96.1
Counting	50.9	18.2	63.4
DeepCluster	77.9	41.9	92.0
Instance	70.1	39.4	89.3
AND	77.6	47.9	93.7
SL	**83.4**	**57.4**	**94.5**
Classifier/Feature	Weighted kNN / FC		
Supervised	91.9	69.7	96.5
Counting	41.7	15.9	43.4
DeepCluster	62.3	22.7	84.9
Instance	60.3	32.7	79.8
AND	74.8	41.5	90.9
SL	**77.6**	**44.2**	**92.8**

Figure 2.2 Linear probing evaluation results.

The proposed method is heavily relied on the assumption of an equal number of samples across clusters. This is a weakness as the assumption might not hold in all real-world scenarios and limit the applicability of the proposed approach. The performance could be suboptimal if there is a situation with imbalanced data distributions.

Imbalanced data handling can be applied to solve the weakness of this paper. This might involve techniques such as oversampling, under sampling or generating synthetic samples to balance the clusters and reduce impact of imbalanced data.

2.1.3 Comparison of the Self-Labelling Methods

Table 2.1 Comparison table for self-labelling methods.

Literature Review	Method Used	Strength	Weakness
2.1.1	● weather feature learning (CNN) ● weather clustering (K-Means algorithm) ● fine-tuning through self-labelling (use weather clustering network to obtain pseudo labels and utilize the labels to fine-tune the network)	● The images can be classified without the use of annotation. ● The approach solved the struggle to effectively capture the distinctive high-level semantic features related to weather conditions.	● Uncertainties exist during the fine-tuning process which may affect the applicability of the method in real-world.
2.1.2	● Self-labelling technique (SeLa) – a modified Sinkhorn-Knopp algorithm	● The proposed approach can simultaneously optimize both feature learning and clustering. ● The proposed method scale effectively to large datasets.	● The proposed method heavily relied on the assumption of an equal number of samples across clusters.

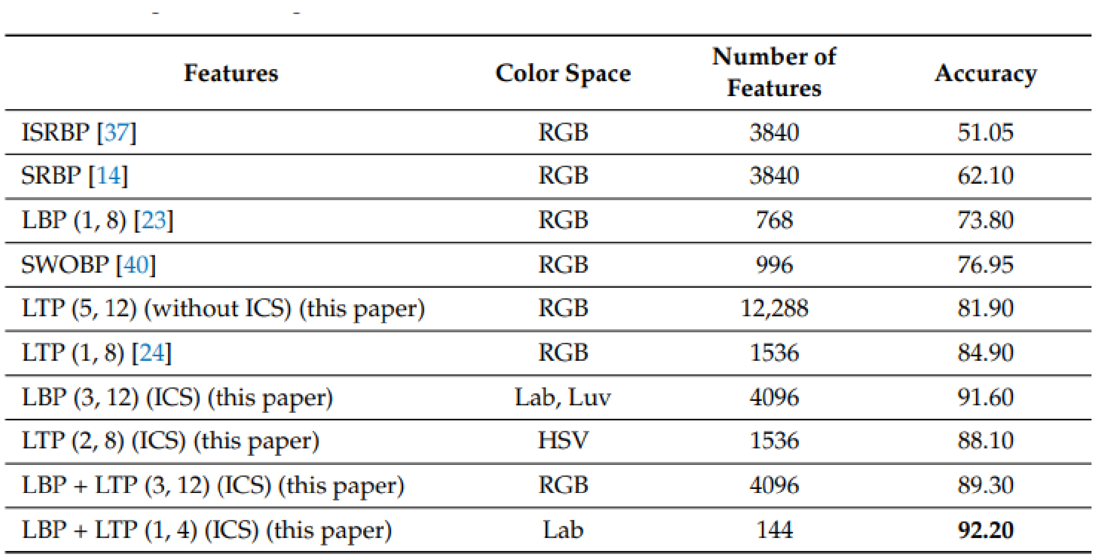
2.2 Classification Method

A classification model is a specific type of machine leaning model designed to classify data into predetermined classes or categories. Its primary purpose is to discern patterns and relationships inherent in the data, enabling it to make predictions about which class a new or unseen data point belongs to [12].

2.2.1 Weather Forecast Based on Color Cloud Image Recognition under the Combination of Local Image Descriptor and Histogram Selection

The cloud classification is an important step in weather forecast as it will significantly impact the future measurements and predictions. By locating and analysing the clouds, meteorologists can refine weather forecasts and gain deeper insights into the climate conditions. Tran-Trung et al. 2022 [13] has introduced several innovative computer vision techniques, including the fusion of colour characteristics, Local Binary Patterns (LBP) features and Local Ternary Patterns (LTP) features which aimed to perform cloud classification. The LBP descriptor and its variants have been used as feature extraction methods to characterize natural textual images. In a pivotal stride to transcend the confines of LBP, this research paper introduced the concept of the LTP feature to address the limitations of the LBP feature. Furthermore, the Intra-Class Similarity (ICS) technique has been applied as a strategic enhancement to the proposed methodology. Through the implementation of histogram selection, ICS effectively reduces the number of histograms required for image characterization. Thus, this paper showcases improved classification performance, contributed for advanced cloud analysis within the domains of meteorology and weather forecasting.

The strength of this paper lies in its systematic analysis and exploration of feature extraction techniques for cloud classification. This paper demonstrates the adaptability of both LBP and LTP feature extraction methods to the classification task. The integration of colour space and texture structure through the use of LBP and LTP features has result in the improvements of classification accuracy. This fusion of colour and texture features enhances the overall performance of could classification. In research done by Jose. J and other authors, the LBP feature and its variants has been declared as an effective feature extraction method as it gives many positive results in the extraction of local features [14]. However, the binary data produced by LBP are extremely noise-sensitive and it depends on intensity differences which lead to the struggles on flat regions of an image. Thus, the integration of LTP feature can effectively overcome the drawbacks of the feature as it can handle noise in a better way in the homogenous regions. In addition, the use of ICS technique helps in selecting the potential histograms and enhance the performance with fewer features. Figure 2.3 shows the results of classification on a dataset using different feature methods. The integration of LBP and LTP features using ICS technique has achieved the highest accuracy of classification.



Features	Color Space	Number of Features	Accuracy
ISRBP [37]	RGB	3840	51.05
SRBP [14]	RGB	3840	62.10
LBP (1, 8) [23]	RGB	768	73.80
SWOBP [40]	RGB	996	76.95
LTP (5, 12) (without ICS) (this paper)	RGB	12,288	81.90
LTP (1, 8) [24]	RGB	1536	84.90
LBP (3, 12) (ICS) (this paper)	Lab, Luv	4096	91.60
LTP (2, 8) (ICS) (this paper)	HSV	1536	88.10
LBP + LTP (3, 12) (ICS) (this paper)	RGB	4096	89.30
LBP + LTP (1, 4) (ICS) (this paper)	Lab	144	**92.20**

Figure 2.3 Results of classification of different methods.

The weakness contain in this paper is the limited parameter optimization for LTP feature. The paper stated that one of the parameters used in the formula for calculating LTP feature extraction for RGB colour space, the parameter value ‘t’ was obtained through exhaustive techniques. This implies that the paper might not have explored more sophisticated optimization methods for determining the most suitable parameter value. Consequently, this limitation could affect the applicability and performance of the LTP feature across various RGB colour spaces.

The recommendation for this paper is to optimize the LTP parameter ‘t’. Instead of relying solely on exhaustive techniques, the authors can consider exploring optimization algorithms to determine the optimal parameter value ‘t’ for the LTP feature. By embracing the optimization techniques, the authors can improve the performance and consistency across different scenarios, particularly when operating within the RGB colour space. This optimization process can refine the LTP feature’s effectiveness and contribute to the proposed method’s overall robustness and reliability.

2.2.2 Classification of Weather Phenomenon from Images by Using Deep Convolutional Neural Network

In this paper, Xiao et al. 2021 [15] have proposed a novel approach using a deep convolutional neural network (CNN) called MeteCNN for weather phenomenon classification. This classification approach can help in the analysis of weather phenomenon which play an essential role in various application such as weather forecast. Additionally, a comprehensive dataset named the Weather Phenomenon Database (WEAPD) which contain 6877 images with 22 weather phenomena has been established. The MeteCNN model has been used to classify the weather phenomenon on WEAPD dataset to test the classification accuracy of that particular model. The MeteCNN model has achieved a satisfy classification accuracy and highlight the effectiveness and superiority of the proposed approach. It helps in the development of high-precision and efficient weather phenomena classification.

One of the strengths of this paper is the establishment of advanced classification model. The proposal of the MeteCNN model can effectively learns intricate features of weather phenomena, outperforming the distinctive features associated with each weather phenomenon category accurately and yield a high-quality classification result. Figure 2.4 shows the architecture of MeteCNN model.

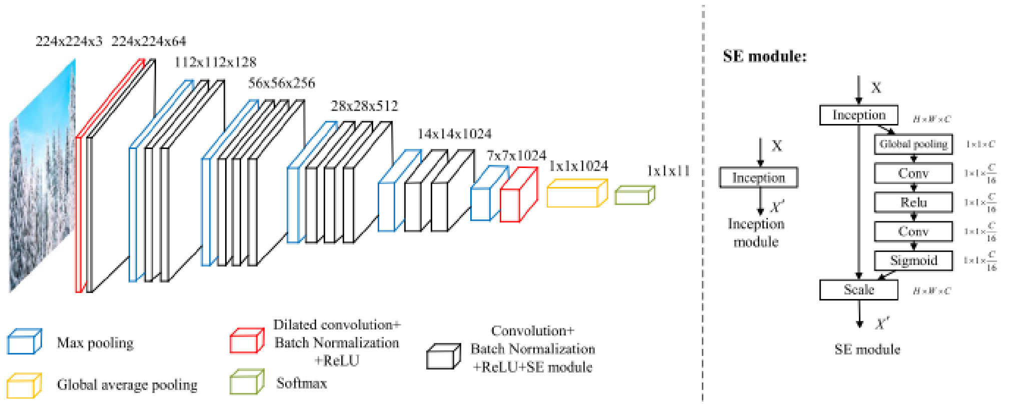


Figure 2.4 MeteCNN model architecture.

The input of MeteCNN is a batch of weather phenomenon images with consistent dimensions. The Convolutional layers within the model serve as proficient feature extractors, transforming the input images to abstract the representations of weather phenomena features. Thus, this model can learn the features associated with weather phenomena effectively. Moreover, an extensive experiment has been carried out to compare the accuracy of MeteCNN model with other models. The experiment results shows that the performance of the proposed model has a competitive classification performance compare with other mainstream models. Thus, the proposed model can be applied to the observation and analysis of weather phenomenon images. It can additionally offer valuable weather insights for diverse field such as environmental monitoring and agriculture especially on weather changes and forecasting. Apart from that, the establishment of the new database which consist of weather phenomena images covering 11 weather phenomenon types has showcases a comprehensive dataset. This surpasses previous established databases, providing researchers with a solid foundation for future weather-related studies.

The weakness of this paper pertains to the potential limitations in the direct applicability of the experimental results to real-world situations. This is because the datasets contained in the WEAPD database are acquired from Internet and academic exchanges. The dataset's limited size and controlled environment might not fully capture the wide range of weather conditions commonly encountered in real-world environments. Consequently, the findings derived from these datasets might not accurately represent the diversity and complexity of weather conditions and scenarios encountered in real-world. Furthermore, the model's robustness to real-world scenarios such as varying resolutions, image qualities and unexpected objects in the frame has not been thoroughly evaluated in the controlled experiments. Thus, the absence of such evaluations raises concerns about the model's potential degradation in performance when exposed to these challenges.

To address the weakness identified in the paper, the dataset acquisition process can be diversified to bridge the gap between controlled datasets and real-world complexities. Augmentation of the dataset with broader range of weather image sourced from various geographical locations and sources can be implemented to ensure the model encounters a wider range of weather conditions and scenarios. Additionally, the proposed model can be evaluated using real-world image. This will provide a more accurate representation of how the model will perform in actual use.

2.2.3 CloudA: A Ground-Based Cloud Classification Method with a Convolutional Neural Network

Wang et al. 2020 [16] proposed a novel approach called CloudA for recognizing and classifying ground-based cloud images. CloudA is a novel convolutional neural network (CNN) equipped with deep learning capabilities. The cloud features detected by CloudA can be visualized through the TensorBoard visualization method, aiding in comprehending the ground-based cloud classification process. An experiment has been carried out in this paper to assess the effectiveness of CloudA and compare its performance against other commonly used methods. The "from-scratch" training approach employed by CloudA is a notable strength of this paper. Unlike many traditional methods that rely on predefined feature extraction techniques or transfer learning from pre-trained modes, it learns relevant features directly from the raw cloud images, making it an end-to-end solution. Other than that, the experiment results shows that CloudA outperforms popular traditional methods in terms of classification accuracy. Therefore, CloudA is a more stable classification system compared to traditional methods.

This paper focuses on classification accuracy within a controlled environment. However, it is important to assess how well CloudA performs in real-world scenarios where factors such as varying geographical regions, weather conditions and lightning can significantly impact the performance of the proposed method.

To address this weakness, the authors can acquire a more diverse dataset that covers a broader range of geographical regions, weather conditions and lightning scenarios. This may ensure the model's adaptability to real-world variations.

2.2.4 Transfer Learning Approach – An Efficient Method to Predict Rainfall Based on Ground-Based Cloud Images

Rainfall prediction is a complex task with profound implication for society. Making predictions that are both timely and precise has the potential to pre-emptively mitigate the adverse impacts of rainfall, thereby minimizing human and economic losses. Ambildhuke et al. 2021 [17] introduced a novel approach to forecast rainfall using ground-based cloud images that correspond to different precipitation levels using transfer learning techniques. The rainfall event will be predicted by identifying cloud types from input cloud images. The dataset comprises three distinct categories representing varying levels of precipitation. The model is initially trained using a Convolutional Neural Network (CNN) architecture and subsequently experimented with well-established pre-trained models such as VGG16, Inception-V3 and XCeption, employing transfer learning techniques. The results attained from employing transfer learning technique exhibit outstanding performance, demonstrating its ability to meet real-time operational needs effectively. Figure 2.5 shows the working model of transfer learning.

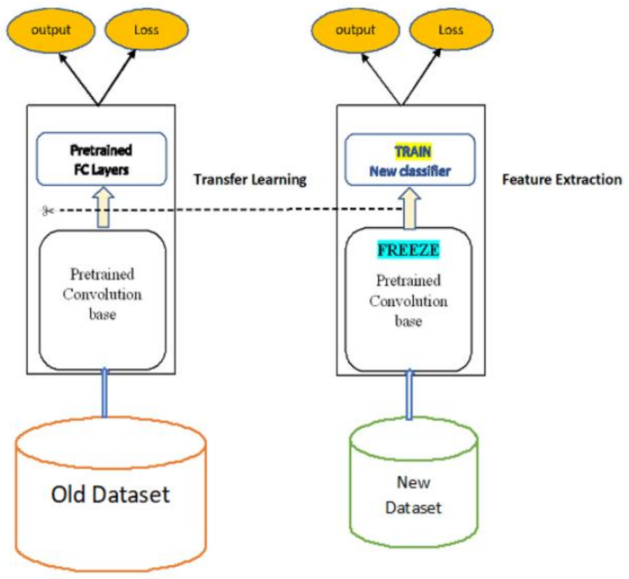


Figure 2.5 Working model of transfer learning.

The strength of this paper lies in its comprehensive exploration and application of transfer learning techniques to forecast rainfall event based on ground-based cloud images. It offers valuable insights into the comparative performance of different models, highlighting the efficiency and effectiveness of the transfer learning approach especially with the XCeption model. Figure 2.6 shows the architecture of XCeption model.

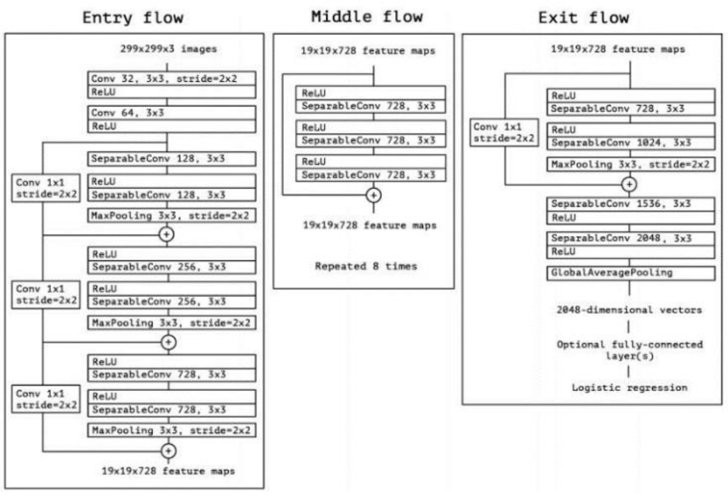


Figure 2.6 The architecture of XCeption model.

In this paper, an experiment has been carried out to predict the rainfall event based on the cloud types contained in the input images. According to the experiment result, the XCeption model has used the least amount of time to complete the experiment and achieved the highest accuracy compared to VGG16 and Inception-V3 models. The accuracy of weather predictions using XCeption model is considered high and can be used in the real world. Thus, this model proves useful for individuals to forecast the rainfall event at any location by providing a current sky image as input.

The weakness present in this paper is the relatively small dataset used for training and validation. This is due to the reason that the model's performance might be constrained by the dataset's size and diversity. The paper acknowledges the potential for improvement through dataset enrichment, implying an awareness of this limitation. However, the paper does not deeply delve into the possible implications of utilizing a small dataset.

The recommendation to address the weakness in this paper is to enrich the dataset. A more extensive and diverse collection of cloud images associated with varying rainfall conditions can be collected from different sources such as meteorological organizations, public database and crowd-sourced data platforms. Apart from that, the authors can enrich the dataset by adding more images that contains diverse meteorological conditions, geographical regions and rainfall intensities. This may help to mitigate the limitation of a small dataset and fortifying the model's predictive capabilities.

2.2.5 Prediction of Rainfall Using Image Processing

In this paper, Kaviarasu et al. 2010 [18] proposed an approach to predict rainfall using digital cloud images as it is more cost-effective compared to satellite images. The methodology used Cloud Mask Algorithm to determine the sky's status and identify cloud formations. Then, K-Means Clustering technique is utilized to classify different types of clouds with a particular focus on Nimbostratus and Cumulonimbus clouds. The authors suggest that the type of rainfall cloud can be predicted by analysing the cloud image colour and density. Overall, the proposed approach offers a solution for individual to predict rainfall by simply capturing cloud images and obtaining details about the rainfall conditions through the image captured.

The strength of this paper lies on its innovative approach as it used novel methods such as Wavelet and K-Means Clustering to recognize cloud types and estimate rainfall. It offers potential improvements over traditional techniques such as Law Textures. This is proved by the results of the experiment carried out by the authors. The accuracy of the proposed method has achieved higher accuracy compared to the traditional method.

The paper did not discuss on the sensitivity of the proposed system to the environmental factors. Factors such as pollution or local geographic features might affect the cloud image analysis and the accuracy of rainfall predictions.

To solve the problem contained in this paper, environmental sensitivity analysis can be conducted to assess how various environmental factors impact the accuracy of cloud image analysis and rainfall predictions. This can be done by collecting data from diverse environmental conditions.

2.2.6 Rainfall Detection in Image using Convolutional Neural Network

Luqman Hakim et al. 2020 [19] introduced an innovative approach for rain detection in images using convolutional neural network (CNN). This approach involves the recognition and classification of images into distinct categories, specifically 'rainy day' and 'clear day'. The dataset for this research comprises CCTV images as it helps to gain a larger amount of rainfall measurement location. Additionally, a rainfall detection experiment is conducted to systematically evaluate the effectiveness of the proposed method.

The strength of the proposed method lies on its robustness to parameter changes. In the experiment, five training processes that conducted using different parameter values has revealed consistently high accuracy values which range from 82.68% to 98.30%. This consistent performance across diverse parameter configurations implies its reliability and adaptability to varying conditions.

The training and validation datasets employed in the study are relatively small in scale, comprising 50 images for training and 20 images for validation. The limited size of datasets may raise concerns about the potential for overfitting where the model might struggle to generalize to new and unseen data.

The weakness can be addressed by applying data augmentation techniques to artificially increase the size of the dataset. This could be done by creating additional training examples through transformations such as rotation, flipping and scaling.

2.2.7 Comparison of the Classification Methods

Table 2.2 Comparison table for classifications methods.

Literature Review	**Method Used**	**Strength**	**Weakness**
2.2.1	● Local Binary Pattern (LBP) + Local Ternary Pattern (LTP)	● The proposed method can overcome the problem of noise-sensitive and achieve high accuracy.	● The limited parameter optimization for LTP feature could affect the performance of the proposed method.
2.2.2	● MeteCNN model	● The proposal model can effectively learn intricate features of weather phenomena. ● It can offer valuable weather insights for diverse field. ● The authors established new database that consist images of different weather phenomena.	● The dataset's limited size might not fully capture all weather conditions that commonly encountered in real-world environments.

2.2.3	● CloudA – A CNN method	● The proposed method is trained from scratch, it directly learns the relevant features from the images, making it an end-to-end solution.	● The performance of the proposed method was examined in a controlled environment. Thus, the result may not be applicable in real-world scenarios.
2.2.4	● Convolutional Neural Network (CNN) ● Transfer learning technique ● XCeption model	● The authors have applied transfer learning techniques with the CNN model to predict rainfall event. ● The proposed approach is innovative where user can input the sky image at any location to predict rainfall event. Thus, it is applicable in real-world scenarios.	● The authors use small dataset for training and validation. Therefore, the model's performance might be constrained by the dataset size.
2.2.5	● Cloud Mask algorithm ● K-Means Clustering	● The proposed method offers potential improvements on the accuracy over traditional methos such as Law Textures.	● Lack of analysis of the sensitivity of the proposed method on the environmental factors.
2.2.6	● Convolutional Neural	● The proposed method demonstrates	● The training and validation datasets

	Network (CNN)	robustness to parameter changes.	used are small in scale.

2.3 Predictive Algorithm

A predictive algorithm is a computational technique or model used in data analysis, machine learning and artificial intelligence to predict or forecast future events or outcomes by analyzing historical data and patterns [12]. These algorithms will learn from existing data, identify relevant patterns and use this knowledge to make predictions about what might happen in the future.

2.3.1 Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting

The scope of this paper is to forecast the intensity of rainfall in a specific area within a short timeframe. A novel approach has been introduced by Shi et al. 2015 [20] to present precipitation nowcasting as a challenge in forecasting spatiotemporal sequences, where both the input and the prediction target involve spatiotemporal sequences. To tackle this problem, this paper extends the fully connected Long Short-Term Memory (FC-LSTM) model by incorporating the structures involving convolutional elements into the transitions from input to output-to-state. This convolutional LSTM (ConvLSTM) method has been proposed and employed for constructing a model that can be trained end-to-end. The experiment results presented in this paper demonstrate the effectiveness of the ConvLSTM network as it outperforms the FC-LSTM and ROVER algorithm.

This paper successfully applies machine learning techniques to address the challenging problem of precipitation nowcasting, a domain that has traditionally relied on less sophisticated methods. In addition, the introduction of ConvLSTM as an extension of LSTM is a significant contribution. This model not only retains the benefits of FC-LSTM while additionally it is well-suited for handling spatiotemporal data. Thus, this innovation addresses a crucial aspect of precipitation nowcasting.

This paper does not explicitly address how well the ConvLSTM model performs under extreme weather conditions. This is critical for many practical applications as weather forecasting often needs to account for extreme weather events.

Sensitivity analysis can be performed to determine the factors or input features that significantly impact the model's performance during extreme weather events. This can guide efforts to improve model robustness.

2.3.2 Summarization of the Predictive Algorithm

Table 2.3 Summarization of the predictive algorithm.

Literature Review	Method Used	Strength	Weakness
2.3.1	● Convolutional Long Short-Term Memory (ConvLSTM)	● The proposed model has solved the problem of precipitation nowcasting which traditionally relied on less sophisticated method.	● The proposed method does not be examined in different conditions, thus the results might not be directly applicable in real-world scenarios.

In conclusion, the literature review on unsupervised learning for self-labelling, classification methods and predictive algorithms in micrometeorology forecasting provides valuable insights into the challenges faced in the field. The findings highlight the need for innovative approaches to address these challenges, particularly in the context of data collection and labelling, weather type classification and rainfall prediction. While advancements in machine learning techniques offer promising solutions, the existing literature also underscores the complexity and limitations of current methodologies. Overall, the literature review supports the existence of the identified challenges in micrometeorology and emphasizes the importance of further research and development efforts to overcome these obstacles.

Chapter 3
System Methodology/Approach

3.1 Design Specifications

This section provides a concise introduction to the methodology employed in the implementing of the project.

3.1.1 Methodology

The project management approach that involves in this project is agile methodology. This project is divided into distinct phases where each phase is serves as a building block for the subsequent one. This approach not only allows for focused attention on specific aspects but also facilitates iterative development and continual refinement [21]. By doing so, the productivity of the project is likely to increase and enables faster project delivery. This will significantly benefit the project since there is limited time available for the development of the proposed system. Moreover, there are multiple concurrent tasks needed to be completed while developing the system. Therefore, the chosen approach is essential in effectively managing these

constraints, ensuring the project's successful delivery within the limited timeframe. In addition, each phase of the project is associated with a testing process to ensures the deliverables has meet the expectations [22]. Thus, this continuous feedback enables the project to adapt to any changes in requirements, errors and evolutionary needs.

3.1.2 General Work Procedure

The micrometeorology forecasting system follows a systematic workflow that begins with setting up a camera module and initiate the collection of cloud images. Subsequently, the images will be uploaded to cloud service platform for fast data retrieval and automation of the process. The images will then be labelled using cloud computer vision techniques to distinguish between images depicting rainy and non-rainy days. This step will combine manual and rule-based approach to create a ground truth dataset. A classification model will then be trained on the ground truth dataset to automate the image annotation process. Then, a predictive model is trained by leveraging classification techniques to recognize temporal patterns in the images, enabling it to predict the occurrence of rainfall events. The final step involves using both models to predict rainfall events in real-time input images and provide forecasted results to users for informed decision-making. Figure 3.1 shows the block diagram for the general work procedure of this project.

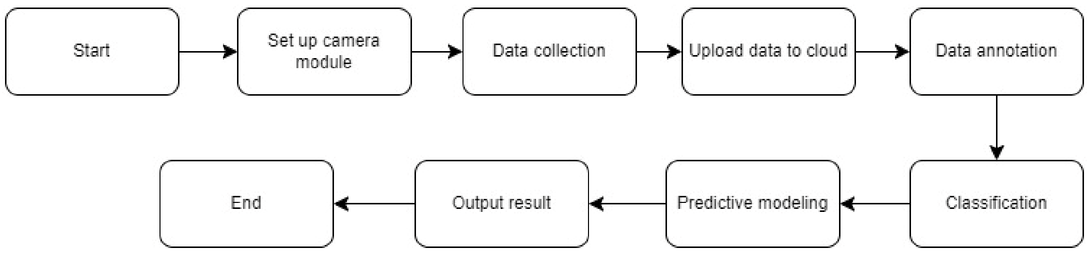


Figure 3.1 Block diagram for general work procedure.

3.2 System Design / Overview

This section contains the system block diagram and the detail explanation of each block in the block diagram. The complete system block diagram is shown in Figure 3.2.

Set up camera module

The system is designed to utilize a camera module for capturing cloud images, forming the foundation of the data collection process. This involves configuring the camera module to capture images at regular intervals, ensuring proper alignment and positioning to capture images from an optimal angle. By meticulously setting up the camera module, the system ensures high-quality image acquisition, essential for accurate analysis in subsequent stages of the workflow.

Data collection

The system is designed to autonomously capture cloud images every 5 minutes between 7AM to 7PM using the configured camera module. This automated process ensures consistent and comprehensive data collection, capturing diverse changes in cloud formations over time. The captured images are saved with a standardized naming convention based on date and time to facilitate organization and retrieval for further analysis. This design ensures a systematic approach to data collection and form the basis for subsequent stages of the workflow.

Upload data to cloud

The system will then automatically upload captured cloud images to a cloud storage platform when connected to the internet. This integration with cloud storage allows for efficient data management and accessibility, enabling seamless collaboration and remote access to the collected data. The system is configured to establish a connection with the cloud storage platform to ensure secure storage and organized archiving of uploaded images. This design collaboration between the system and cloud infrastructure enhances data integrity and availability, ensuring prompt backup and accessibility of captured images for subsequent stages of the workflow.

Retrieve data from cloud

In this step, the system initiates the retrieval process from the cloud storage platform using the command line interface (CLI). It ensures connectivity to the internet and authentication with the cloud storage platform. The retrieval command is executed to fetch the desired data. The system monitors the execution for errors or unexpected behaviour and logs the output for reference. In case of errors, appropriate notifications are provided to the user.

Rule-based labelling

In this step, captured images will undergo a systematic labelling process to determine the weather conditions, specifically focusing on the presence of rain. A predefined rule is established to analyse whether the region of interest (ROI) within the images appears blurry. This analysis is conducted by quantifying the blurriness through the Laplacian value computation within the ROI. If the calculated Laplacian value falls below a predefined threshold, the system will categorize the image as depicting rainy weather. This rule-based approach ensures automated and consistent weather labelling based on objective criteria, which is advantageous when dealing with a large number of images.

Manual labelling

The previous labelled images will undergo a meticulous manual review to identify and correct any potential mislabelling. This manual inspection allows for human judgement to rectify inaccuracies that may have occurred during the automated labelling process. If mislabelled images are identified, their labels are adjusted accordingly to ensure accuracy. By integrating automated rule-based labelling with manual verification, the system achieves a more precise and reliable ground truth dataset. This hybrid approach enhances the integrity of the dataset by incorporating human oversight while minimizing manual effort. Ultimately, this step plays a vital role in refining the accuracy of the dataset, ensuring it accurately reflects the true weather conditions depicted in the images and create a ground truth dataset.

Train and fine-tune classification model

The next step involves training and fine-tuning the classification model using previously created ground truth dataset. The primary goal is to equip the model with the ability to automatically label future captured images by learning the patterns and features from the ground truth dataset. The ground truth dataset will be reorganized into two directories and used to train the model so that it can understand the correlation between visual features and accurate weather label. Throughout the fine-tuning process, the model’s performance will be refined and adjusted based on feedback. This iterative approach enhances the model's accuracy and generalization capabilities, ultimately optimizing its ability to accurately classify weather conditions depicted in the images. Multiple models will be trained in this step to explore various approaches.

Classification model evaluation

The performance of the classification models will undergo rigorous evaluation. This step aims to determine the model's classification key metrics to ensure models' reliability in classifying weather condition depicted in new images. The models will be train iteratively to enhance their accuracies. Then, they will be evaluated to determine the most effective approach for the forecasting system. If the model demonstrates satisfactory results on the testing set, it will be deployed to label future captured images.

Build and train predictive model

This step involves training multiple classification models to forecast the occurrence of future rainfall events based on extracted patterns and features from the training dataset. These models will be trained to recognize patterns indicative of impending rainy conditions in captured images. Through iterative adjustments to the models' parameters, they learn to establish correlations between image features and corresponding weather conditions. This enables the model to classify images before rainy events, effectively recognizing patterns that precede rainfall.

Predictive model evaluation

The previously trained models will undergo an assessment using a testing dataset to compute key metrics such as accuracy, precision and recall to gauge the models' proficiency in predicting the occurrence of rainfall event for new images. Through iterative adjustments and fine-tuning, the models will be optimized to enhance their accuracy and reliability in predicting new images. Following evaluation, the best-performing model will be selected for deployment in forecasting future rainfall events.

Rainfall event prediction

The system will receive new images as input and predictions will be generated based on the previously trained model. The system leverages the model's knowledge acquired during the training phase to make real-time assessment of incoming data and provide valuable insights to the likelihood of future rainfall events.

Output result

The system will provide users with the outcome on whether a rainfall event is anticipated in next few minutes.

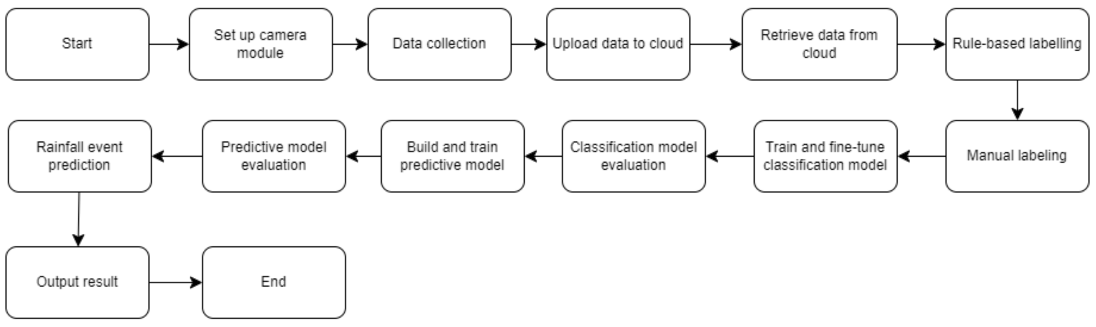

Start
Set up camera module
Data collection
Upload data to cloud
Retrieve data from cloud
Rule-based labelling
Manual labeling
Train and fine-tune classification model
Classification model evaluation
Build and train predictive model
Predictive model evaluation
Rainfall event prediction
Output result
End

Figure 3.2 Complete system block diagram.

3.3 Use Case Design

The use case design section describes how users interact with the system and outlines the specific tasks they can perform [23]. Additionally, it encompasses use case from various perspectives, shedding light on how different stakeholders engage with the system and the tasks pertinent to their roles. Furthermore, it illustrates the system's handling of input from users and the external environment to produce outputs or responses tailored to each stakeholder's needs.

3.3.1 Use Case Diagram from System Developer's Perspective

This subsection provides an insight into the system's functionality and behaviour as perceived by the developer or creator. Besides that, this subsection also elucidates the underlying architecture and design choices made by the developers to meet the system's requirements and objectives. It offers an overview of the system's behaviour and capabilities from a development standpoint.

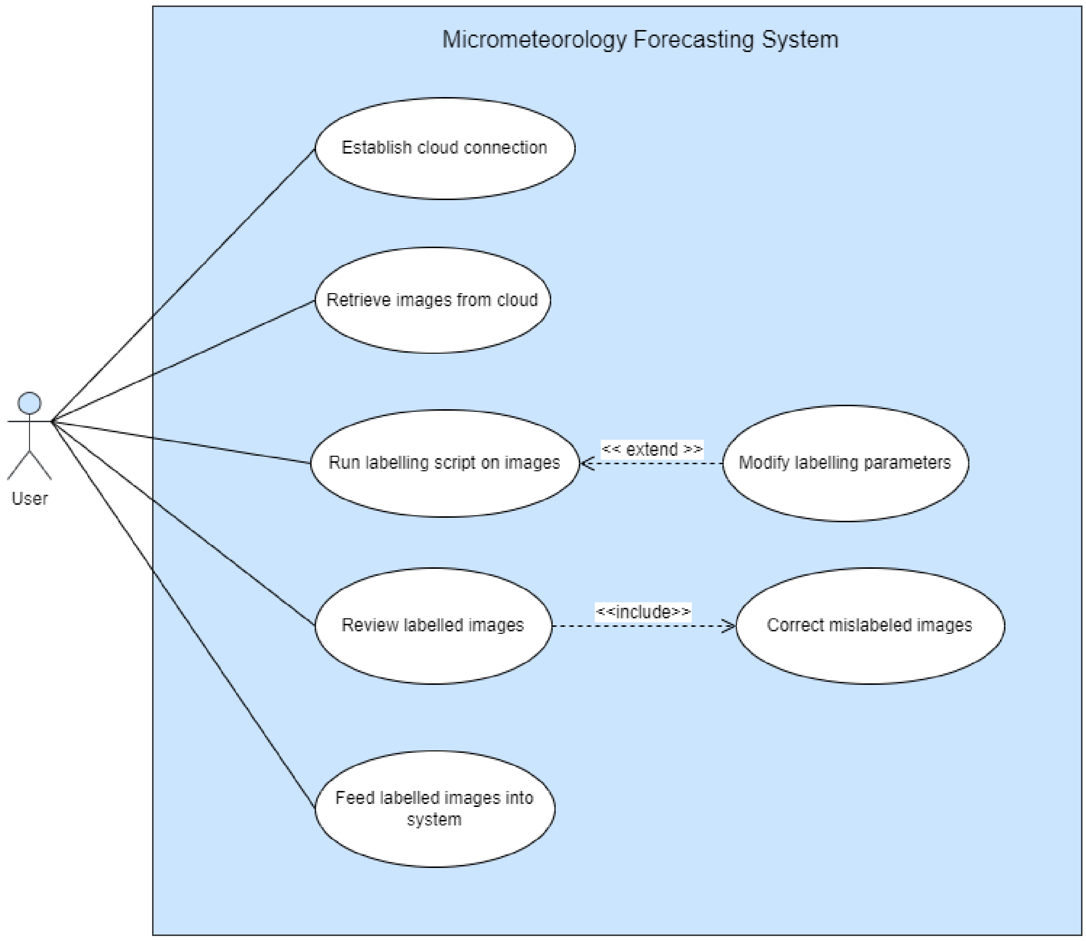

Micrometeorology Forecasting System
Establish cloud connection
Retrieve images from cloud
Run labelling script on images
<< extend >>
Modify labelling parameters
Review labelled images
<<include>>
Correct mislabeled images
Feed labelled images into system
User

Figure 3.3 Use case diagram from system developer's perspective.

3.3.2 Use Case Description

Table 3.1 Use case description – Establish cloud connection.

Use Case ID:	1
Use Case Name:	Establish cloud connection
Brief Description:	This use case describes the process of establishing a connection between the system and the cloud storage platform to enable data transfer and storage.
Actor:	System developer
Preconditions:	1. The system is powered on. 2. The system developer has access credential for the cloud storage platform.
Postconditions:	The system is connected to the cloud storage platform, enabling data transfer and storage.
Basic Flow:	1. The system developer initiates the process of establishing a cloud connection. 2. The system developer provides the authentication credentials that prompts by the system. 3. The system validates the credentials and attempts to establish a connection with the cloud storage platform. 4. If the authentication is successful, the system establishes a secure connection with the cloud storage platform and notifies the system developer that the cloud connection has been successfully established.
Alternate Flows:	-
Exceptions:	1. If the system developer provides invalid authentication credentials: ● The system displays an error message indicating authentication failure and prompt system developer to re-enter the authentication credentials. 2. If there are network issues or server downtime: ● The system displays an error message indicating connection failure. ● The system developer is prompted to retry establishing the connection or check the network status.
Relationship:	-

Table 3.2 Use case description – Retrieve images from cloud.

Use Case ID:	2
Use Case Name:	Retrieve images from cloud
Brief Description:	System developer retrieves data from the cloud storage platform.
Actor:	System developer
Preconditions:	Cloud connection established
Postconditions:	Data successfully retrieved.
Basic Flow:	1. System developer initiates retrieval process and specifies data and destination. 2. System sends retrieval command. 3. Cloud platform retrieves data and transfer to system developer's device. 4. System notifies system developer of completion.
Alternate Flows:	1a. If system developer provides invalid details, system prompts for correction. 4a. If retrieval command fails, system prompts system developer to retry.
Exceptions:	If cloud connection lost, system prompts system developer to reconnect.
Relationship:	

Table 3.3 Use case description – Run labelling script on images.

Use Case ID:	3
Use Case Name:	Run labelling script on images
Brief Description:	System developer executes a labelling script to assign labels to images.
Actor:	System developer
Preconditions:	Images are available for labelling.
Postconditions:	Images are labelled.
Basic Flow:	1. System developer provides input on the images to be labelled. 2. System executes the labelling script on the specified images. 3. System assigns labels to the images based on the script.
Alternate Flows:	-
Exceptions:	-
Relationship:	Extend - Modify labelling parameters

Table 3.4 Use case description – Modify labelling parameters.

Use Case ID:	4
Use Case Name:	Modify labelling parameters
Brief Description:	System developer adjusts parameters used for image labelling.
Actor:	System developer
Preconditions:	-
Postconditions:	Labelling parameters are successfully updated and applied for future image labelling.
Basic Flow:	1. System developer accesses the labelling parameters settings. 2. System presents the current labelling parameters to the system developer. 3. System developer modifies the parameters according to the requirements. 4. System updates the labelling parameters based on the system developer's modification.
Alternate Flows:	-
Exceptions:	-
Relationship:	-

Table 3.5 Use case description – Review labelled images.

Use Case ID:	5
Use Case Name:	Review labelled images
Brief Description:	System developer review labelled images to ensure accuracy and correctness of labelling.
Actor:	System developer
Preconditions:	Labelled images are available for review.
Postconditions:	Labelled images are reviewed and any inaccuracies are corrected.
Basic Flow:	1. System developer accesses the list of labelled images available for review. 2. System presents labelled images to the user for review. 3. System developer examines each labelled image to verify its accuracy. 4. If inaccuracies are found, system developer corrects the labels as needed. 5. System updates the labels based on the system developer's corrections.
Alternate Flows:	-
Exceptions:	-
Relationship:	Include - Correct mislabelled images

Table 3.6 Use case description – Correct mislabelled images.

Use Case ID:	6
Use Case Name:	Correct mislabelled images
Brief Description:	System developer corrects any mislabelled images identified during the review process.
Actor:	System developer
Preconditions:	Mislabelled images are identified during the review process.
Postconditions:	Mislabelled images are corrected.
Basic Flow:	1. System developer identifies and selects a mislabelled image during the review process. 2. System developer updates the label to correct the misclassification. 3. System updates the label based on the system developer’s correction.
Alternate Flows:	-
Exceptions:	-
Relationship:	-

Table 3.7 Use case description – Feed labelled images into system.

Use Case ID:	7
Use Case Name:	Feed labelled images into system
Brief Description:	System developer feed the labelled images into the system to train the classification and predictive models, enabling the system to predict rainfall events.
Actor:	System developer
Preconditions:	Classification and predictive models are available in the system
Postconditions:	Classification and predictive models are trained.
Basic Flow:	1. System developer selects labelled images to be fed into the system for training. 2. System receives the labelled images and preprocesses them for training. 3. System trains the classification and predictive models using the labelled images. 4. Upon completion of training, the system updates the models with the newly acquired knowledge.
Alternate Flows:	-
Exceptions:	-
Relationship:	-

3.3.3 Use Case Diagram from Researcher's Perspective

This section highlights the potential interactions and functionalities of the system from the viewpoint of researchers, scientists or data analyst who intend to utilize the system to collect cloud images and perform research-related tasks. It represents the envisioned use of the system, which may not have been fully developed yet.

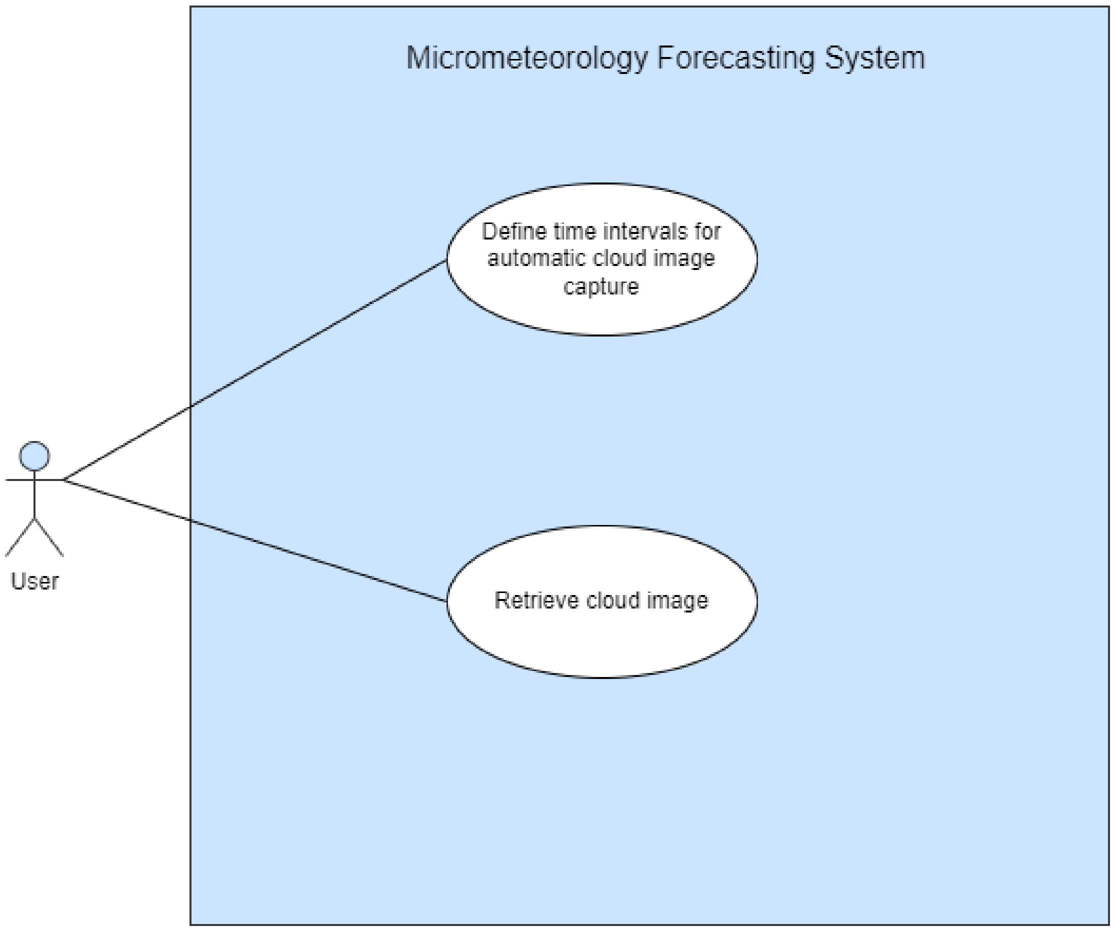

Micrometeorology Forecasting System
Define time intervals for automatic cloud image capture
Retrieve cloud image
User

Figure 3.4 Use case diagram from researcher's perspective.

3.3.4 Use Case Description

Table 3.8 Use case description – Define time intervals for automatic cloud image capture.

Use Case ID:	1
Use Case Name:	Define time intervals for automatic cloud image capture
Brief Description:	Researchers specify time intervals such as the frequency and duration within the system for automatic capture of cloud images.
Actor:	Researcher
Preconditions:	Researcher has access to the system.
Postconditions:	Time intervals are successfully defined in the system.
Basic Flow:	1. Researcher navigates to the 'Define Time Intervals' section. 2. Researcher specifies the frequency and duration for cloud image capture. 3. Researcher submits the defined time intervals. 4. System validates and saves the defined time intervals. 5. System confirms successful saving of time intervals.
Alternate Flows:	3a. Researcher specifies invalid time intervals. ● System prompts the researcher to correct the input. ● Researcher revises the time intervals. ● Proceed to step 3.
Exceptions:	-
Relationship:	-

Table 3.9 Use case description – Retrieve capture cloud images.

Use Case ID:	2
Use Case Name:	Retrieve capture cloud images
Brief Description:	Researchers access the system to retrieve cloud images.
Actor:	Researcher
Preconditions:	Researcher has access to the system.
Postconditions:	Cloud images are successfully retrieved from the system.
Basic Flow:	1. Researcher navigates to the 'Retrieve Cloud Images' section. 2. Researcher selects the desired time interval for image retrieval. 3. System retrieves the cloud images captured during the specified time interval. 4. System displays the retrieved cloud images to the researcher. 5. Researcher downloads the images from the system. 6. Researcher performs analysis or further processing on the retrieved images.
Alternate Flows:	3a. Researcher selects an invalid time interval. ● System notifies the researcher of invalid selection. ● Researcher selects a valid time interval. ● Proceed to step 3.
Exceptions:	-
Relationship:	-

3.3.5 Use Case Diagram from End User's Perspective

This section provides an overview of the potential interactions and functionalities of the system from the viewpoint of end users, such as individuals seeking rainfall forecasts and updates. It outlines the envisioned use of the system, which have not been fully developed yet. The use cases described in this section reflect the intended functionalities and interactions with the system from the end user's perspective.

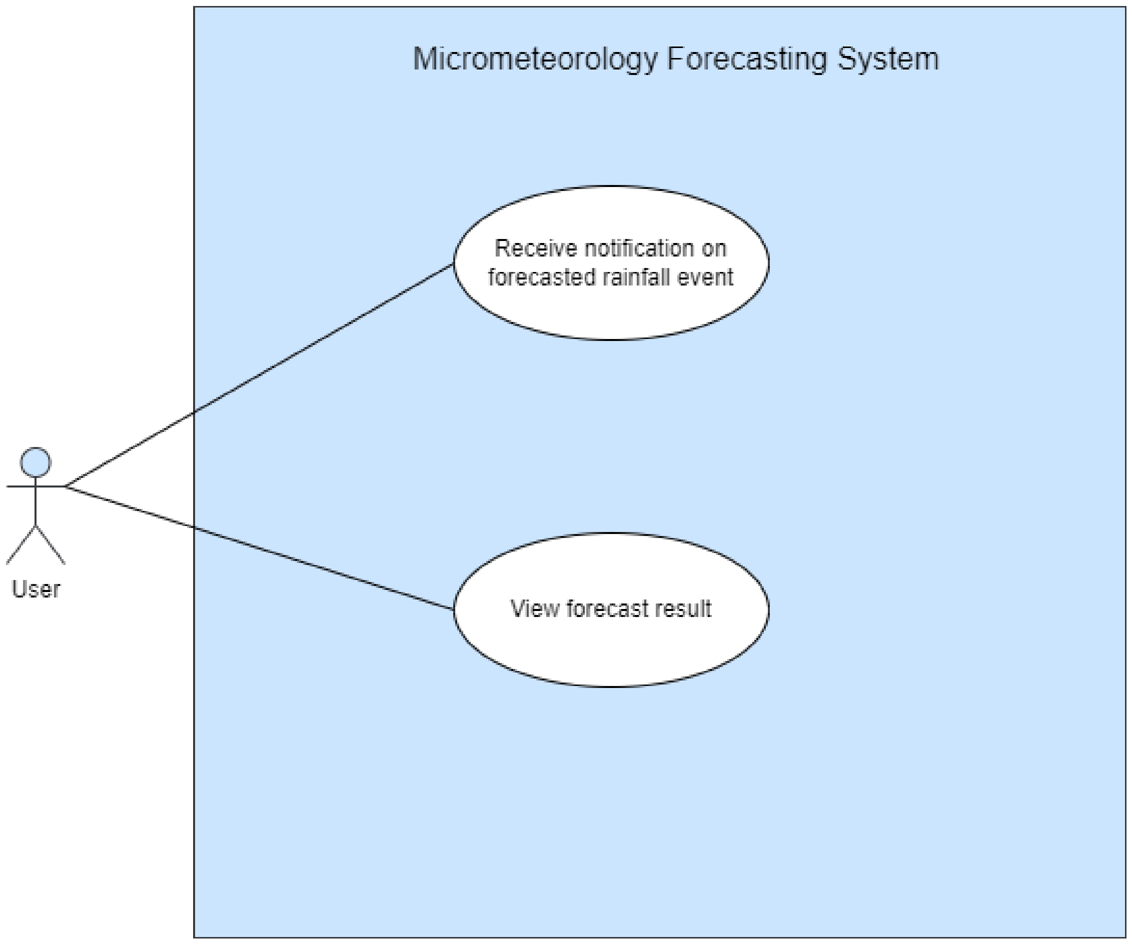

Micrometeorology Forecasting System
Receive notification on forecasted rainfall event
View forecast result
User

Figure 3.5 Use case diagram from end user's perspective.

3.3.6 Use Case Description

Table 3.10 Use case description – Receive notification on forecasted rainfall event.

Use Case ID:	1
Use Case Name:	Receive notification on forecasted rainfall event
Brief Description:	The general user receives notifications when rainfall event is predicted.
Actor:	General user
Preconditions:	-
Postconditions:	-
Basic Flow:	1. System monitors the captured cloud images to forecast rainfall event. 2. System sends notification to user.
Alternate Flows:	-
Exceptions:	-
Relationship:	-

Table 3.11 Use case description – View forecast result.

Use Case ID:	2
Use Case Name:	View forecast result
Brief Description:	The general user views the forecast results.
Actor:	General user
Preconditions:	-
Postconditions:	User gains access to the forecast result.
Basic Flow:	1. User accesses the system. 2. User navigates to the forecast section. 3. System displays forecast results.
Alternate Flows:	-
Exceptions:	-
Relationship:	-

3.4 Activity Diagram

This section presents activity diagrams from two distinct viewpoints, which is system developer and end users. These diagrams are essential tools in both developing and comprehending software systems, providing visual depictions of activity flows within the system.

3.4.1 Activity Diagram from System Developer's Perspective

This subsection provides the perspective of system developers engaged in the development and construction of the system. The activity diagram created from the system developer viewpoint offers valuable insights into the internal mechanisms of the software, detailing how the system is built. Figure 3.6 shows the activity diagram from system developer's perspective.

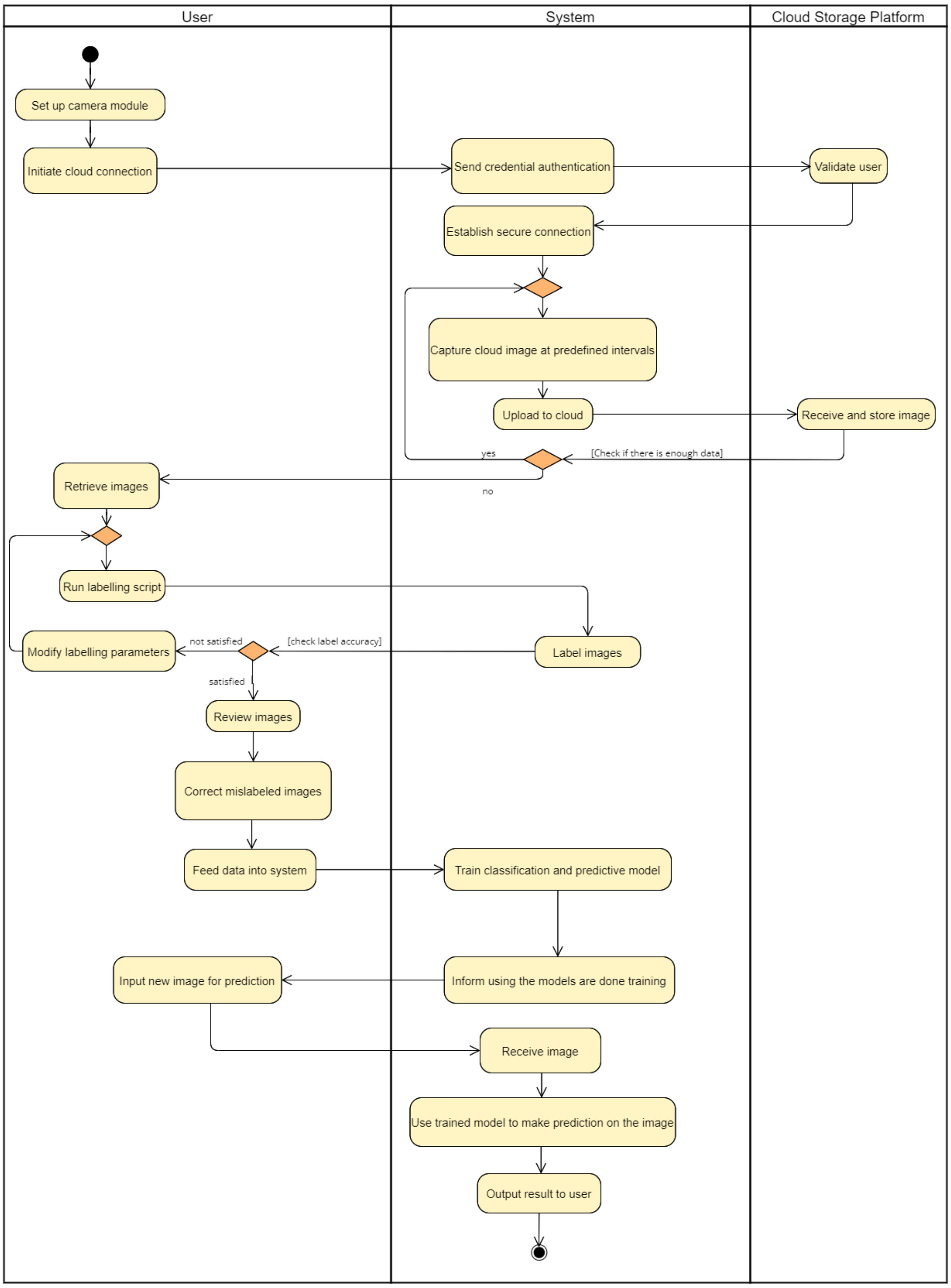

User
System
Cloud Storage Platform
Set up camera module
Initiate cloud connection
Send credential authentication
Validate user
Establish secure connection
Capture cloud image at predefined intervals
Upload to cloud
Receive and store image
yes
[Check if there is enough data]
no
Retrieve images
Run labelling script
Label images
[check label accuracy]
not satisfied
Modify labelling parameters
satisfied
Review images
Correct mislabeled images
Feed data into system
Train classification and predictive model
Inform using the models are done training
Input new image for prediction
Receive image
Use trained model to make prediction on the image
Output result to user

Figure 3.6 Activity diagram from system developer's perspective.

3.4.2 Activity Diagram from End User's Perspective

The activity diagram presented from the end user's perspective offers a simplified representation of the potential interaction between the users with the system. This perspective provides valuable insights into usability of the software, enabling developers to optimize the system for intuitive and efficient interaction. Figure 3.7 shows the activity diagram from end user's perspective.

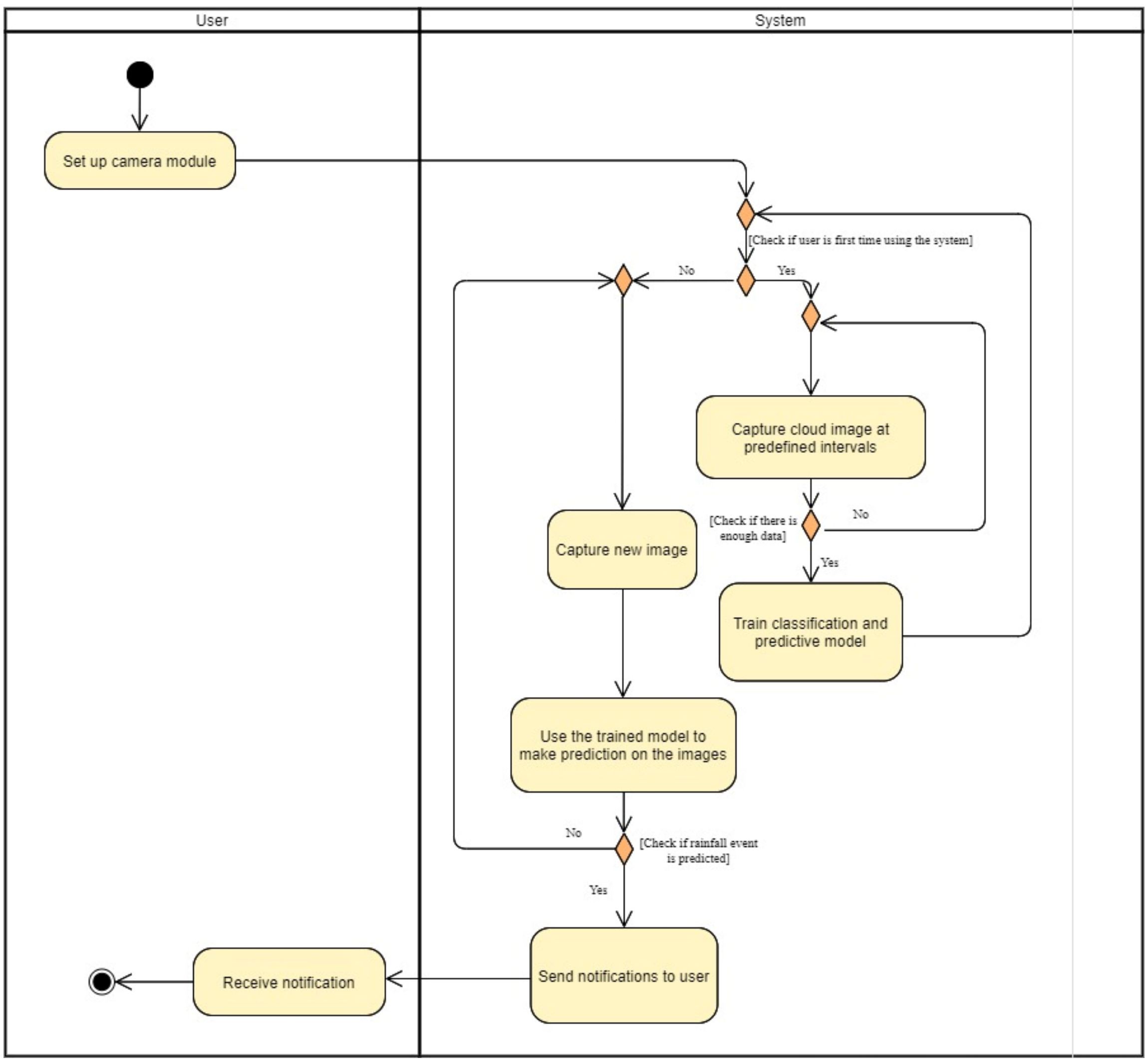

User
System
Set up camera module
[Check if user is first time using the system]
No
Yes
Capture cloud image at predefined intervals
[Check if there is enough data]
No
Yes
Capture new image
Train classification and predictive model
Use the trained model to make prediction on the images
No
[Check if rainfall event is predicted]
Yes
Receive notification
Send notifications to user

Figure 3.7 Activity diagram from end user's perspective.

3.5 System Architecture Diagram

First and foremost, the system is starts by capturing cloud images using a camera module. Then, a hybrid labelling approach that integrates rule-based and manual labelling methods will be employed to label the captured images. This hybrid approach ensures the accuracy and effectiveness when dealing with a large amount of data. The rule-based labelling is employed by leveraging machine learning techniques to label the images. This method is particularly advantageous for handling large datasets efficiently within a short period. Subsequently, the manual labelling method is used to verify the correctness of the images' labels and to rectify any mislabelled images. Hence, this method is utilized to ensure the correctness of all labels within the dataset. The ground truth dataset that created from the previous step will then be used to train a VGG16 classification model. VGG16 is a classification model and is employed to classify the cloud images. After classifying the dataset, a Xception model will be trained to recognize the patterns in cloud images that typically precede rainfall. The implementation of all these processes will result in a micrometeorology forecasting system that fit to a specific geographical region. Lastly, the system will capture new cloud images at fixed time intervals and return the rainfall event prediction result. Figure 3.8 shows the architecture diagram of the system.

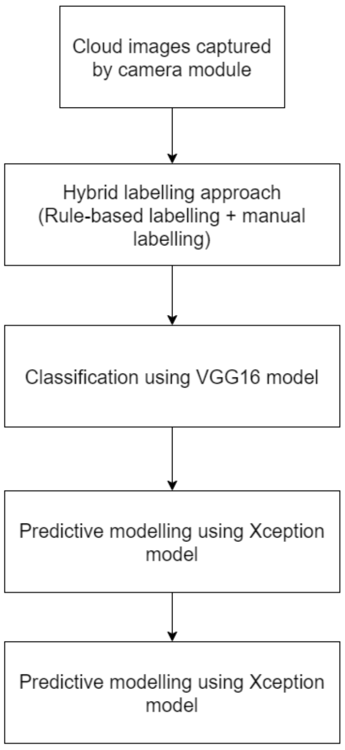

Cloud images captured by camera module
Hybrid labelling approach (Rule-based labelling + manual labelling)
Classification using VGG16 model
Predictive modelling using Xception model
Predictive modelling using Xception model

Figure 3.8 Architecture diagram of micrometeorology forecasting system.

3.6 Project Timeline

This section highlights the system's development timeline and presents the distribution of rainy and non-rainy images collected during the initial 15 days of data collection period. Figure 3.9 shows the project timeline.

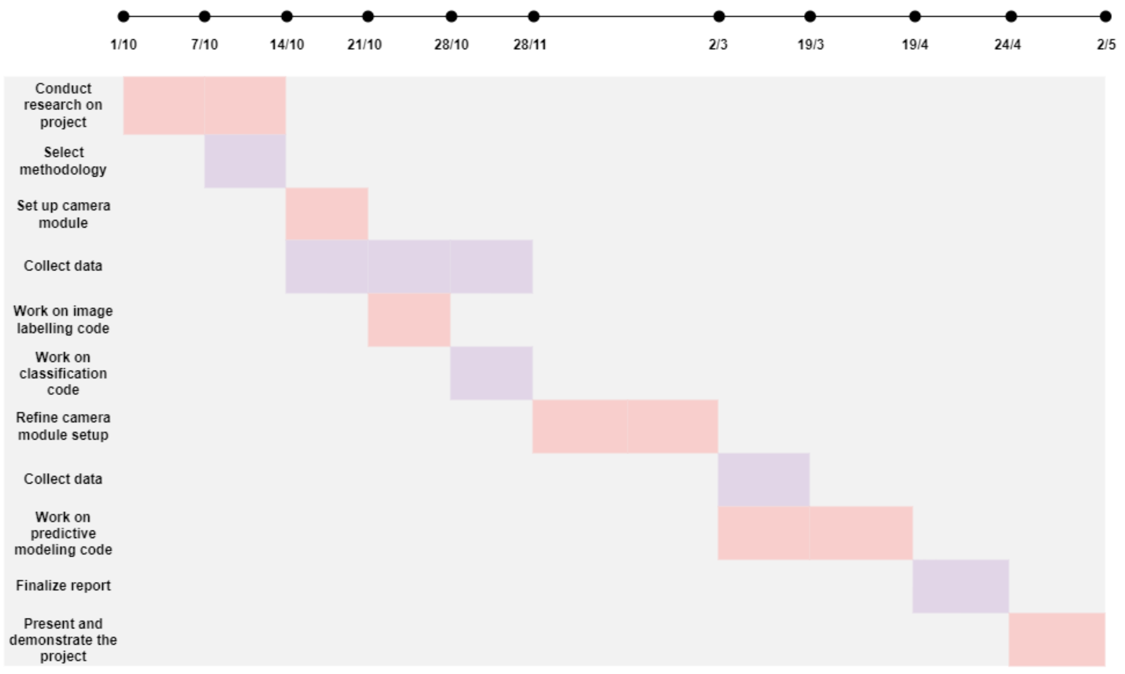


Figure 3.9 Project timeline.

Additionally, a bar chart is generated to depict the data collected during the initial 15 days for both classes using the latest camera setup method. This visualization provides insights into the distribution of rainy and non-rainy images over the specified timeframe.

The initial days show a lower volume of data collected, primarily attributed to an overheating issue with the wire extension, leading to automatic electrical cutoffs. This happened particularly during periods of high temperature such as noon. Therefore, some modifications have been made to the camera module setup and has successfully resolved this issue, resulting in consistent data collection.

Furthermore, an observation reveals a disparity between the number of rainy and non-rainy images, with fewer rainy instances recorded. This discrepancy can be attributed to the predominant non-rainy weather conditions during the data collection period. Figure 3.10 shows the distribution of rainy and non-rainy images over the initial 15 days with latest camera setup.

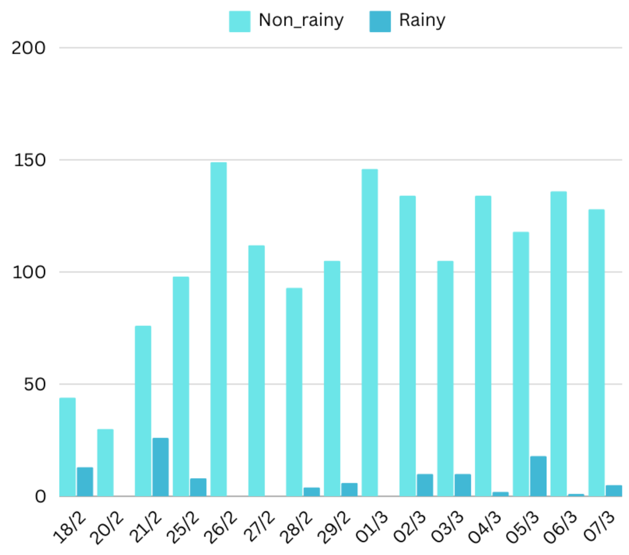


Figure 3.10 Distribution of rainy and non-rainy images over the initial 15 days of data collection process.

Chapter 4
System Implementation

This section offers an overview of the implementation process undertaken in this project. It outlines the key steps and methodologies employed to develop and deploy the system. Each stage of the implementation is discussed in detail, highlighting the strategies, techniques, and technologies utilized. Additionally, insights into the challenges encountered and the solutions devised to overcome them are presented, providing a comprehensive understanding of the system's development journey.

4.1 Tools and Technologies Involved

Hardware

The hardware involved in this project is computer, a Raspberry Pi single-board computer and a camera module. The laptop is the main component is this project which serves as the primary tool for developing the micrometeorology forecast system. Additionally, the Raspberry Pi is utilized to configure and operate the camera module. This enables the camera module to capture the cloud images and provide real-time data to the system. Tables below shows the hardware specifications of the laptop, Raspberry Pi single-board computer and camera module.

Table 4.1 Specifications of laptop.

Description	Specifications
Model	Asus Tuf Gaming A15 FA506I
Processor	AMD Ryzen 5-4600H
Operating System	Windows 11
Graphic	NVIDIA GeForce GTX 1650 GDDR6 4GB
Memory	8GB DDR4 3200Mhz RAM
Storage	512GB NVMe PCIe SSD

Table 4.2 Specifications of Raspberry Pi single-board computer.

Description	Specifications
Model	Rasberry Pi 3 Model B+
Processor	Cortex-A53 (ARMv8) 64-bit SoC
Memory	1GB RAM
Storage	16GB MicroSD card

Table 4.3 Specifications of camera module.

Description	Specifications
Model	Raspberry Pi 8MP Camera Module V2.1
Image Sensor	Sony IMX219
Physical Dimensions	25mm x 23mm x 9mm
Interface	CSI connector
Supported OS	Raspbian

4.2 Setting up

4.2.1 Software

Before starting to develop the micrometeorology forecasting system, two software applications must be downloaded and installed in my laptop.

1. RealVNC Viewer
   - A software application that facilitates remote access and control of the Raspberry Pi and camera module. It provides a graphical interface to interact with the desktop of the Raspberry Pi.
2. Anaconda Navigator (anaconda3)
   - A graphical user interface (GUI) included with the Anaconda distribution. It serves as a platform for managing and launching various tools and environments such as Jupyter Notebook.
3. Jupyter Notebook (anaconda 3)
   - An open-source web application used as the primary coding platform in this project. It facilitates interactive and collaboration development of code, data analysis and visualization.

4. AWS Cloud Storage Platform
   - A cloud storage platform that offers scalable and secure storage solutions. It facilitates efficient data management and accessibility, supporting the storage and analysis of data.

4.2.2 Set Up of Camera Module

The camera module setup serves as the foundational step in data collection. This involves physically connecting the camera module a Raspberry Pi single-board computer to establish a seamless interface for capturing cloud images. Precise alignment and placement of the camera module are crucial to optimize image capture and ensure accurate representation of the sky. Careful consideration is given to factors such as angle, elevation and orientation to maximize the visibility of cloud formations. This can avoid some problems such as backlighting, unobvious representation of rain drop and light expose problem. Besides that, the Raspberry Pi's power supply is facilitated through the use of a 5-meters and 15-meters wire extensions. These extensions enable the camera module to be deployed in various locations while maintaining a reliable power source. Additionally, a timer plug is utilized to automate power management, allowing for scheduled operation and energy efficient. This ensures that the camera module is powered on and off at designated times, streamlining the data collection process and conserving energy resources. The configuration allows the entire setup to operate effectively in outdoor conditions. Figure 4.1 shows the setup of camera module with the wire extension and Figure 4.2 shows the timer plug connecting to the adapter of the wire extension.

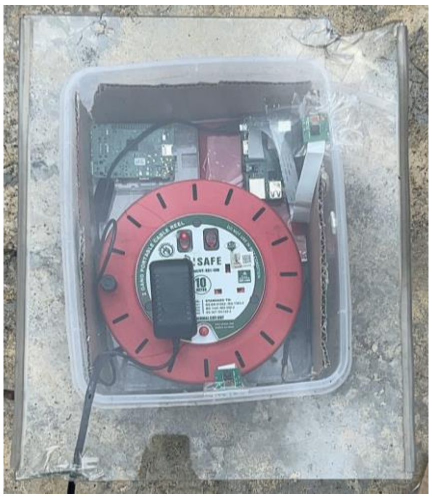


Figure 4.1 Setup of camera module with wire extension.

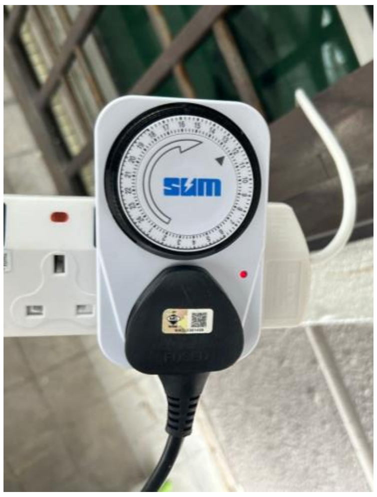


Figure 4.2 Timer plug connected to the wire extension adapter.

The connection between the Raspberry Pi and the laptop is made through the use of the RealVNC Viewer software. Calibration processes are conducted to fine-tune camera settings, such as focus and expose to achieve optimal image quality.

Additionally, environmental considerations play a role in camera module setup. Factors such as weatherproofing and stability are addressed to safeguard the equipment from environmental elements to ensure reliable and consistent data collection over time. To achieve this, the camera module is securely housed within a protective box, featuring a glass cover with a trim strip along its four borders. This design prevents raindrops from entering the box and shields the equipment from adverse weather conditions. To address potential backlighting and overheat issues caused by sunlight, a practical solution is implemented. Cardboard sheets are strategically placed above and around the Raspberry Pi and the wire extension to act as shields, effectively blocking direct sunlight and reducing heat buildup. This arrangement helps optimize image capture by minimizing glare and shadows caused by harsh lighting conditions. By enclosing both the Raspberry Pi and the wire extension within this protective enclosure, the system ensures reliable data collection for the system. Figure 4.3 shows the complete setup of the camera module.

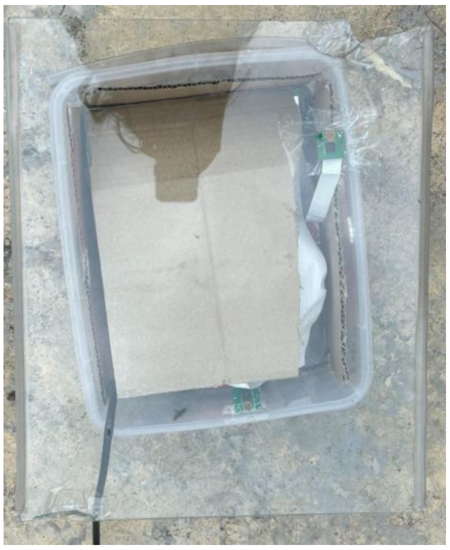


Figure 4.3 Complete setup of the camera module.

4.3 Data Collection

In this process, the Raspberry Pi is configured to orchestrate the image capture process systematically. This is achieved through the development of a Python script that serves as the control mechanism for the camera module. The script is saved in the Raspberry Pi and prompt the camera module to capture images at predefined intervals, specifically from 7:00 AM to 7:30 PM. The capture frequency is set to every 5 minutes during the timeframe.

Furthermore, the script includes instructions to save each captured image with a timestamp-based nomenclature. The timestamp comprised of both date and time information to identify the capture time for each image. This enabled future reference and used when it comes to select the suitable data to train the predictive model. The python script ensure the Raspberry Pi operates autonomously to capture, name and store cloud images in a structured and time-sensitive manner, laying the groundwork for subsequent data processing and analysis. The 'nohup' Linux command will be employed to run the script in the background. Figure 4.4 shows the examples of the data collected.

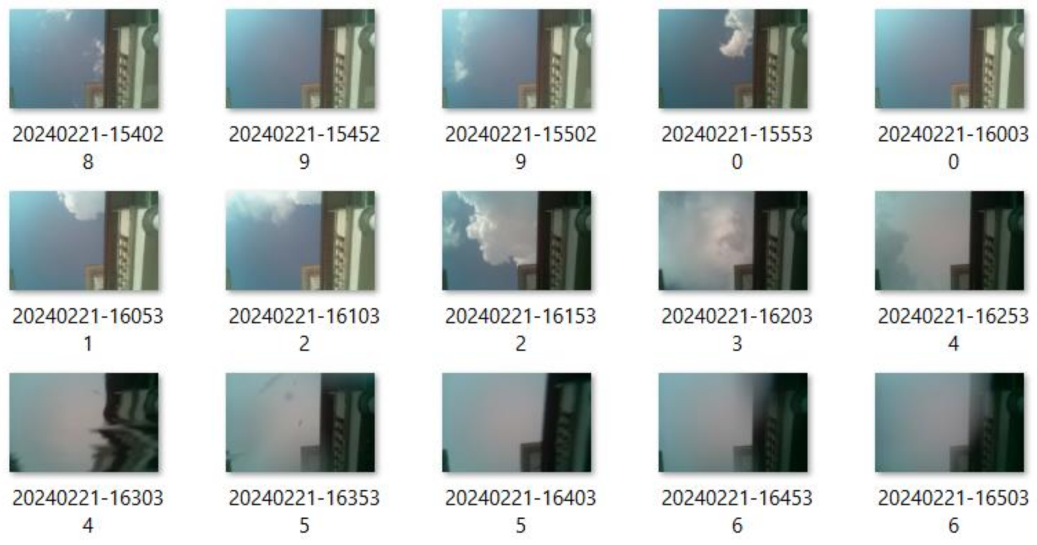
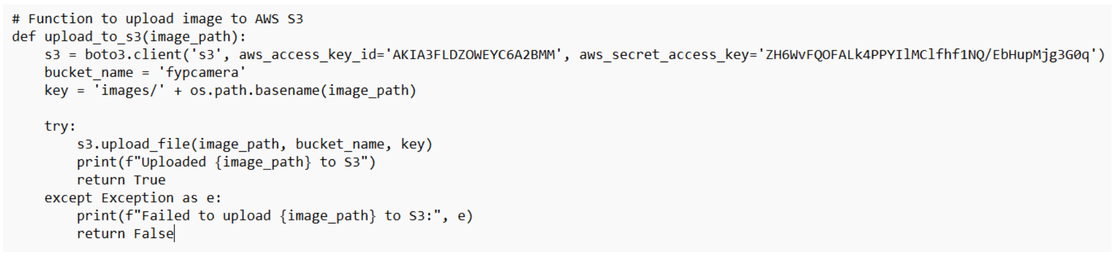


Figure 4.4 Examples of data collected.

4.4 Upload Data to Cloud

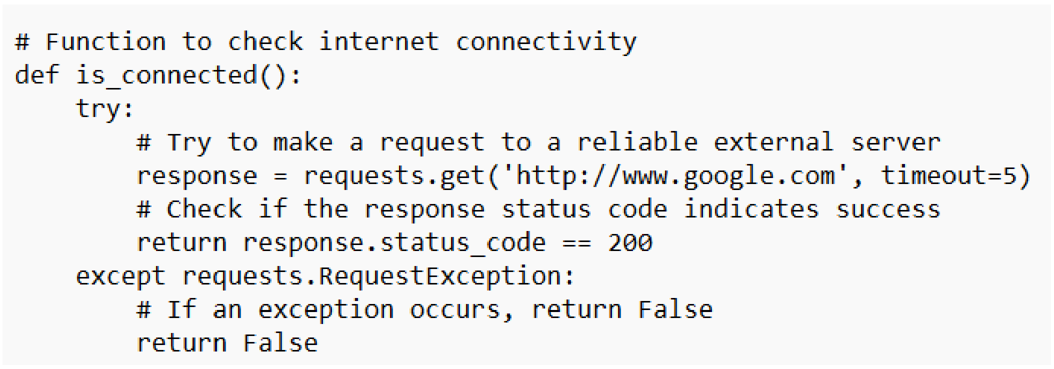
To ease the data retrieval process, AWS Cloud Storage Platform has been utilized to ensure automate and seamless transfer of captured cloud images to the cloud. To achieve this, initial connection will need to be established between Raspberry Pi and the AWS Cloud. This is done by entering the authentication credentials in the script. Figure 4.5 shows the function to upload image to AWS cloud storage platform.



```python
# Function to upload image to AWS S3
def upload_to_s3(image_path):
    s3 = boto3.client('s3', aws_access_key_id='AKIA3FLDZOWEYC6A2BMM', aws_secret_access_key='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')
    bucket_name = 'fypcamera'
    key = 'images/' + os.path.basename(image_path)

    try:
        s3.upload_file(image_path, bucket_name, key)
        print(f"Uploaded {image_path} to S3")
        return True
    except Exception as e:
        print(f"Failed to upload {image_path} to S3:", e)
        return False
```

Figure 4.5 Function to upload image to AWS cloud storage platform.

After establishing the initial connection, the system is configured to automatically upload captured images once the Raspberry Pi is connected to the internet. A Python script has been used to manages the upload process, leveraging the AWS Command Line Interface (CLI) to securely transfer the images to a designated bucket in the cloud. The function to check the internet connectivity is shown in Figure 4.6, while the function to save the image path to a file is shown in Figure 4.7. Other than that, Figure 4.8 shows the function to upload pending images to the cloud once it is connected to internet.



```python
# Function to check internet connectivity
def is_connected():
    try:
        # Try to make a request to a reliable external server
        response = requests.get('http://www.google.com', timeout=5)
        # Check if the response status code indicates success
        return response.status_code == 200
    except requests.RequestException:
        # If an exception occurs, return False
        return False
```

Figure 4.6 Function to check the internet connectivity.

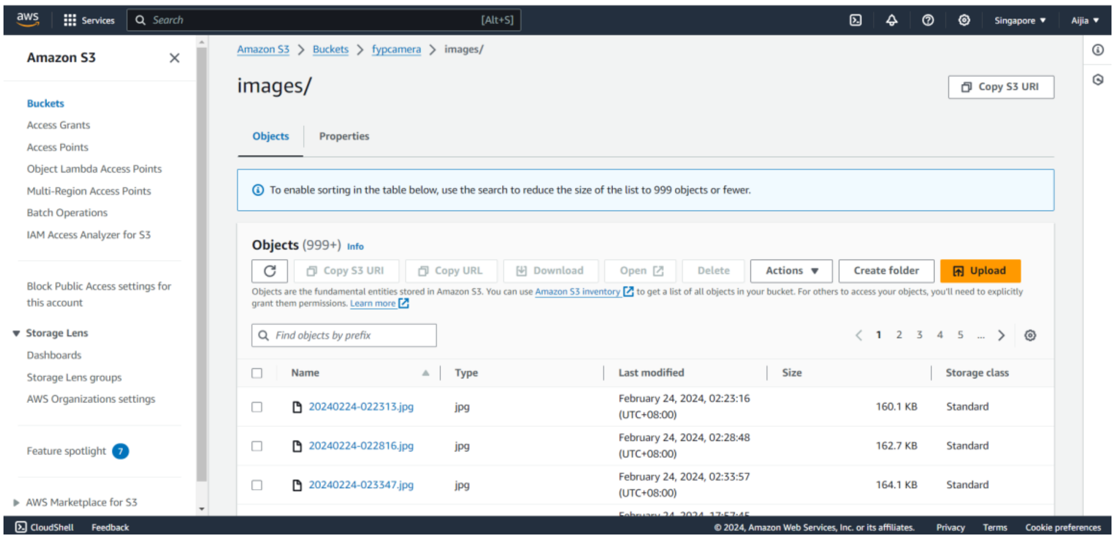
```
# Function to save image path to a file
def save_image_path(image_path):
    with open('pending_images.txt', 'a') as file:
        file.write(image_path + '\n')
```

Figure 4.7 Function to save image path to a text file.

```
# Check internet connectivity
if is_connected():
    # Read pending image paths from file
    with open('pending_images.txt', 'r') as file:
        pending_images = file.readlines()
    # Upload pending images
    for pending_image in pending_images:
        if upload_to_s3(pending_image.strip()):
            # Remove uploaded image from file
            pending_images.remove(pending_image)
    # Write remaining pending images back to file
    with open('pending_images.txt', 'w') as file:
        file.writelines(pending_images)
else:
    print("No internet connection. Images will be uploaded later.")
```

Figure 4.8 Function to upload pending images to AWS cloud storage platform.

This automated workflow eliminates the need for manual intervention, ensuring efficient data management and enabling easy retrieval of data for analysis. Figure 4.9 and Figure 4.10 displays the screenshot of the AWS S3 bucket.



images/

To enable sorting in the table below, use the search to reduce the size of the list to 999 objects or fewer.

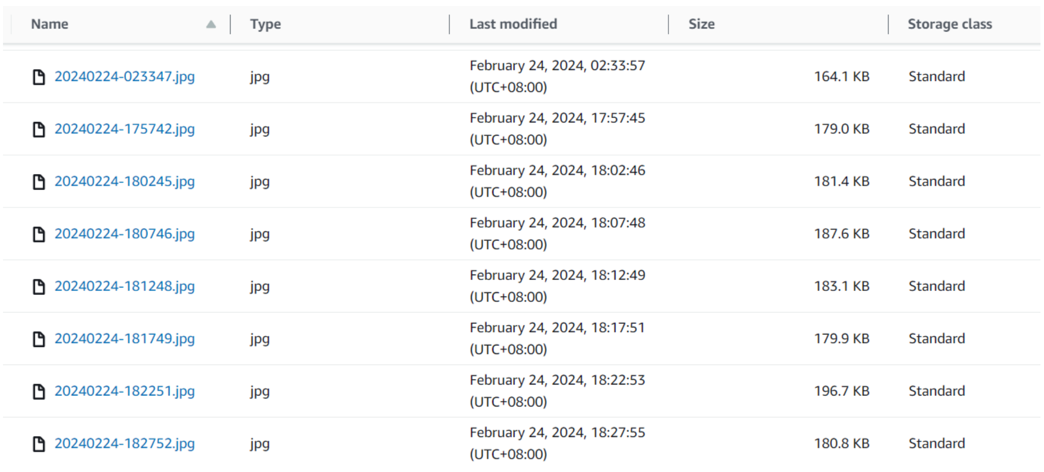
Objects (999+) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
20240224-022313.jpg	jpg	February 24, 2024, 02:23:16 (UTC+08:00)	160.1 KB	Standard
20240224-022816.jpg	jpg	February 24, 2024, 02:28:48 (UTC+08:00)	162.7 KB	Standard
20240224-023347.jpg	jpg	February 24, 2024, 02:33:57 (UTC+08:00)	164.1 KB	Standard

© 2024, Amazon Web Services, Inc. or its affiliates.

Figure 4.9 First screenshot of the AWS S3 bucket.

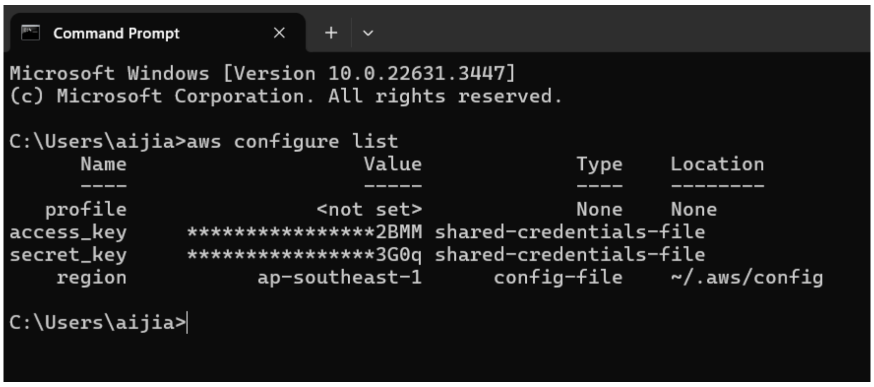


Name	Type	Last modified	Size	Storage class
20240224-023347.jpg	jpg	February 24, 2024, 02:33:57 (UTC+08:00)	164.1 KB	Standard
20240224-175742.jpg	jpg	February 24, 2024, 17:57:45 (UTC+08:00)	179.0 KB	Standard
20240224-180245.jpg	jpg	February 24, 2024, 18:02:46 (UTC+08:00)	181.4 KB	Standard
20240224-180746.jpg	jpg	February 24, 2024, 18:07:48 (UTC+08:00)	187.6 KB	Standard
20240224-181248.jpg	jpg	February 24, 2024, 18:12:49 (UTC+08:00)	183.1 KB	Standard
20240224-181749.jpg	jpg	February 24, 2024, 18:17:51 (UTC+08:00)	179.9 KB	Standard
20240224-182251.jpg	jpg	February 24, 2024, 18:22:53 (UTC+08:00)	196.7 KB	Standard
20240224-182752.jpg	jpg	February 24, 2024, 18:27:55 (UTC+08:00)	180.8 KB	Standard

Figure 4.10 Second screenshot of the AWS S3 bucket.

4.5 Retrieve Data from Cloud

To retrieve data from the cloud, this involves establishing a remote connection to access the captured cloud images stored in the cloud storage platform. This remote access eliminates the need for manual transfer of data from the Raspberry Pi, enhancing efficiency and convenience. The system initiates the retrieval process by establishing connectivity with the cloud storage platform using appropriate authentication credentials. Once connected, the system sends a retrieval command specifying the data to be retrieved, triggering the cloud storage platform to transfer the captured images to the local storage or memory of the system without manual intervention. Upon successful retrieval, the system will notify the user to ensure they are informed of the completion of the retrieval process. This retrieval mechanism ensures efficient data access for further processing and analysis, enhancing the overall workflow of the system. Figure 4.11 shows the configuration in laptop to establish connectivity with the AWS cloud.



```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aijia>aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************2BMM shared-credentials-file
secret_key     ****************3G0q shared-credentials-file
    region           ap-southeast-1      config-file    ~/.aws/config

C:\Users\aijia>
```

Figure 4.11 Configuration setup on laptop for AWS cloud connectivity.

4.6 Data Annotation

The data annotation process employs a hybrid approach which combine both rule-based labelling and manual labelling methods to enhance the accuracy and efficiency.

4.6.1 Rule-based Labelling

Initially, the rule-based labelling method utilizes machine learning techniques to assess and label images automatically. Laplacian value is calculated to determine whether the image appear blurry in the ROI. If the value falls below the threshold, the image will be labelled as rainy, vice versa. The Laplacian operator is used for edge detector, which highlights the area of rapid intensity change or edges in the image [24]. In simpler terms, it identifies areas where the brightness or colour changes abruptly, indicating the presence of edges or boundaries between objects in the image. When applied to an image, the Laplacian operator produces an output image called the Laplacian of the original image. This outputted image typically contains high values at edges and low values elsewhere, effectively emphasizing the edges and discontinuities present in the original image. If the Laplacian value falls below a predefined threshold, indicating a lack of sharp edges or rapid intensity changes, the image will then be labelled as rainy, as it likely exhibits characteristic of the Laplacian operator to identify visual cues associated with image charity and sharpness.

In this specific example, a region of interest (ROI) representing a balcony is examined for blurriness to check if there is the occurrence of rainfall event. Figure 4.12 illustrates the ROI in non-rainy images, whereas Figure 4.13 showcases the ROI in rainy images, where the balcony will appear blurry in rainy image.

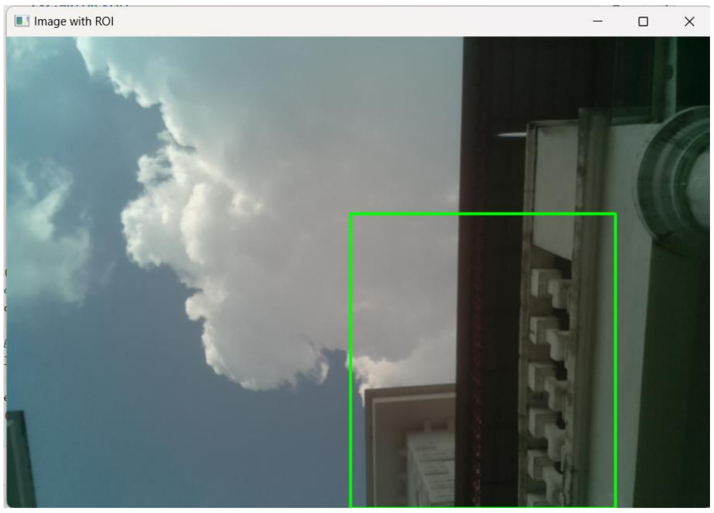


Figure 4.12 ROI in non-rain images.

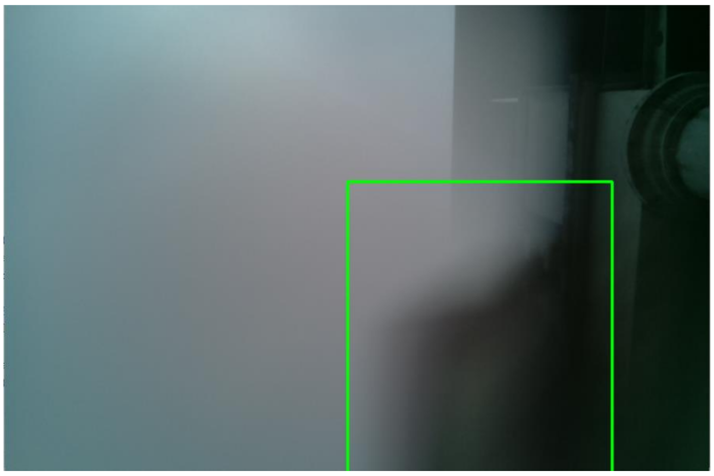


Figure 4.13 ROI in rainy images.

The function will evaluate the Laplacian variance of this region and the dataset will then undergo relabelling, where rainy images will have '-1' appended to the timestamp and non-rainy images will have '-0' added. Figure 4.14 shows the examples of relabelled dataset.

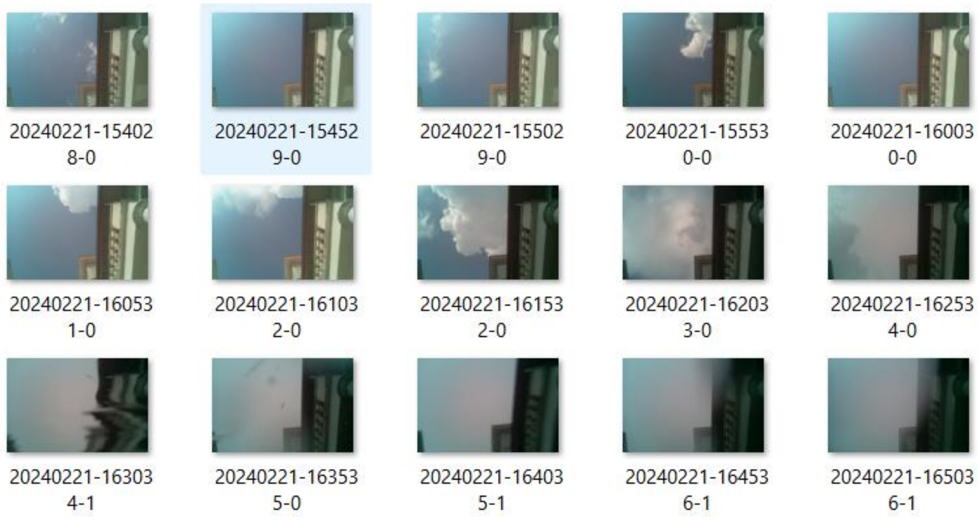


Figure 4.14 Examples of relabelled dataset.

4.6.2 Manual Labelling

Following the rule-based labelling, the dataset undergoes a crucial step of manual annotation. Human reviewers verify the correctness of the automatically assigned labels and rectify any misclassification. This comprehensive strategy results in a labelled dataset that is not only efficiently generated but also reliably accurate, forming the basis for robust model training and evaluation.

4.7 Classification

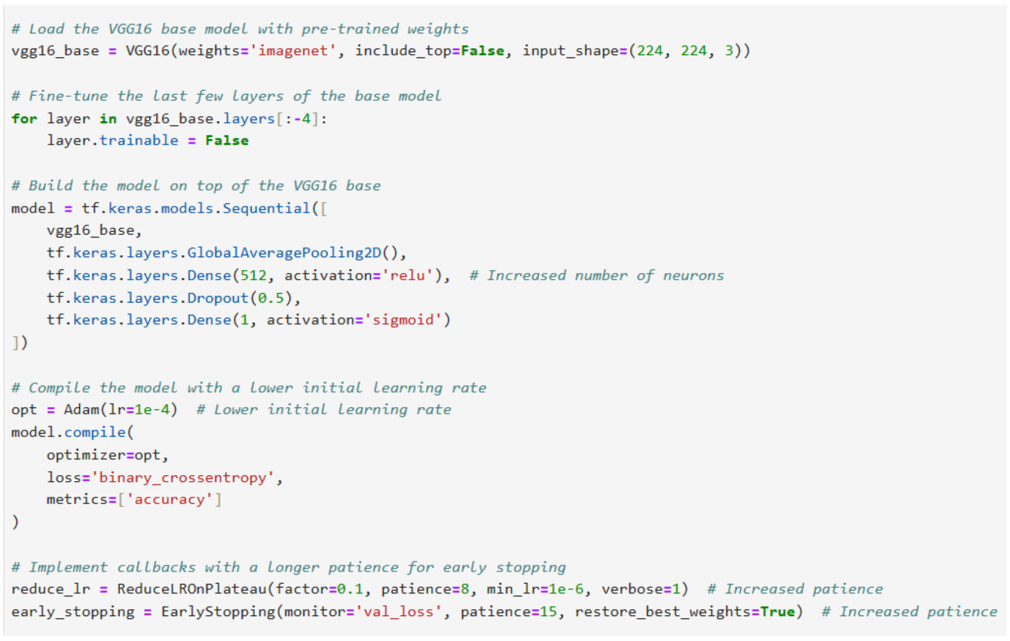
In this step, three models including VGG16, DenseNet121 and Inception-V3 are trained to classify the weather conditions based on the captured cloud images. These models are trained using convolutional neural networks (CNNs) and other classification techniques, with parameters adjusted to optimize their performance. The dataset utilized to train and test the models consists of 419 images for training, 124 images for validation and 105 images for testing.

4.7.1 VGG16 Model

A CNN architecture is implemented by utilizing the VGG16 model as a feature extractor [25]. Then, the pre-trained weights from ImageNet are utilized to leverage learned representations of generic image features, enhancing the model's ability to generalize to new data.

The ground truth dataset is subjected to diverse data augmentation techniques, including rotation, zoom and horizontal flip. This purpose of this step is to enrich the training set and reduce the risk of overfitting.

In the VGG16 model classification code, a dropout layer follows the flattening process will serve as a regularization mechanism to mitigate overfitting by randomly dropping out units during training. Additionally, the model is optimized using the Adam optimizer with an initial learning rate and a Learning Rate Scheduler act as a callback. This Learning Rate Scheduler will dynamically adjust the learning rate throughout the epochs for efficient convergence initially and fine-tuning later. The training process spans 30 epochs, and the model is evaluated using a separate test dataset once the training is complete. In overall, the code combines pre-trained features, data augmentation, regularization and adaptive learning rate strategies to enhance the robustness of the image classification model. Figure 4.15 shows the configuration of the VGG16 model. The result will be discussed in next chapter.



```
# Load the VGG16 base model with pre-trained weights
vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Fine-tune the last few layers of the base model
for layer in vgg16_base.layers[:-4]:
    layer.trainable = False

# Build the model on top of the VGG16 base
model = tf.keras.models.Sequential([
    vgg16_base,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation='relu'),  # Increased number of neurons
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model with a lower initial learning rate
opt = Adam(lr=1e-4)  # Lower initial learning rate
model.compile(
    optimizer=opt,
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Implement callbacks with a longer patience for early stopping
reduce_lr = ReduceLROnPlateau(factor=0.1, patience=8, min_lr=1e-6, verbose=1)  # Increased patience
early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)  # Increased patience
```

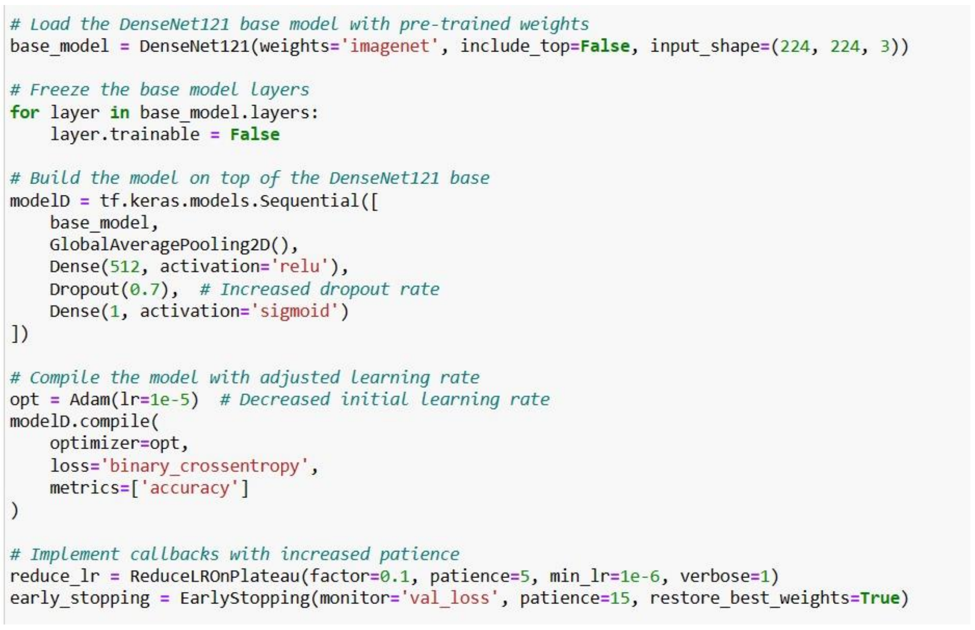
Figure 4.15 Configuration of the VGG16 model.

4.7.2 DenseNet121 Model

DenseNet121, a deep convolutional neural network architecture, is employed due to its pre-training on the ImageNet dataset which enables it to extract intricate features from images effectively [26]. The model architecture features dense connectivity between layers, facilitating enhanced feature reuse and gradient flow throughout the network. To adapt this model to the weather classification task, several techniques are employed.

Firstly, data augmentation is applied to increase the diversify of the training dataset. Transfer learning is leverage by using the pre-trained weights of the DenseNet121 base model, allowing the model to extract relevant features without extensive training from scratch. Furthermore, fine-tuning is performed by freezing the pre-trained base layers and training only the top layers on the task-specific dataset.

Besides that, adjustable learning rates are utilized to facilitate smoother convergence during training, with a ReduceLROnPlateau callback dynamically adjusting the learning rate based on validation performance. Dropout regularization with a higher dropout rate of 0.7 is employed to reduce overfitting and early stopping is implemented to prevent the model from fitting too closely to the training data. This is done by halting training when the validation loss begins to increase, thus ensuring optimal generalization performance. These techniques collectively enable the Densenet121 model to effectively learn from the classification dataset. Figure 4.16 shows the configuration of the DenseNet121 model.



```
# Load the DenseNet121 base model with pre-trained weights
base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Build the model on top of the DenseNet121 base
modelD = tf.keras.models.Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(512, activation='relu'),
    Dropout(0.7),  # Increased dropout rate
    Dense(1, activation='sigmoid')
])

# Compile the model with adjusted learning rate
opt = Adam(lr=1e-5)  # Decreased initial learning rate
modelD.compile(
    optimizer=opt,
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Implement callbacks with increased patience
reduce_lr = ReduceLROnPlateau(factor=0.1, patience=5, min_lr=1e-6, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)
```

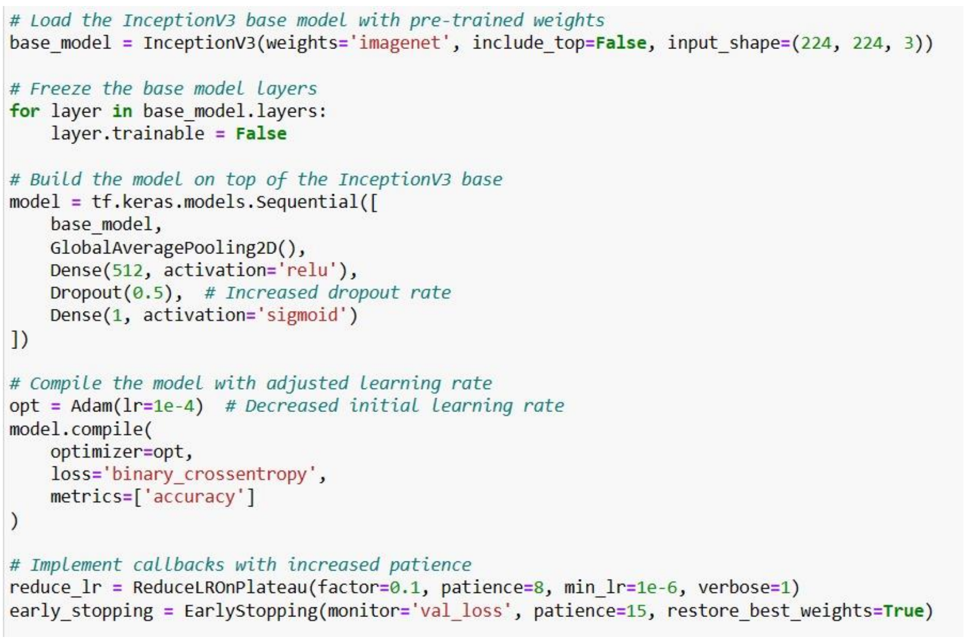
Figure 4.16 Configuration of the DenseNet121 model.

4.7.3 InceptionV3 Model

The InceptionV3 model is a convolutional neural network (CNN) architecture designed by Google, known for its efficiency and accuracy in image classification tasks [27]. During training, data augmentation techniques are applied to increase the diversify of the training dataset. Transfer learning is then employed by initializing the model with pretrained weights from the InceptionV3 base model. This enables the model to benefit from the learned representations of generic image features, allowing for faster convergence.

Furthermore, fine-tuning is performed by freezing the pretrained base layers while training only the top layers of the model on the task- specific dataset. This approach allows the model to specialize its learned representations to the specific characteristics of the weather classification task, without significantly altering the well-established features learned from ImageNet. A ReduceLROnPlateau callback is employed to dynamically adjust the learning rate based on validation performance to ensure that the model's learning rate adapts to the training progress and avoid potential issues such as plateauing.

Additionally, dropout regularization with a higher dropout rate of 0.5 is employed to reduce overfitting by randomly dropping out units during training, preventing the model from relying too heavily on any individual feature. Finally, early stopping is implemented to halt the training process once the validation loss begins to rise, thereby averting overfitting of the model to the training data. Figure 4.17 displays the configuration of the InceptionV3 model.



```python
# Load the InceptionV3 base model with pre-trained weights
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Build the model on top of the InceptionV3 base
model = tf.keras.models.Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(512, activation='relu'),
    Dropout(0.5),  # Increased dropout rate
    Dense(1, activation='sigmoid')
])

# Compile the model with adjusted learning rate
opt = Adam(lr=1e-4)  # Decreased initial learning rate
model.compile(
    optimizer=opt,
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Implement callbacks with increased patience
reduce_lr = ReduceLROnPlateau(factor=0.1, patience=8, min_lr=1e-6, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)
```

Figure 4.17 Configuration of the InceptionV3 model.

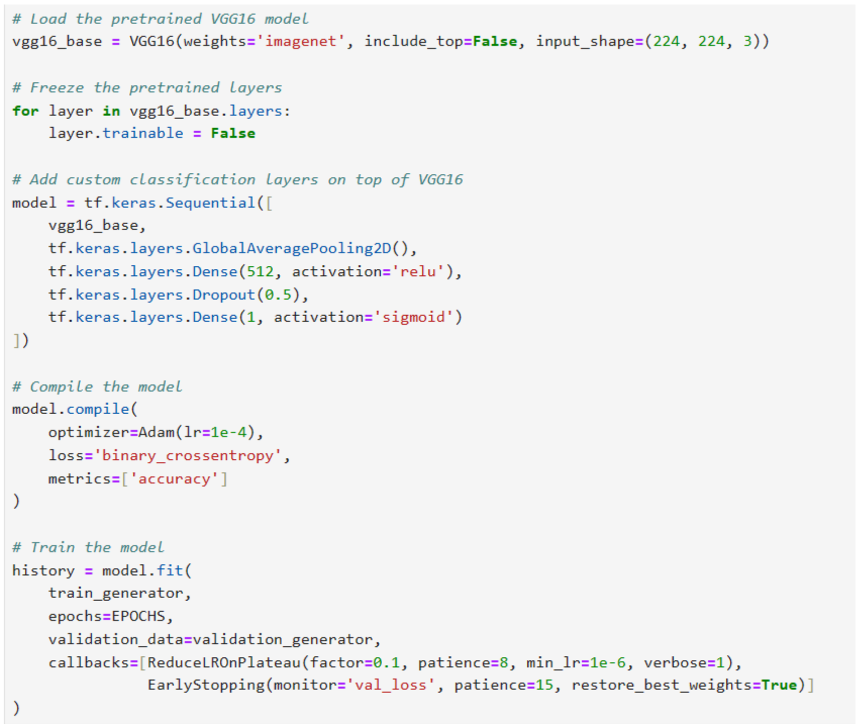
4.8 Train Predictive Model

This section encompasses the implementation of predictive modelling techniques geared towards classifying cloud patterns indicative of impending rainfall. Three distinct convolutional neural network (CNN) models, including VGG16, MobileNet and InceptionV3 models are trained to analyse cloud images and classify them based on their features. The primary objective is to enable the system to accurately predict rainfall events by recognizing specific cloud patterns associated with precipitation. Each CNN is trained on the labelled cloud images, with a focus on capturing the nuanced characteristics of clouds before rain. This section delineates the training methodology employed for each CNN model.

4.8.1 VGG16 Model

The selection of VGG16 model for this task is informed by its superior performance in previous classification tasks, therefore it has the potential to effectively predict rainfall events based on cloud patterns. Initially, the training dataset has undergone data augmentation to increase the diversify of the dataset. The model architecture involves loading the VGG16 model with pretrained weights from the ImageNet dataset and freezing the pretrained layers to retain learned features and prevent overfitting. Custom classification layers are then added on top of the VGG16 base as shown in Figure 4.18.

Following model architecture setup, compilation and training take place, where the model is equipped with the Adam optimizer. Callbacks for learning rate reduction and early stopping are utilized to enhance model performance and prevent overfitting during training. Subsequently, the trained model is evaluated on the test dataset to evaluate its performance, with test loss and accuracy calculated and displayed. A classification report is generated to evaluate model performance. Figure 4.18 shows the configuration of VGG16 model.



```
# Load the pretrained VGG16 model
vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the pretrained layers
for layer in vgg16_base.layers:
    layer.trainable = False

# Add custom classification layers on top of VGG16
model = tf.keras.Sequential([
    vgg16_base,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(
    optimizer=Adam(lr=1e-4),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Train the model
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    callbacks=[ReduceLROnPlateau(factor=0.1, patience=8, min_lr=1e-6, verbose=1),
               EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)]
)
```

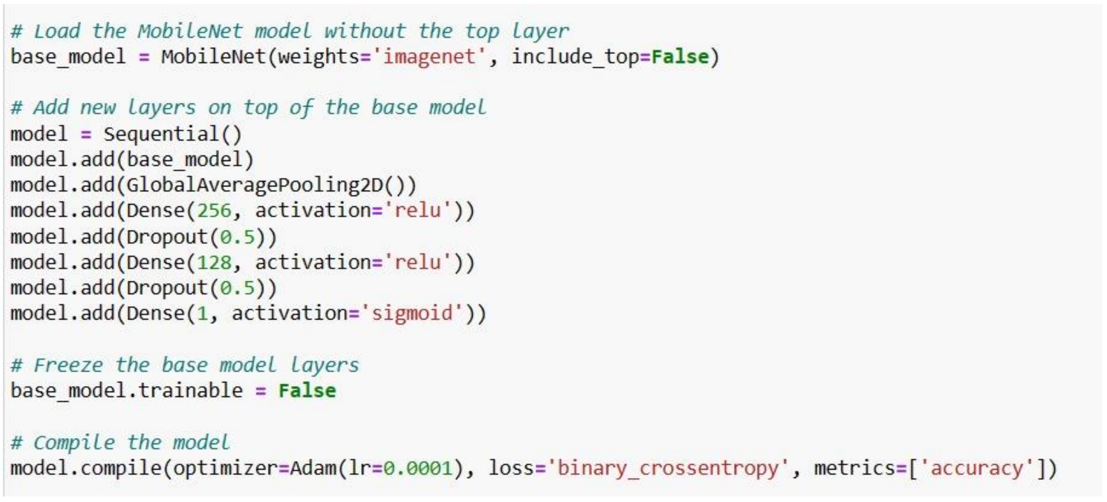
Figure 4.18 Configuration of VGG16 model.

4.8.2 MobileNet Model

Additionally, the MobileNet model has been chosen for this predictive modeling task due to its efficiency and suitability for small datasets [28]. Due to time constraints, the dataset collected for this project is not as extensive as desired. Consequently, this model is chosen as it can help to maximize the predictive performance despite the limitations posed by the dataset size.

The implementation begins by loading the MobileNet model without its top layer, excluding the classification head. Custom layers are then added atop the base MobileNet architecture to tailor it to cloud classification task. Following the addition of these trainable layers, the base MobileNet layers are frozen to preserve the pre-learned features. The model is equipped with an Adam optimizer and binary cross-entropy loss function. Data augmentation techniques,

including rotation, shifting, shearing and zooming are employed to augment the training set, enhancing model generalization and robustness.
z the model's performance is evaluated on the test set to assess its generalization ability. Classification metrics are computed and summarized in a classification report. Figure 4.19 shows the configuration of the MobileNet model.

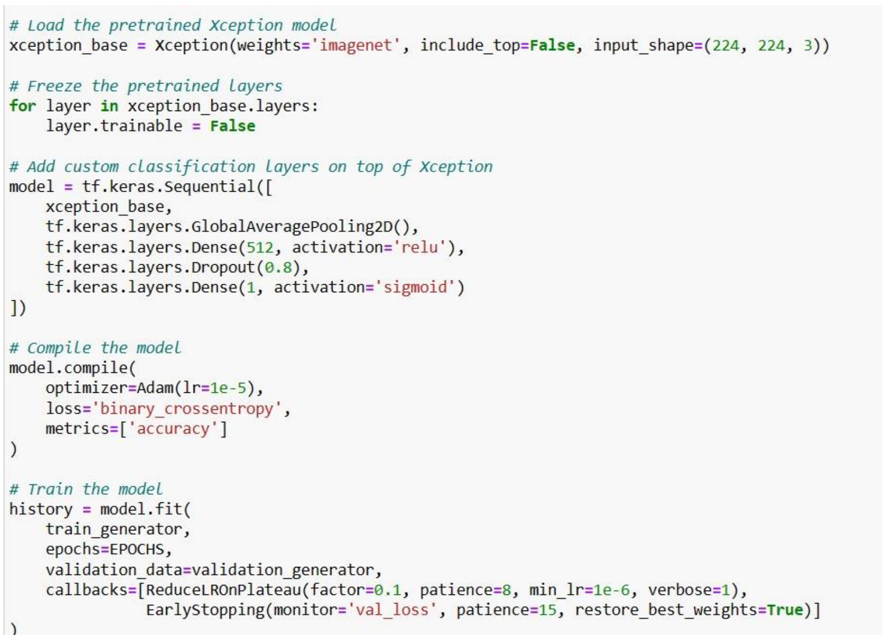


```
# Load the MobileNet model without the top layer
base_model = MobileNet(weights='imagenet', include_top=False)

# Add new layers on top of the base model
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Freeze the base model layers
base_model.trainable = False

# Compile the model
model.compile(optimizer=Adam(lr=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 4.19 Configuration of the MobileNet model.

4.8.3 Xception Model

Besides VGG16 and MobileNet, the Xception model has also been used to train on the dataset. It is a deep convolutional neural network architecture that extends the concept of inceptions modules, designed to improve computational efficiency and performance in image classification tasks [29]. The training dataset undergoes data augmentation techniques, and the augmented data is fed into the model through data generators. Then, the pretrained Xception model is loaded with the fully connected layers excluded, and its layers are frozen to retain the pretrained weights. Custom classification layers are added atop the Xception base, and the model is then compiled using an Adam optimizer and accuracy metric. Additionally, callbacks such as ReduceLROnPlateau and early stopping are employed to adjust the learning rate and prevent overfitting. Following the training phase, the model undergoes an evaluation process using the test set. Classification metrics are then computed and summarized in a classification report. Figure 4.20 shows the configuration of the Xception model.



```
# Load the pretrained Xception model
xception_base = Xception(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the pretrained layers
for layer in xception_base.layers:
    layer.trainable = False

# Add custom classification layers on top of Xception
model = tf.keras.Sequential([
    xception_base,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.8),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(
    optimizer=Adam(lr=1e-5),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Train the model
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    callbacks=[ReduceLROnPlateau(factor=0.1, patience=8, min_lr=1e-6, verbose=1),
               EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)]
)
```

Figure 4.20 Configuration of Xception model.

4.9 Implementation Issues and Challenges

The first implementation challenge lies in establishing an optimal camera module setup for rainfall detection. Selecting an ideal location have faced few challenges such as determining suitable location, addressing potential obstructions, and fine-tuning viewing angles for comprehensive coverage. Besides that, exposure and backlighting issues might occur due to variations in the intensity and direction of the sunlight, leading to overexposed or underexposed images. This might cause the subject to appear dark in the captured image. These challenges are particularly pronounced in outdoor environments, where changing weather conditions and the dynamic position of the sun impact the overall lighting conditions. To solve this issue, careful adjustment of camera settings and implementation of shading techniques such as using a reflector to reflect light to the camera and strategic placement of cardboard to shield the wire extension and Raspberry Pi from direct sunlight are needed. This measure helped mitigate overheating risks and minimized shadow reflections, ensuring consistent and reliable image capture.

The second challenge lies in the accurate identification and distinction of raindrops using rule-based labelling method. This task is fraught with complexities stemming from diverse factors including inconsistent image resolutions, varying light conditions and the nuanced nature of raindrop patterns. Therefore, the adoption of machine learning techniques become essential. Moreover, it is essential to establish a parameter and continually fine-tune its value to align with the dynamic visual characteristics of raindrops under varying meteorological conditions.

The third challenge encountered during the implementation phase was the training of the predictive model, which was hindered by the limited dataset collected within the allotted timeframe. Insufficient data posed a significant obstacle to model training as machine learning algorithms require substantial and diverse datasets to effectively learn and generalize patterns. Due to time constraints, the dataset contained fewer samples than desired, particularly for the ‘rainy’ class. This scarcity of data compromised the model’s ability to accurately identify and classify the two weather patterns, leading to suboptimal performance and reduced predictive accuracy. To mitigate this challenge, efforts were made to augment the dataset through techniques such as data synthesis and transfer learning. However, despite these efforts, the limited dataset remained a significant constraint in achieving the desired level of model performance.

4.10 Concluding Remark

In conclusion, the implementation of the proposed system encountered several challenges and issues that has been addressed to ensure its effectiveness and reliability. From establishing an optimal camera module setup for rainfall detection to building a predictive model to predict rainfall event, each step presented its own set of complexities. However, these challenges were effectively addressed. Despite the limited dataset, efforts were made to augment data and optimize model, showcasing resilience and adaptability in the face of constraints. The successful implementation of the system demonstrates its potential to revolutionize cloud data collection and micrometeorology forecasting, paving the way for more convenient weather predictions and informed decision-making in various applications.

Chapter 5

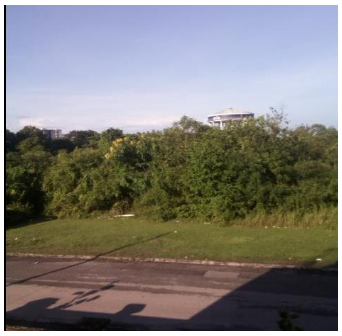
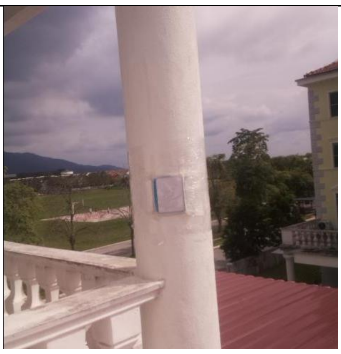
System Evaluation And Discussion

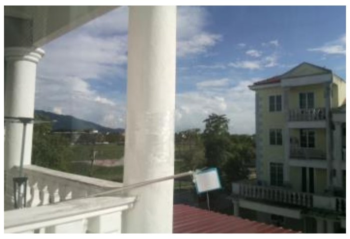
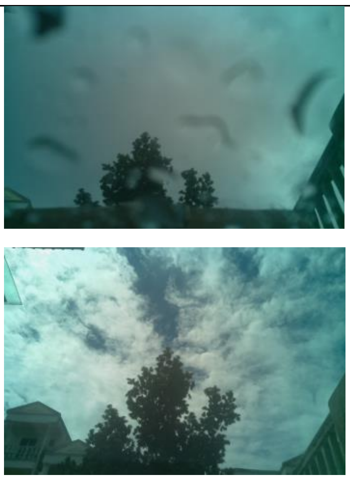
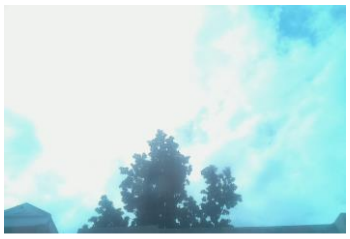
5.1 System Performance Evaluation on Camera and Data Collection

5.1.1 Camera Module Setup Evaluation

This section assesses the camera module setup's performance across different locations to assess its effectiveness in capturing cloud images and detecting rainfall events. Each location presents unique challenges and considerations such as lighting conditions, obstruction of view and environmental factors affecting image quality, including tools and techniques employed. Additionally, this section highlights key findings and observations based on the captured image to optimize the camera module setup for improved performance and reliability across different locations. Table 5.1 shows the different setup of the camera module.

Table 5.1 Different setup of the camera module.

No.	Cloud Image	Method / Tools Used	Discussion
1		-	- The location is not suitable to capture cloud images due to the challenges posed by low resolution of the camera module which hinders the effective detection of raindrops.
2		- A magic water paper that will change colour after contact with water and returns to white colour once it is dry.	- The location of the magic water paper is suboptimal due to a roof above the pole, preventing raindrops from reaching the paper.

3		- A magic water paper with an attached stick extends beyond the roof, allowing it to contact with raindrops.	- The location and method employed are not suitable to capture and detect rainfall as the camera's resolution is insufficient to discern small changes in the colour of the magic water paper.
4		- A magic water paper placed on a glass surface above the camera module to monitor changes in the paper's colour.	- The issue of overexposure to sunlight occurs when placing the magic water paper on one side of the camera module, resulting in suboptimal capture of cloud images.
5		- A glass is placed on top of a box that housed the camera module. The camera module is adjusted to a 45-degree angle.	- Raindrop can be captured clearly but there are some obstacles hindering the complete capture of cloud images.
6		- A multi-camera adapter is used to connect two camera modules to capture images from different angles. One camera captures images focused solely on clouds, while the other includes a tree in the frame to facilitate the labelling process.	- There are compatibility issues between the camera modules, multi-camera adapter and Raspberry Pi which affect image quality.

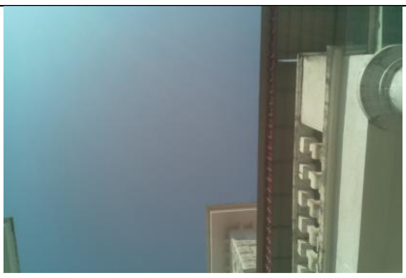
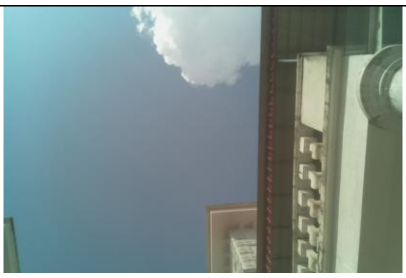
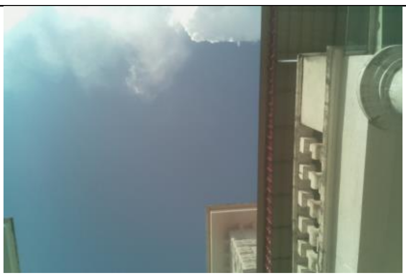
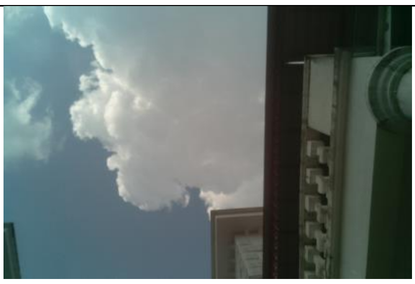
		- This approach is necessary because the labelling script relies on detecting raindrops based on the blurriness of objects.	
7		- A camera module is placing beneath a glass to capture the cloud images.	- Optimal cloud image is captured, and raindrop can be detected. - The labelling script has been modified to detect raindrops based on this setup.

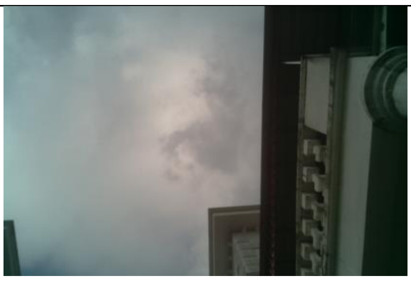
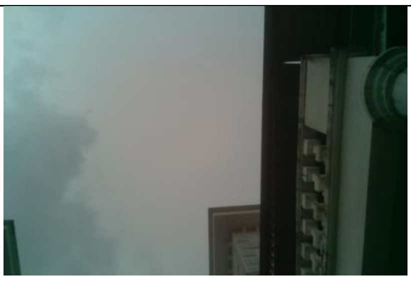
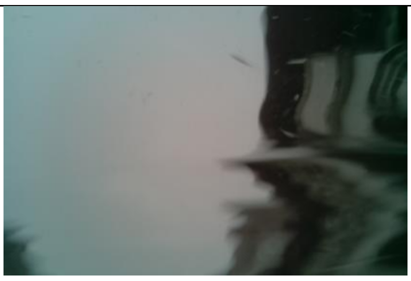
Based on the experiment, the seventh camera setup up method as outlined in the table facilitated optimal image capture. Therefore, this method is employed in the system to ensure consistent and high-quality data acquisition for micrometeorology analysis.

5.1.2 Data Collection Process Evaluation

This section will evaluate the system's capability to capture a diverse range of cloud formations accurately. The camera module is configured to capture images at regular intervals while the system is operational and connected to the internet for data upload. Table 5.2 shows the example of images captured within 30 minutes.

Table 5.2 Example of images captured in 30 minutes.

No.	Time	Cloud Image	Discussion
1	20240221-16:00:30		- Clear sky. - Image quality: Good
2	20240221-16:05:31		- Clear sky with a small cloud. - Image quality: Good
3	20240221-16:10:32		- The sky is partly cloudy, with several clouds visible. - Image quality: Good
4	20240221-16:15:32		- Partly cloudy. - Image quality: Good

5	20240221-16:20:32		- Cloudy. - Image quality: Good
6	20240221-16:25:34		- Overcast. - Image quality: Good
7	20240221-16:30:34		- Rainy. - Image quality: Not very good as the shape of raindrop cannot be captured clearly.

Based on the table above, the system can capture diverse range of cloud formation with good image quality. This will help in the creation of ground truth dataset to train the classification and predictive models. Throughout the entire data collection process, a total of 3950 images were gathered, with the rainy class comprising 160 images.

5.1.3 System Operation Evaluation

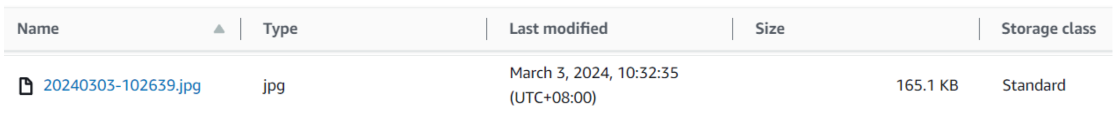
This section aims to evaluate the system's responsiveness in capturing and uploading cloud images to cloud storage platform. To assess this, four test cases have been designed and implemented to measure the performance of the system. Table 5.3 shows the result obtained from these test cases.

Table 5.3 Performance evaluation of cloud image capture and upload system.

Test Case ID	**Test Case**	**Execution Steps**	**Outcome**	**Discussion**
CASE_01	Upload images when connected to internet	1. Disconnect the system from the internet for one hour. 2. Capture a series of images during that period. 3. Reconnect the system to the internet. 4. Monitor the system's behaviour to ensure all cloud images are uploaded to cloud storage platform.	- The system takes 28 seconds to connect to the internet. - The system takes 58 seconds to upload 15 images to cloud storage platform.	- The system is able to store all the images files locally on the system and memory in when it is not connected to the internet. - The system able to upload all the stored images once it is connected to the internet.

CASE_02	System reliability to capture and upload images consistently over an extended period.	1. Initiate the system and camera module setup. 2. Monitor the system's operation continuously from 7AM to 7PM.	- The system capture and upload 144 cloud images.	- The system demonstrates consistent performance over the 12-hour monitoring period.
CASE_03	Data integrity check to ensure accurate and reliable storage.	1. Access the cloud storage platform and retrieve a sample of uploaded images. 2. Verify the completeness and accuracy of the images by comparing them against the original captured images. 3. Check for any missing files or corruption in the uploaded data.	- All the captured images have been uploaded completely. - No missing files or corruption in the uploaded data.	- The uploaded data maintains its integrity during the transfer process.

In addition, a test case was conducted to assess the system's ability to upload image after a 4-minute interval following image capture. As depicted in Figure 5.1, the image was captured at 10:26:39 and subsequently uploaded at 10:32:35, indicating a 5-minute delay. This test underscores the system's capability to upload pending images upon establishing an internet connection.



Name	Type	Last modified	Size	Storage class
20240303-102639.jpg	jpg	March 3, 2024, 10:32:35 (UTC+08:00)	165.1 KB	Standard

Figure 5.1 Screenshot of AWS S3 bucket which depicting delayed image upload

5.2 Model Performance Evaluation and Discussion

To evaluate the labelling, classification and prediction processes, several key metrics were utilized to assess the performance of the model comprehensively. The key metrics used are accuracy, precision, recall and F1 score. The accuracy asses the overall correctness, while precision specifically evaluates the accuracy of positive predictions. Besides that, the recall metric examines the model's capability to identify all positive instances and F1 score provides a balanced measure of precision and recall. These metrics collectively offer insights into the method or the model's effectiveness in accurately labelling and predicting the cloud images. Figure 5.2 shows the formula of the key metrics.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

TP = True positive

TN = True negative

FP = False positive

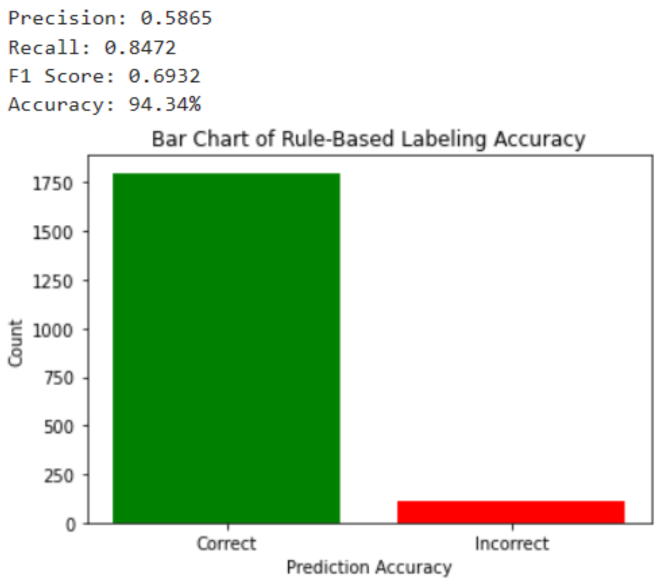
FN = False negative

Figure 5.2 Formula of the key metrics.

5.2.1 Labelling Method Performance Evaluation and Discussion

The proposed method is chosen to label image in this system is because users can modify the threshold to fit different dataset or image with different background. The performance metrics indicate that while the method achieves a commendable accuracy of 94.34%. there are areas for improvement. Specifically, the precision metric, which measures the proportion of correctly labelled rainy images out of all images labelled as rainy, stands a 0.5865. This suggests that approximately 58.65% of images labelled as rainy are indeed rainy. Furthermore, the recall metric, which assesses the method's ability to correctly identify rainy images out of all actual rainy images, is relatively high at 0.8473, indicating that around 84.72% of actual images are correctly identified. The F1 score, offering a balanced measure of precision and recall, is calculated at 0.6932. While the method demonstrates strong performance in capturing rainy instances and achieving high overall accuracy, there is potential for enhancement in precision to reduce false positives.

The occurrence of false positives, particularly in images containing exposed sunlight, poses a challenge to the labelling method's accuracy. These false positives arise when images affected by intense sunlight are incorrectly labelled as rainy. This issue can lead to misinterpretations and inaccuracies in image classification. Figure 5.3 shows the bar chart illustrating the accuracy of rule-based labelling, while Figure 5.4 shows the confusion matrix obtained.



```
Precision: 0.5865
Recall: 0.8472
F1 Score: 0.6932
Accuracy: 94.34%
```

Figure 5.3 Bar chart illustrating the accuracy of rule-based labelling.

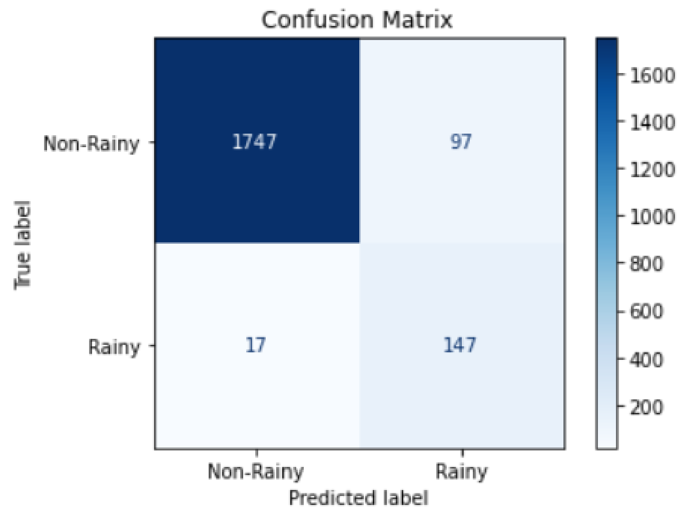


Figure 5.4 Confusion matrix for rule-based labelling.

5.2.2 Error Analysis for Rule-based Labelling Method

Sunlight-exposed cloud image

In the rule-based labelling process, one notable challenge is the issue of exposure to sunlight, where the Laplacian value may appear similar to rainy images due to the blurriness caused by overexposure. This similarity in Laplacian values between sunlight-exposed images and rainy images will lead to misclassification, resulting in false positives. Therefore, this type of non-rainy images is incorrectly labelled as rainy. The examples of sunlight-exposed cloud image are shown in Figure 5.5 and Figure 5.6.

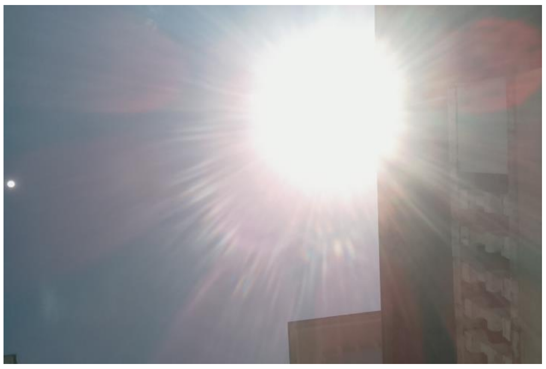


Figure 5.5 Sunlight-exposed cloud image 1.

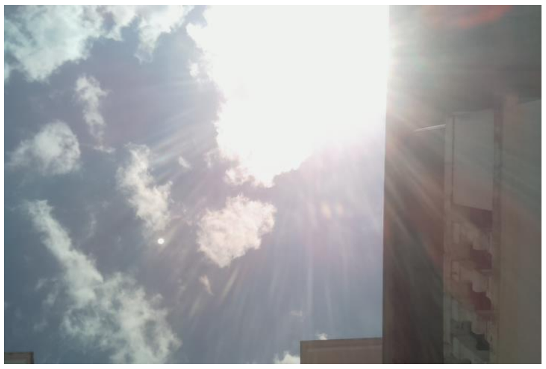


Figure 5.6 Sunlight-exposed cloud image 2.

Overcast or Heavily Clouded Images

In instances where the sky is heavily obscured by cloud cover or when images appear excessively dark due to limited light, the system encounters difficulty in accurately discerning between rainy and non-rainy weather conditions. When images exhibit extensive cloud cover or appear excessively dark, details within the image become obscured, resulting in an overall increase in blurriness. Therefore, it will cause the Laplacian value to fall below the threshold, leading to false positives where images portraying cloudy or dark conditions are inaccurately labelled as representing rainfall. This issue highlights the system’s difficulty in effectively discerning between blurriness associated with cloud cover or darkness and blurriness indicative of actual rainy weather conditions. Figure 5.7 and Figure 5.8 display the examples of heavily clouded image.

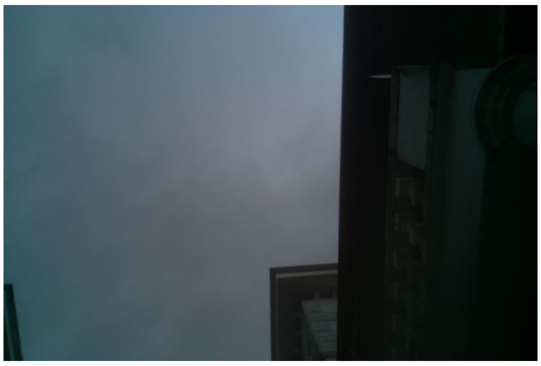


Figure 5.7 Heavily clouded image 1.

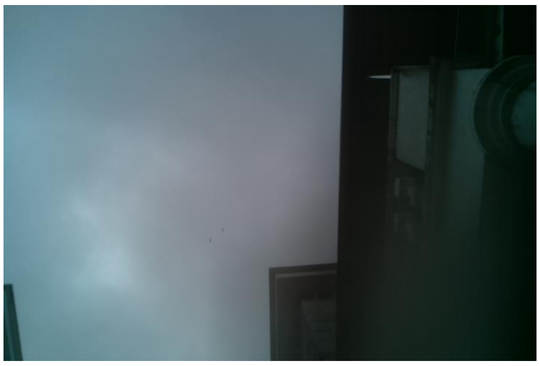


Figure 5.8 Heavily clouded image 2.

Cloud image depicting light rain conditions

In scenarios where images feature minimal raindrop coverage, the challenge arises from the limited extent of blurriness within the image. As the blurriness caused by light rainfall is typically subtle and confined to small areas, the system may struggle to detect these nuances effectively. Consequently, images depicting light rainfall may fail to meet the system's blurriness threshold criteria for classification as rainy weather. This results in false negatives, where images portraying light rain conditions are incorrectly labelled as non-rainy. The issue stems from the fact that the Laplacian value is higher than the predefined threshold, leading to misclassification despite the presence of rainfall. Figure 5.9 and Figure 5.10 shows the examples of light rain cloud image.

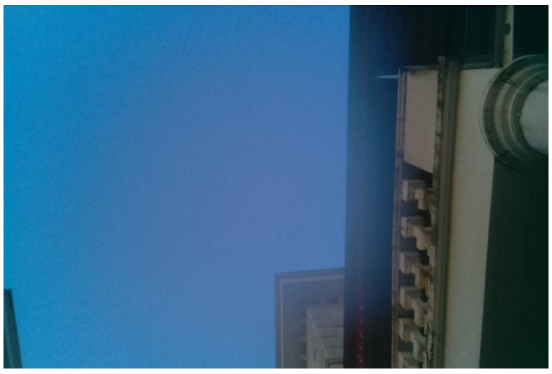


Figure 5.9 Light rain cloud image 1.

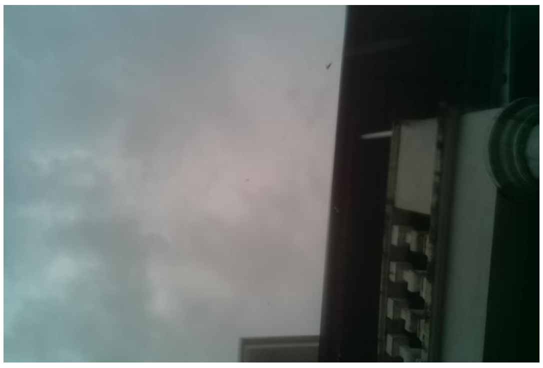


Figure 5.10 Light rain cloud image 2.

5.3 Classification Models Performance Evaluation and Discussion

This section compares the performance of the three trained classification models to select the best-performing model for the system. The dataset used for training comprises 403 images from 'non_rainy' class and 120 images from 'rainy' class. To mitigate the issue of class imbalance, the under-sampling technique is used where the number of data instances in the 'non_rainy' class was reduced to 403, thereby balancing the representation of classes. Additionally, the validation dataset comprises 124 images, while the test set consists of 105 images.

Each model's results are evaluated based on the classification key metrics. The section provides a detailed comparison of each model and highlight the factors contributing to their performance. Ultimately, the best-performing model is identified and chosen for integration into the system.

5.3.1 VGG16 Model Evaluation

The VGG16 model was trained and evaluated on the dataset for 30 epochs. Based on the accuracy and loss graphs for the VGG16 model evaluation, we observe a positive trend in both training and validation accuracy throughout the epochs.

At the outset, the model started with a training accuracy of approximately 75.89% and a validation accuracy of 80.65%. Over subsequent epochs, there was a noticeable increase in both training and validation accuracy, with the model achieving a peak training accuracy of 95.94% and a validation accuracy of 95.97%. This upward trend suggests that the model is effectively extracting patterns and information from the training data and is demonstrating good generalization performance on unseen data.

In terms of loss metrics, both training and validation loss exhibit a consistent decrease over the epochs. The training loss decreases from 0.5549 to approximately 0.1124, while the validation loss decreases from 0.3759 to around 0.1014. This downward trend indicates that the model improves its classification performance and converges towards optimal parameters.

However, fluctuations and irregularities in the accuracy and loss curves are observed during the training process. These fluctuations occurred is because the small and imbalanced dataset used. The limited size of the dataset may lead to increased variability in the model's performance metrics, resulting in the zigzag pattern observed in the graphs [30]. Additionally, class imbalances are one of the factors that contribute to the fluctuations in accuracy. If certain classes are underrepresented or overrepresented in the dataset, the model may struggle to accurately classify instances from these classes, leading to fluctuations in accuracy as the model adjusts its predictions during training [31].

Despite these fluctuations, the overall trend in the performance metrics should still indicate improvement over time, as seen in the gradual decrease in loss and increase in accuracy across epochs. Figure 5.11 shows the accuracy and loss graph for VGG16 model, while Figure 5.12 shows the confusion matrix obtained. The confusion matrix will be used to compare with other models in Chapter 5.3.4.

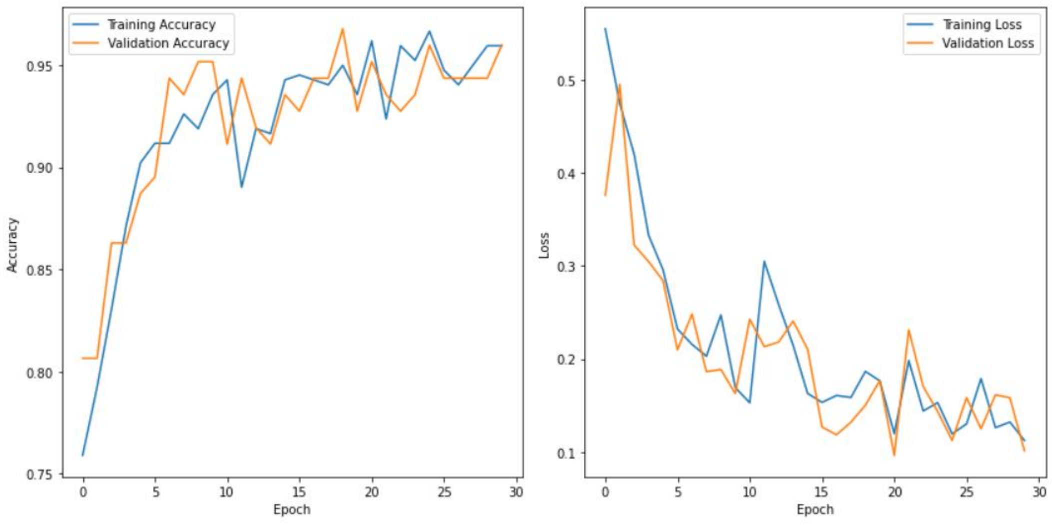
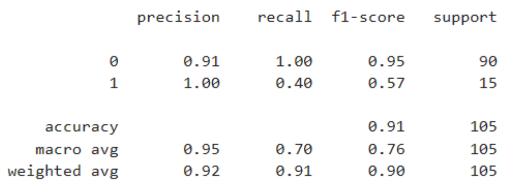


Figure 5.11 Accuracy and loss graph for VGG16 Model.



```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        90
           1       1.00      0.40      0.57        15

    accuracy                           0.91       105
   macro avg       0.95      0.70      0.76       105
weighted avg       0.92      0.91      0.90       105
```

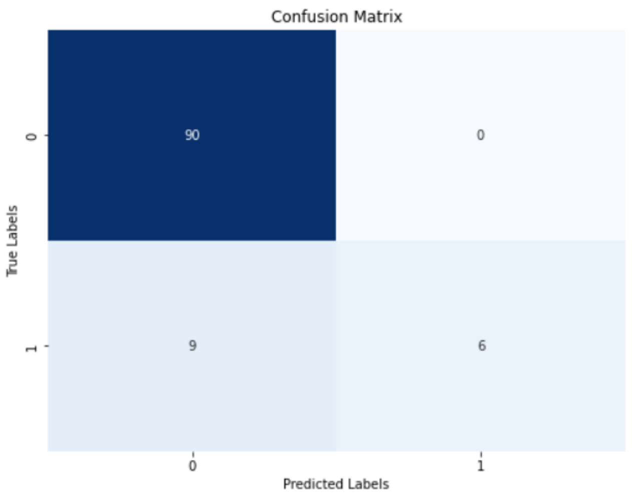


Figure 5.12 Confusion matrix for VGG16 Model.

5.3.2 DenseNet121 Model Evaluation

The training was conducted over 50 epochs and was halted at epoch 20 due to the early stopping mechanism to prevent overfitting. It has a similar trend to the VGG16 model in terms of accuracy and loss metrics. The DenseNet121 model initially began with a training accuracy of approximately 78.20% and a validation accuracy of 86.29%, both metrics showed a positive trend, reaching peak values of 92.16% and 89.52% respectively. Training and validation losses consistently decreased from 0.4988 to 0.2040 and from 0.3178 to 0.2400, indicating improved classification performance.

However, there is a slight fluctuation in both training and validation accuracy and loss, suggesting potential overfitting or convergence issues. This phenomenon suggests potential challenges with model convergence or optimization, leading to difficulties in further improving performance metrics such as accuracy and loss. The observed limitations underscore the importance of addressing model robustness, particularly in scenarios characterized by data variability or domain shifts. Therefore, further data collection process may be necessary to enhance the model's robustness and generalization capacity in real-world applications. In overall, the average training accuracy is 89.74% and average validation accuracy is 88.81%. Figure 5.13 shows the accuracy and loss graph for DenseNet121 model, while Figure 5.14 displays the confusion matrix obtained.

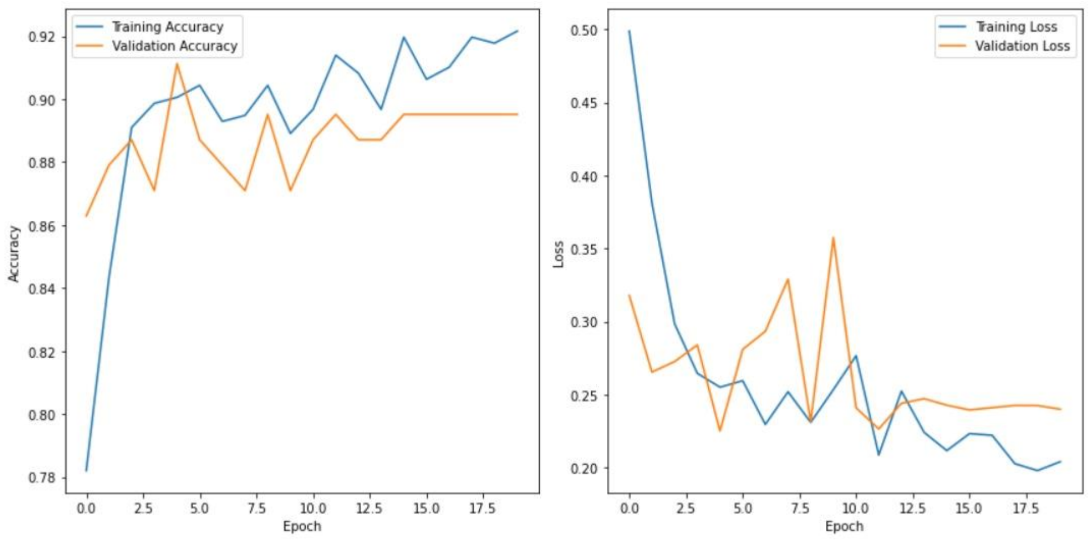


Figure 5.13 Accuracy and loss graph for DenseNet121 Model.

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        90
           1       1.00      0.20      0.33        15

    accuracy                           0.89       105
   macro avg       0.94      0.60      0.64       105
weighted avg       0.90      0.89      0.85       105
```

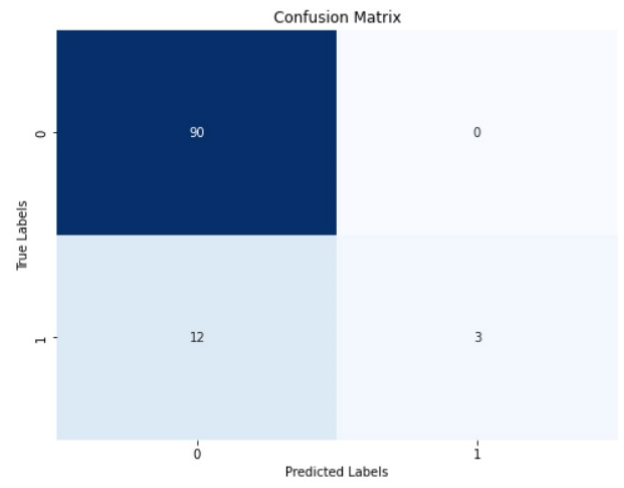


Figure 5.14 Confusion matrix for DenseNet121 Model.

5.3.3 InceptionV3 Model Evaluation

The InceptionV3 model was trained over 30 epochs, and the trend is similar to those seen in VGG16 and DenseNet121 models regarding accuracy and loss metrics emerge. The model began training with a training accuracy of approximately 71.13% and a validation accuracy of 81.45% and reaching peak training and validation accuracies of 87.57% and 92.74% respectively. Regarding loss metrics, both training and validation losses demonstrated a consistent decrease throughout the epochs. The training loss decreased from 0.8724 to approximately 0.2593, while the validation loss decreased from 0.3876 to around 0.1801. This decline suggests that the model continually improved its classification performance and approached optimal parameter setting over the training period. However, similar to the other models, fluctuations and irregularities in accuracy and loss curves are observed during training due to factors such as dataset' small size and class imbalances. This has led to increased variability in the model's performance metrics and cause the model to struggle in accurately classifying instances from certain classes. In overall, the average training and validation accuracies for the InceptionV3 model are 85.78% Figure 5.15 shows the accuracy and loss graph for the InceptionV3 model, while Figure 5.16 shows the confusion matrix.

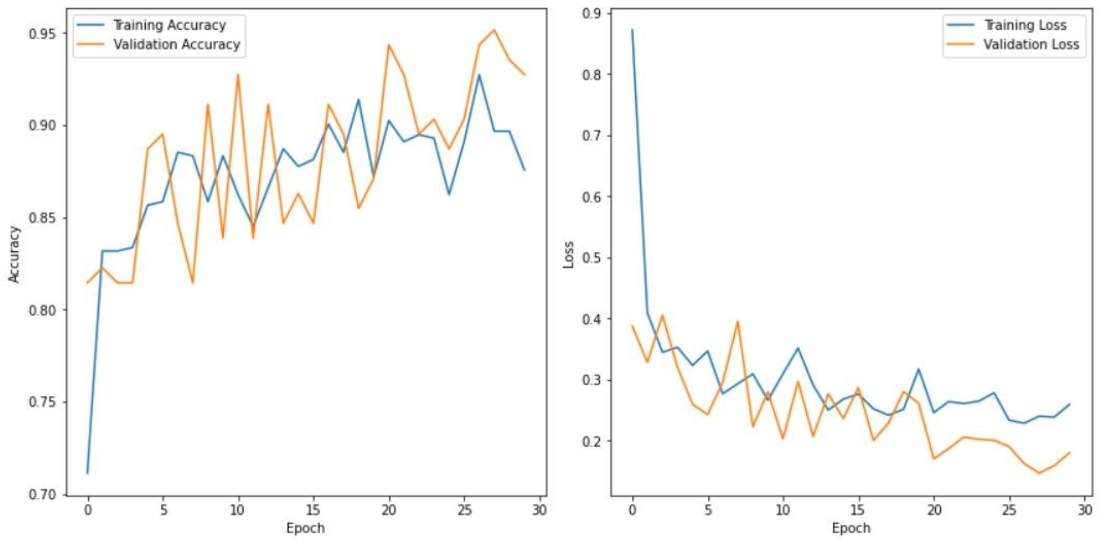


Figure 5.15 Accuracy and loss graph for InceptionV3 Model.

```
           0       0.87      1.00      0.93        90
           1       1.00      0.13      0.24        15

    accuracy                           0.88       105
   macro avg       0.94      0.57      0.58       105
weighted avg       0.89      0.88      0.83       105
```

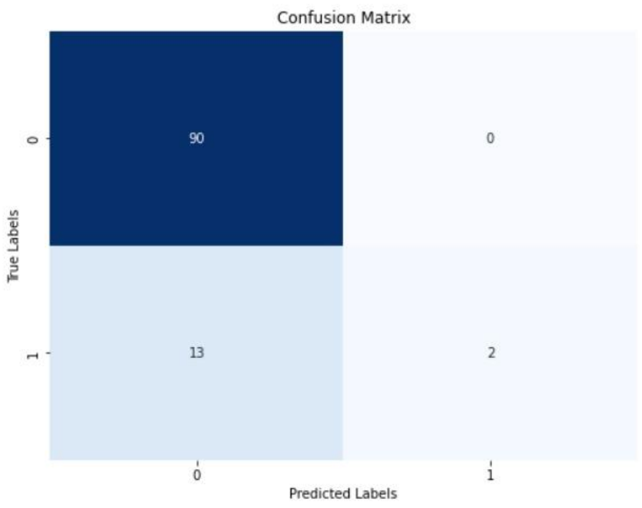


Figure 5.16 Confusion matrix for InceptionV3 Model.

5.3.4 Classification Models Comparison

This section will assess and contrast the performance of each model. Subsequently, the most effective model will be selected for deployment within the system. The tables below display the key metrics of each model on the testing dataset.

Table 5.4 Performance metrics for Class 0 (non-rainy) across three models.

Class 0 (non_rainy)			
Models	Precision	Recall	F1-score
VGG16	0.91	1.00	0.95
InceptionV3	0.88	1.00	0.94
DenseNet121	0.87	1.00	0.93

Table 5.5 Performance metrics for Class 1 (rainy) across three models.

Class 1 (rainy)			
Models	Precision	Recall	F1-score
VGG16	1.00	0.40	0.57
InceptionV3	1.00	0.20	0.33
DenseNet121	1.00	0.13	0.24

Table 5.6 Test accuracy comparison for the three models.

Models	Testing Accuracy
VGG16	0.91
InceptionV3	0.89
DenseNet121	0.88

Comparing the precision, recall and F1-score for Class 0 (non_rainy), all models perform similarly well, with VGG16 exhibiting slightly higher scores across the board. Based on Table 5.3.4.1, it can be observed that the VGG16 model achieves high precision of 91%, indicating it can make more accurate positive predictions in identifying non-rainy instances compared to other models. When it comes to recall, the model achieved a perfect 100%, meaning that it correctly identified all non-rainy instances.

However, for Class 1 (rainy), VGG16 also outperforms the other models significantly. It achieved a perfect 100%, indicating it made accurate positive predictions, particularly in identifying rainy instances. However, its recall for Class 1 is only 40%, suggesting that it missed some rainy instances. In comparison, InceptionV3 and DenseNet121 achieved lower precision, recall and F1-score across the two classes, suggesting less accurate predictions overall.

Despite this, all models achieved relatively high testing accuracies, with VGG16 leading at 91%, followed closely by InceptionV3 at 89% and DenseNet121 at 88%. These results suggest that VGG16 demonstrates superior performance in accurately classifying both rainy and non-rainy instances, making it the most suitable model for deployment in the system.

5.4 Predictive Modelling Evaluation

This section assesses and contrasts the performance of the predictive models in identifying patterns indicative of rainfall events within the dataset, which comprises two distinct classes, ‘cloudy’ and ‘non-cloudy’. The ‘cloudy’ class encompasses images captured before and during rainfall, providing insights into cloud patterns associated with impending rain. These patterns are crucial for predicting rainfall events accurately. The predictive models are trained to recognize and classify these patterns, enabling them to distinguish between cloudy instances signalling potential rainfall and non-cloudy instances. The dataset used consists of 995 training images, 280 validation images and 105 testing images. After evaluating these models, the best-performing model will be selected and deployed in the system.

5.4.1 VGG16 Model Evaluation

The training results of the VGG16 model exhibit some fluctuations throughout the epochs. In the initial epochs, the model starts with a moderate accuracy of around 59% on the training data and 67% on the validation set. Despite minimal improvement in the accuracy in the subsequent epochs, which hovering around 67%, the model’s performance gradually escalates in later epochs. Towards the end of the training period, the model achieves significant progress, with the training accuracy reaching approximately 81% and the validation accuracy around 78%. The declining pattern observed in the loss graph during training suggests that the model is still be able to minimize its loss function and converging towards an optimal solution.

However, there are some fluctuations occurred in the accuracy and loss graph. These fluctuations could be attributed to the complexity of the features present in the data [32]. In the case of predicting rainfall events based on the cloud images, the features can vary significantly, ranging from subtle variations in cloud formations to more pronounced indicators of imminent rainfall. As the model learns to distinguish between these features, it may encounter challenges in effectively capturing and generalizing them, leading to fluctuations in performance. Figure 5.17 shows the accuracy and loss graph for VGG16 model, while Figure 5.18 shows the confusion matrix obtained.

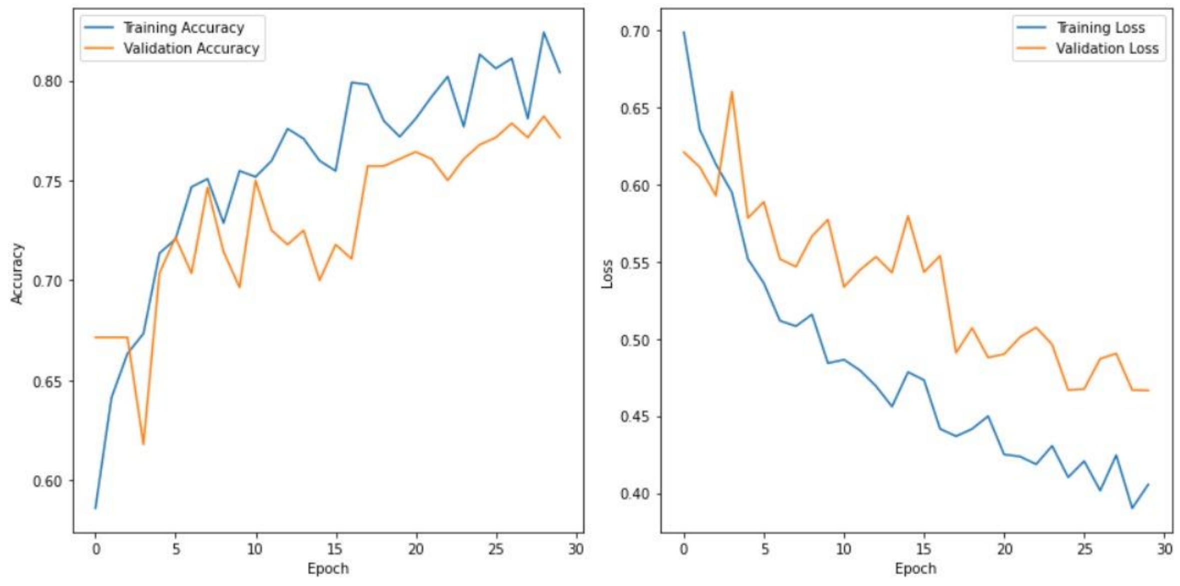


Figure 5.17 Accuracy and loss graph for VGG16 Model.

```
              precision    recall  f1-score   support

           0       1.00      0.13      0.24        30
           1       0.74      1.00      0.85        75

    accuracy                           0.75       105
   macro avg       0.87      0.57      0.54       105
weighted avg       0.82      0.75      0.68       105
```

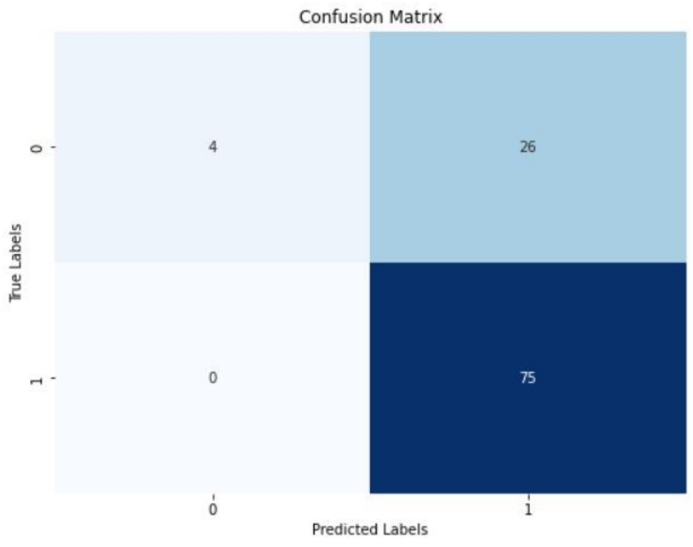


Figure 5.18 Confusion matrix for VGG16 Model.

5.4.2 MobileNet Model Evaluation

The evaluation of the MobileNet model over 30 epochs demonstrates a progressive improvement in both training and validation accuracies. Initially, the model achieved a training accuracy of approximately 60% and a validation of around 63% in the first epoch. Subsequently, as the training progressed, the accuracies reached approximately 78% for training and 77% for validation by the final epoch. This upward trend in accuracies indicates that the model has learned and generalized patterns from the training data.

However, it's worth noting that the model's performance exhibits some fluctuations throughout the training process. For instance, there are instances where the validation accuracy slightly decreases before rising again in subsequent epochs. Despite these fluctuations, the overall trend demonstrates a consistent improvement in performance over time. Figure 5.19 shows the accuracy and loss graph for the MobileNet model, while Figure 5.20 shows the confusion matrix obtained.

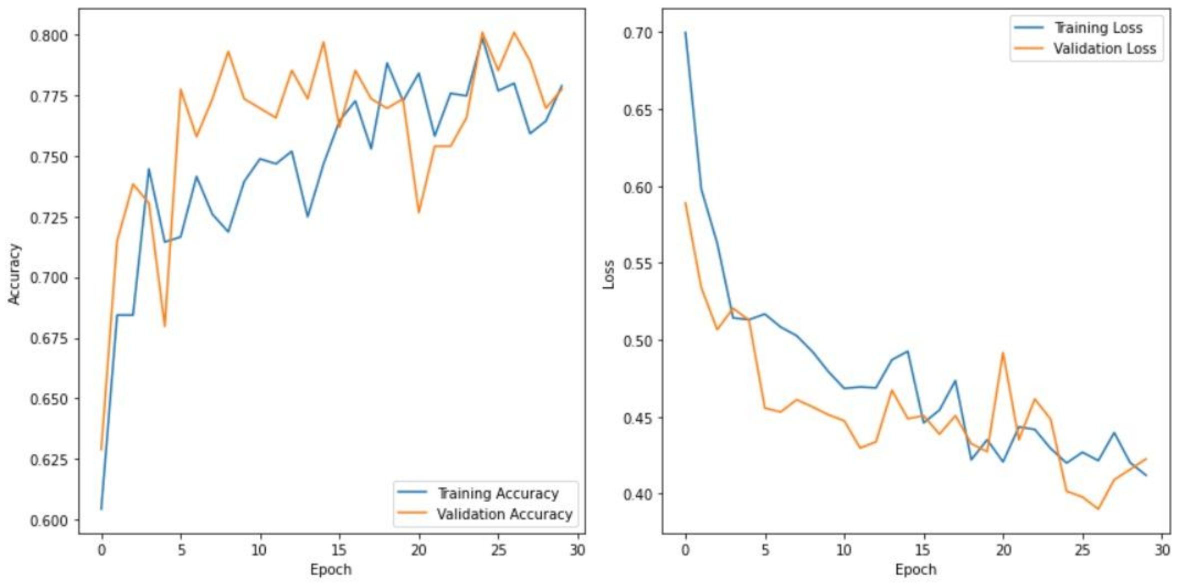


Figure 5.19 Accuracy and loss graph for MobileNet Model.

```
              precision    recall  f1-score   support

           0       1.00      0.30      0.46        30
           1       0.78      1.00      0.88        75

    accuracy                           0.80       105
   macro avg       0.89      0.65      0.67       105
weighted avg       0.84      0.80      0.76       105
```

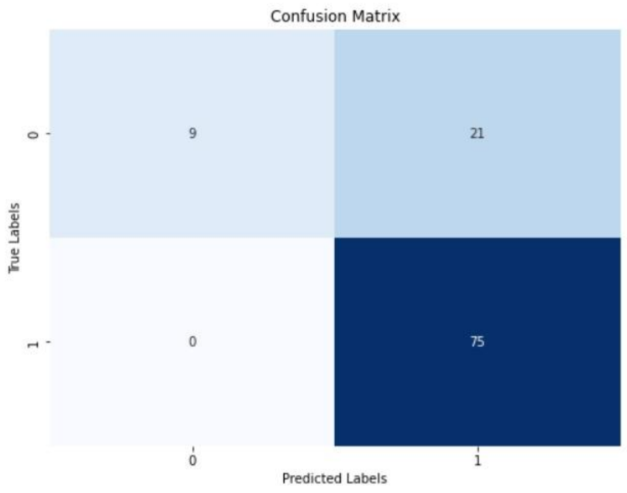


Figure 5.20 Confusion matrix for MobileNet Model.

5.4.3 Xception Model Evaluation

During the initial epochs, the Xception model achieved a reasonable performance during training, with the training accuracy gradually increasing from approximately 64.82% in the first epoch to around 85.93% in the final epoch. Similarly, the validation accuracy also showed an increasing trend, reaching 80.71% by the end of training.

Regarding the loss, both the training and validation losses showed a decreasing trend over the epoch. This suggests that the model was able to effectively minimize its error on both the training and validation sets.

However, the overfitting issue had occurred during the training process, where the model may have learned to memorize the training data rather than generalize well to unseen data. Although the training accuracy steadily increased throughout the epochs, the validation accuracy plateaued around 80.71%, indicating potential overfitting [33]. This overfitting issue may be due to the factors such as limited data and class imbalance issue. Figure 5.21 displays the accuracy and loss graph for the Xception model, while Figure 5.22 shows the confusion matrix obtained.

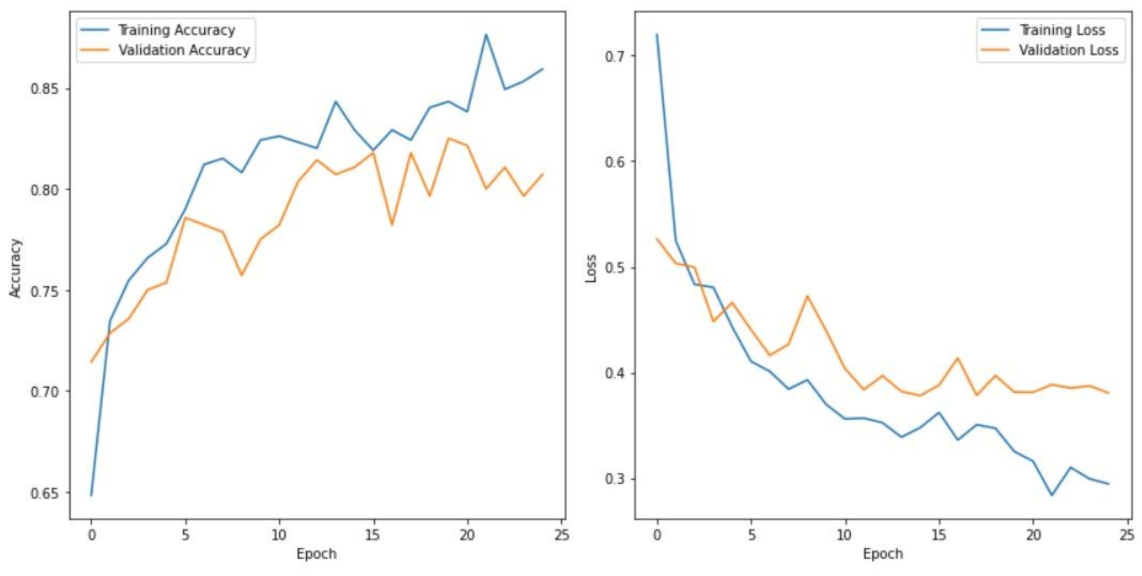


Figure 5.21 Accuracy and loss graph for Xception model.

```
              precision    recall  f1-score   support

           0       0.88      0.23      0.37        30
           1       0.76      0.99      0.86        75

    accuracy                           0.77       105
   macro avg       0.82      0.61      0.61       105
weighted avg       0.79      0.77      0.72       105
```

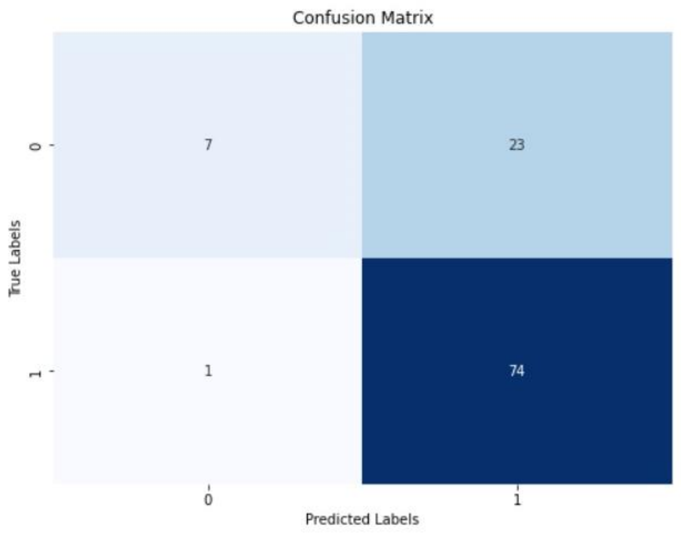


Figure 5.22 Confusion matrix for Xception model.

5.4.4 Predictive Models Comparison

This section will assess and contrast the performance of each model. Subsequently, the most effective model will be selected for deployment within the system. The tables below display the key metrics of each model on the testing dataset.

Table 5.7 Performance metrics for Class 0 (cloudy) across three models.

Class 0 (cloudy)			
Models	Precision	Recall	F1-score
VGG16	1.00	0.13	0.24
MobileNet	1.00	0.30	0.46
Xception	0.88	0.23	0.37

Table 5.8 Performance metrics for Class 1 (non_cloudy) across three models.

Class 1 (non_cloudy)			
Models	Precision	Recall	F1-score
VGG16	0.74	1.00	0.85
MobileNet	0.78	1.00	0.88
Xception	0.76	0.99	0.86

Table 5.9 Test accuracy comparison for the three models.

Models	Testing Accuracy
VGG16	0.75
MobileNet	0.80
Xception	0.77

To evaluate and determine the optimal model among VGG16, MobileNet and Xception, an analysis of key performance metrics was conducted on the testing dataset. MobileNet emerged as the frontrunner as it exhibits the highest F1-score for Class 0 (cloudy), indicative of superior performance in correctly identifying cloudy images while minimizing false positives. Although both VGG16 and MobileNet demonstrated elevated precision for Class 0, MobileNet's has higher recall rate which position it as the most proficient in accurately identifying true cloudy instances.

For Class 1 (non-cloudy), all models showcased commendable precision, recall, and F1-scores, where there are only minimal differences among them. However, MobileNet maintained a slight advantage with the highest testing accuracy, which is 80%, surpassing Xception's 77% and VGG16's 75%.

In the experiment, there are less data collected due to time constraints. However, the model should be designed to use extensive dataset for training. Therefore, the choice of model may shift. Generally, deeper and more complex models like Xception tend to excel when trained on large datasets due to their capacity to capture intricate patterns and features. Therefore, with a sufficient dataset, Xception would likely be the preferred choice. Its deeper architecture allows it to learn a more comprehensive representation of the data.

5.5 Concluding Remark

To wrap up this chapter, the evaluation provided valuable insights into the system's performance across camera setup, data collection, classification and predictive modelling. While each aspect demonstrated promising results, there are notable areas for improvement, particularly in collecting more data to increase the classification and predictive models' robustness. These positive outcomes signify a significant progress in leveraging technology for weather analysis, paving the way for more precise and reliable forecasts.

Chapter 6
Conclusion And Recommendation

6.1 Conclusion

In conclusion, this project explores a new and innovative approach to forecast rainfall events by providing a data engineering solution for micrometeorology forecasting. The essence of this endeavour lies in addressing the inefficiencies of existing methods, which are both costly and time-consuming. The reliance on complex tools and manual interpretation not only incurs significant expenses but also imposes limitations on the spatial resolution required for localized weather prediction. The driving force behind this project is to mitigate these challenges by proposing a data engineering solution for an automated micrometeorology forecasting system. This system is designed to empower users in weather-dependent sectors such as outdoor retail and tourism with actionable insights for decision-making and schedule planning. Besides that, researchers can use this system to collect cloud data, advancing research and analysis in meteorology.

This project introduces innovation by revolutionizing micrometeorology forecasting through automation and advanced data engineering techniques. The novel system utilizes a comprehensive data engineering approach, incorporating data collection, processing and labelling into a seamless workflow. The integration of cloud-based data storage platforms streamlines the data pipeline, enabling real-time access to weather data. Moreover, this project serves as an exploration of novel forecasting methods, utilizing machine learning algorithms for data analysis and predictive modeling. It aims to pioneer new approaches in micrometeorology forecasting by leveraging predictive modeling techniques.

Overall, this project presents a paradigm shift in micrometeorology forecasting, leveraging data engineering principles and advanced analytics to enhance the efficiency and accessibility of weather predictions.

6.2 Future Works

Moving forward, further enhancements and developments can be made to strengthen the system’s capabilities and broaden its applications. One avenue for future work involves collecting more data to further train the machine learning models, thereby improving their accuracy and robustness. Furthermore, a mobile application or user interface can be developed to simplify the system’s usage for end users. Lastly, more techniques can be explored such as using time-series images as data to enable more accurate rainfall predictions.

REFERENCES

[1] "Top 5 Predictive Analytics Models and Algorithms - insightsoftware." Accessed: Nov. 24, 2023. [Online]. Available: https://insightsoftware.com/blog/top-5-predictive-analytics-models-and-algorithms/

[2] J. P. Gerrity, J. R. Gyakum, R. A. Anthes, L. F. Bosart, and E. A. O'Lenic, "Weather forecasting and prediction," 2023, doi: 10.1036/1097-8542.742600.

[3] D. F. Mujtaba and N. R. Mahapatra, "Deep Learning for Spatiotemporal Modeling of Illegal, Unreported, and Unregulated Fishing Events," in *Proceedings - 2022 International Conference on Computational Science and Computational Intelligence, CSCI 2022,* Institute of Electrical and Electronics Engineers Inc., 2022, pp. 423–425. doi: 10.1109/CSCI58124.2022.00082.

[4] A. Timofeeva, "Automatic Image Annotation with Ensemble of Convolutional Neural Networks."

[5] F. Wang, "A survey on automatic image annotation and trends of the new age," in *Procedia Engineering*, 2011, pp. 434–438. doi: 10.1016/j.proeng.2011.11.2526.

[6] "Image Classification Basics - PyImageSearch." Accessed: Nov. 27, 2023. [Online]. Available: https://pyimagesearch.com/2021/04/17/image-classification-basics/

[7] "Image Classification - an overview | ScienceDirect Topics." Accessed: Nov. 28, 2023. [Online]. Available: https://www.sciencedirect.com/topics/engineering/image-classification

[8] "Study of state of the art Image classification models and their application to Face recognition using Transfer Learning | by Lalit Kumar | Analytics Vidhya | Medium." Accessed: Nov. 29, 2023. [Online]. Available: https://medium.com/analytics-vidhya/study-of-state-of-the-art-image-classification-models-and-their-application-to-face-recognition-bdd6b3820ac

[9] "What is Unsupervised Learning? | Definition from TechTarget." Accessed: Sep. 11, 2023. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning

[10] K. Xie, L. Huang, Z. Wei, W. Zhang, and Q. Qin, "WCATN: Unsupervised deep learning to classify weather conditions from outdoor images," *Eng Appl Artif Intell*, vol. 113, Aug. 2022, doi: 10.1016/j.engappai.2022.104928.

[11] Y. M. Asano, C. Rupprecht, and A. Vedaldi, “Self-labelling via simultaneous clustering and representation learning,” Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.05371

[12] “Classification in Machine Learning: A Guide for Beginners | DataCamp.” Accessed: Sep. 11, 2023. [Online]. Available: https://www.datacamp.com/blog/classification-machine-learning

[13] K. Tran-Trung, H. Duong Thi Hong, and V. Truong Hoang, “Weather Forecast Based on Color Cloud Image Recognition under the Combination of Local Image Descriptor and Histogram Selection,” *Electronics (Switzerland)*, vol. 11, no. 21, Nov. 2022, doi: 10.3390/electronics11213460.

[14] J. A. Jose, C. S. Kumar, and S. Sureshkumar, “Tuna classification using super learner ensemble of region-based CNN-grouped 2D-LBP models,” *Information Processing in Agriculture*, vol. 9, no. 1, pp. 68–79, Mar. 2022, doi: 10.1016/j.inpa.2021.01.001.

[15] H. Xiao, F. Zhang, Z. Shen, K. Wu, and J. Zhang, “Classification of Weather Phenomenon From Images by Using Deep Convolutional Neural Network,” *Earth and Space Science*, vol. 8, no. 5, May 2021, doi: 10.1029/2020EA001604.

[16] M. Wang, S. Zhou, Z. Yang, and Z. Liu, “Clouda: A ground-based cloud classification method with a convolutional neural network,” *J Atmos Ocean Technol*, vol. 37, no. 9, pp. 1661–1668, Sep. 2020, doi: 10.1175/JTECH-D-19-0189.1.

[17] G. M. Ambildhuke and B. G. Banik, “Transfer Learning Approach - An Efficient Method to Predict Rainfall Based on Ground-Based Cloud Images,” *Ingenierie des Systemes d’Information*, vol. 26, no. 4, pp. 345–356, Aug. 2021, doi: 10.18280/ISI.260402.

[18] K. Kaviarasu, P. Sujith2, and M. G. Ayaappan, *Prediction of rainfall using image processing*. 2010. [Online]. Available: https://www.researchgate.net/publication/312290396

[19] A. Luqman Hakim and Prawito, “Rain detection in image using convolutional neural network,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Jun. 2020. doi: 10.1088/1742-6596/1528/1/012010.

[20] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting,” Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.04214

[21] "Benefits of Agile | Planview." Accessed: Sep. 03, 2023. [Online]. Available: https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/benefits-agile/

[22] "What is Agile Testing? Process & Life Cycle." Accessed: Sep. 03, 2023. [Online]. Available: https://www.guru99.com/agile-testing-a-beginner-s-guide.html

[23] S. W. Ambler, "UML Use-Case Diagrams," *The Elements of UML^{TM} 2.0 Style*, pp. 33–46, Jul. 2005, doi: 10.1017/CBO9780511817533.005.

[24] X. Wang, "Laplacian operator-based edge detectors," *IEEE Trans Pattern Anal Mach Intell*, vol. 29, no. 5, pp. 886–890, May 2007, doi: 10.1109/TPAMI.2007.1027.

[25] "Everything you need to know about VGG16 | by Great Learning | Medium." Accessed: Apr. 25, 2024. [Online]. Available: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

[26] "Understanding and visualizing DenseNets | by Pablo Ruiz | Towards Data Science." Accessed: Apr. 25, 2024. [Online]. Available: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

[27] "Classify Large Scale Images using pre-trained Inception v3 CNN model | Towards Data Science." Accessed: Apr. 25, 2024. [Online]. Available: https://towardsdatascience.com/classify-any-object-using-pre-trained-cnn-model-77437d61e05f

[28] A. Safonova, G. Ghazaryan, S. Stiller, M. Main-Knorn, C. Nendel, and M. Ryo, "Ten deep learning techniques to address small data problems with remote sensing," *International Journal of Applied Earth Observation and Geoinformation*, vol. 125. Elsevier B.V., Dec. 01, 2023. doi: 10.1016/j.jag.2023.103569.

[29] "Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification) | by Sik-Ho Tsang | Towards Data Science." Accessed: Apr. 25, 2024. [Online]. Available: https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

[30] "When the dataset is small, features are your friends | by Krzysztof Pałczyński | Towards Data Science." Accessed: Apr. 25, 2024. [Online]. Available: https://towardsdatascience.com/when-the-dataset-is-small-features-are-your-friends-6e7f8dcc819e

[31] P. Thölke *et al.*, “Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data,” *Neuroimage*, vol. 277, p. 120253, Aug. 2023, doi: 10.1016/J.NEUROIMAGE.2023.120253.

[32] E. Fahlman and F. Östlund, “Data Complexity and its effect on Classification Accuracy in Multi Class Classification Problems A study using synthetic datasets,” *DEGREE PROJECT IN TECHNOLOGY*.

[33] “ML | Underfitting and Overfitting - GeeksforGeeks.” Accessed: Apr. 25, 2024. [Online]. Available: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

APPENDICES

Code to prompt camera module to capture images at fixed time intervals at upload to AWS cloud

```
import os
import time
from datetime import datetime, time as dt_time
from picamera import PiCamera
import boto3

# Initialize the camera
camera = PiCamera()
camera.awb_mode = 'auto'  # Set white balance (you may need to experiment with these
values)
time.sleep(2)  # Allow time for the camera to adjust white balance
directory = "/home/pi/data/"
if not os.path.exists(directory):
    os.makedirs(directory)
start_time = dt_time(7, 0, 0)
end_time = dt_time(20, 30, 0)

# Function to check internet connectivity
def is_connected():
    try:
        # Try to make a request to a reliable external server
        response = requests.get('http://www.google.com', timeout=5)
        # Check if the response status code indicates success
        return response.status_code == 200
    except requests.RequestException:
        return False

# Function to upload image to AWS S3
def upload_to_s3(image_path):
    s3 = boto3.client('s3', aws_access_key_id='AKIA3FLDZOWEYC6A2BMM',
aws_secret_access_key='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')
    bucket_name = 'fypcamera'
    key = 'images/' + os.path.basename(image_path)

    try:
        s3.upload_file(image_path, bucket_name, key)
        print(f"Uploaded {image_path} to S3")
        return True
    except Exception as e:
        print(f"Failed to upload {image_path} to S3:", e)
        return False
def save_image_path(image_path):
    with open('pending_images.txt', 'a') as file:
        file.write(image_path + '\n')
while True:
    current_time = datetime.now().time()

    # Check if the current time is between 7:00 AM and 7:30 PM
    if start_time <= current_time <= end_time:
        timestamp = datetime.now().strftime("%Y%m%d-%H%M%S")
        image_name = directory + "{}.jpg".format(timestamp)

        camera.capture(image_name)
        print("Captured {}".format(image_name))
        save_image_path(image_name)
        # Check internet connectivity
        if is_connected():
            # Read pending image paths from file
            with open('pending_images.txt', 'r') as file:
                pending_images = file.readlines()
            # Upload pending images
            for pending_image in pending_images:
                if upload_to_s3(pending_image.strip()):
                    # Remove uploaded image from file
                    pending_images.remove(pending_image)
            # Write remaining pending images back to file
            with open('pending_images.txt', 'w') as file:
                file.writelines(pending_images)
        else:
            print("No internet connection. Images will be uploaded later.")
    time.sleep(300)  # 300 seconds = 5 minutes
camera.close()
```

Rule-based labelling code

```
import cv2
import numpy as np
import os

def is_blurry(region, threshold=23):
    grayscale_region = cv2.cvtColor(region, cv2.COLOR_BGR2GRAY)
    # Calculate the Laplacian of the image
    laplacian = cv2.Laplacian(grayscale_region, cv2.CV_64F)
    laplacian_var = np.var(laplacian)
   # If the variance is below the threshold, consider it as blurry (indicating rain)
    return laplacian_var < threshold
def label_images_in_directory(directory_path):
    files = os.listdir(directory_path)
    for file in files:
        if file.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(directory_path, file)
            image = cv2.imread(image_path)
            # Define the region of interest (ROI) where the tree is located
            # Adjust these coordinates based on your setup
            x, y, w, h = 350, 180, 270, 300
            roi = image[y:y + h, x:x + w]
            # Check if the region is blurry
            is_blurry_flag = is_blurry(roi)
            # Rename the image based on blurry or not blurry
            new_file_name = file[:-4] + ("-1" if is_blurry_flag else "-0") + file[-4:]
            # Construct the full path to the new file
            new_image_path = os.path.join(directory_path, new_file_name)
            os.rename(image_path, new_image_path)
def main():
    directory_path = 'picture'
    label_images_in_directory(directory_path)
if __name__ == "__main__":
    main()
```

Classification code

```
import os
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

BATCH_SIZE = 32
IMAGE_SIZE = (224, 224)
EPOCHS = 30
CLASSIFICATION_DIR = 'Classification'

datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,  # Increased rotation range
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
)

train_generator = datagen.flow_from_directory(
    os.path.join(CLASSIFICATION_DIR, 'train'),
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)
# Define data generators without augmentation for validation and testing
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = val_datagen.flow_from_directory(
    os.path.join(CLASSIFICATION_DIR, 'val'),
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)
```

```
test_generator = test_datagen.flow_from_directory(
    os.path.join(CLASSIFICATION_DIR, 'test'),
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)
# Load the VGG16 base model with pre-trained weights
vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Fine-tune the last few layers of the base model
for layer in vgg16_base.layers[:-4]:
    layer.trainable = False

# Build the model on top of the VGG16 base
model = tf.keras.models.Sequential([
    vgg16_base,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model with a lower initial learning rate
opt = Adam(lr=1e-4)
model.compile(
    optimizer=opt,
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Implement callbacks with a longer patience for early stopping
reduce_lr = ReduceLROnPlateau(factor=0.1, patience=8, min_lr=1e-6, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)

history = model.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    callbacks=[reduce_lr, early_stopping]
)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()

# Make predictions
y_pred = model.predict(test_generator)
y_true = test_generator.classes
# Convert predicted probabilities to class labels
y_pred_labels = np.where(y_pred > 0.5, 1, 0)
# Print classification report
print(classification_report(y_true, y_pred_labels))
# Plot confusion matrix
cm = confusion_matrix(y_true, y_pred_labels)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

Predictive model code

```
import os
import tensorflow as tf
from tensorflow.keras.applications import Xception

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

BATCH_SIZE = 32
IMAGE_SIZE = (224, 224)
EPOCHS = 25
CLASSIFICATION_DIR = 'Prediction'

train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
)

train_generator = train_datagen.flow_from_directory(
    os.path.join(CLASSIFICATION_DIR, 'train'),
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = validation_datagen.flow_from_directory(
    os.path.join(CLASSIFICATION_DIR, 'val'),
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
test_generator = test_datagen.flow_from_directory(
    os.path.join(CLASSIFICATION_DIR, 'test'),
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Load the pretrained Xception model
xception_base = Xception(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the pretrained layers
for layer in xception_base.layers:
    layer.trainable = False

# Add custom classification layers on top of Xception
model = tf.keras.Sequential([
    xception_base,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.8),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer=Adam(lr=1e-5),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Train the model
history = model.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    callbacks=[ReduceLROnPlateau(factor=0.1, patience=8, min_lr=1e-6, verbose=1),
               EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)]
)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()
# Make predictions
y_pred = model.predict(test_generator)
y_true = test_generator.classes

# Convert predicted probabilities to class labels
y_pred_labels = np.where(y_pred > 0.5, 1, 0)

# Print classification report
print(classification_report(y_true, y_pred_labels))
cm = confusion_matrix(y_true, y_pred_labels)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Sem 3, Year 3	**Study week no.: 4**
Student Name & ID: CHAI AI JIA 20ACB03286	
Supervisor: Dr. Ooi Boon Yaik	
Project Title: A Data Engineering Solution For Micrometeorology Forecasting Using Cloud Recognition	

1. WORK DONE I have reviewed the research materials required for the second part of the final year project. The materials focus on the methodologies for forecasting rainfall based on cloud images.
2. WORK TO BE DONE I need to enhance the setup of the camera module to capture cloud images more effectively.
3. PROBLEMS ENCOUNTERED Finding the right angle and method to set up the camera module is challenging, leading to difficulties in identifying the optimal configuration.
4. SELF EVALUATION OF THE PROGRESS The project is progressing well and is in accordance with the planed timeline.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Sem 3, Year 3	**Study week no.: 7**
Student Name & ID: CHAI AI JIA 20ACB03286	
Supervisor: Dr. Ooi Boon Yaik	
Project Title: A Data Engineering Solution For Micrometeorology Forecasting Using Cloud Recognition	

1. WORK DONE I have refined the setup of the camera module.
2. WORK TO BE DONE I need to find ways to build a predictive model to forecast the rainfall events based on the cloud images.
3. PROBLEMS ENCOUNTERED The data collected is not as many as what I have planned, and class imbalance issue occur in my dataset.
4. SELF EVALUATION OF THE PROGRESS The project is progressing well and is in accordance with the planed timeline.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Sem 3, Year 3	**Study week no.: 12**
Student Name & ID: CHAI AI JIA 20ACB03286	
Supervisor: Dr. Ooi Boon Yaik	
Project Title: A Data Engineering Solution For Micrometeorology Forecasting Using Cloud Recognition	

1. WORK DONE I have done the research on the predictive modelling.
2. WORK TO BE DONE I need to refine the predictive model built.
3. PROBLEMS ENCOUNTERED The collected data is insufficient for training the predictive model using the previous proposed method.
4. SELF EVALUATION OF THE PROGRESS The project is progressing well and is in accordance with the planed timeline.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

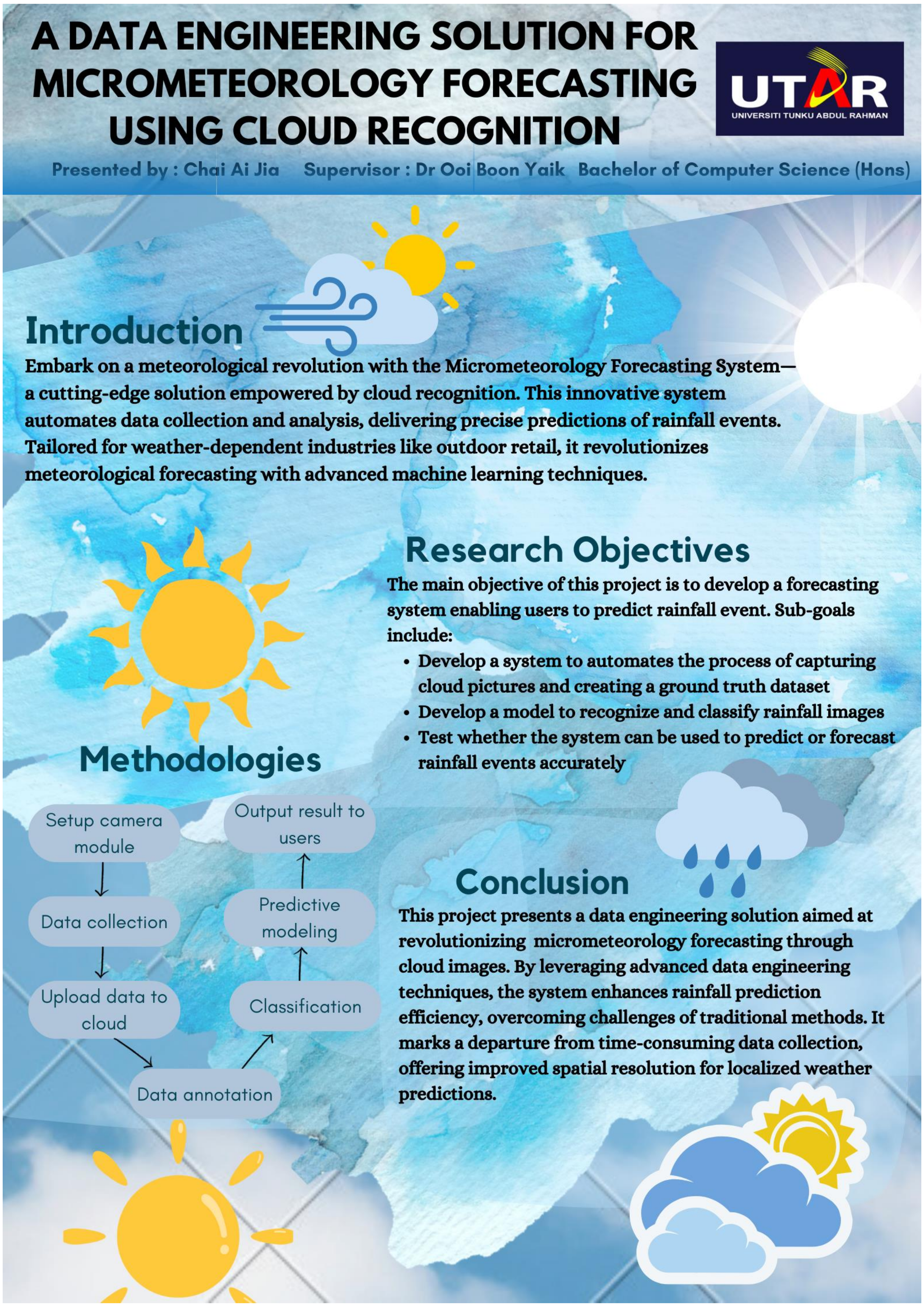
Trimester, Year: Sem 3, Year 3	**Study week no.: 13**
Student Name & ID: CHAI AI JIA 20ACB03286	
Supervisor: Dr. Ooi Boon Yaik	
Project Title: A Data Engineering Solution For Micrometeorology Forecasting Using Cloud Recognition	

1. WORK DONE I have done developing the system.
2. WORK TO BE DONE I need to refine and finalize the report.
3. PROBLEMS ENCOUNTERED I need to consult my supervisor to ensure the adequacy of my report writing style.
4. SELF EVALUATION OF THE PROGRESS The project is progressing well and is in accordance with the planed timeline.

[illegible]
Supervisor's signature

[illegible]
Student's signature

POSTER



A DATA ENGINEERING SOLUTION FOR MICROMETEOROLOGY FORECASTING USING CLOUD RECOGNITION

UNIVERSITI TUNKU ABDUL RAHMAN

Presented by : Chai Ai Jia   Supervisor : Dr Ooi Boon Yaik   Bachelor of Computer Science (Hons)

Introduction

Embark on a meteorological revolution with the Micrometeorology Forecasting System—a cutting-edge solution empowered by cloud recognition. This innovative system automates data collection and analysis, delivering precise predictions of rainfall events. Tailored for weather-dependent industries like outdoor retail, it revolutionizes meteorological forecasting with advanced machine learning techniques.

Research Objectives

The main objective of this project is to develop a forecasting system enabling users to predict rainfall event. Sub-goals include:

- Develop a system to automates the process of capturing cloud pictures and creating a ground truth dataset
- Develop a model to recognize and classify rainfall images
- Test whether the system can be used to predict or forecast rainfall events accurately

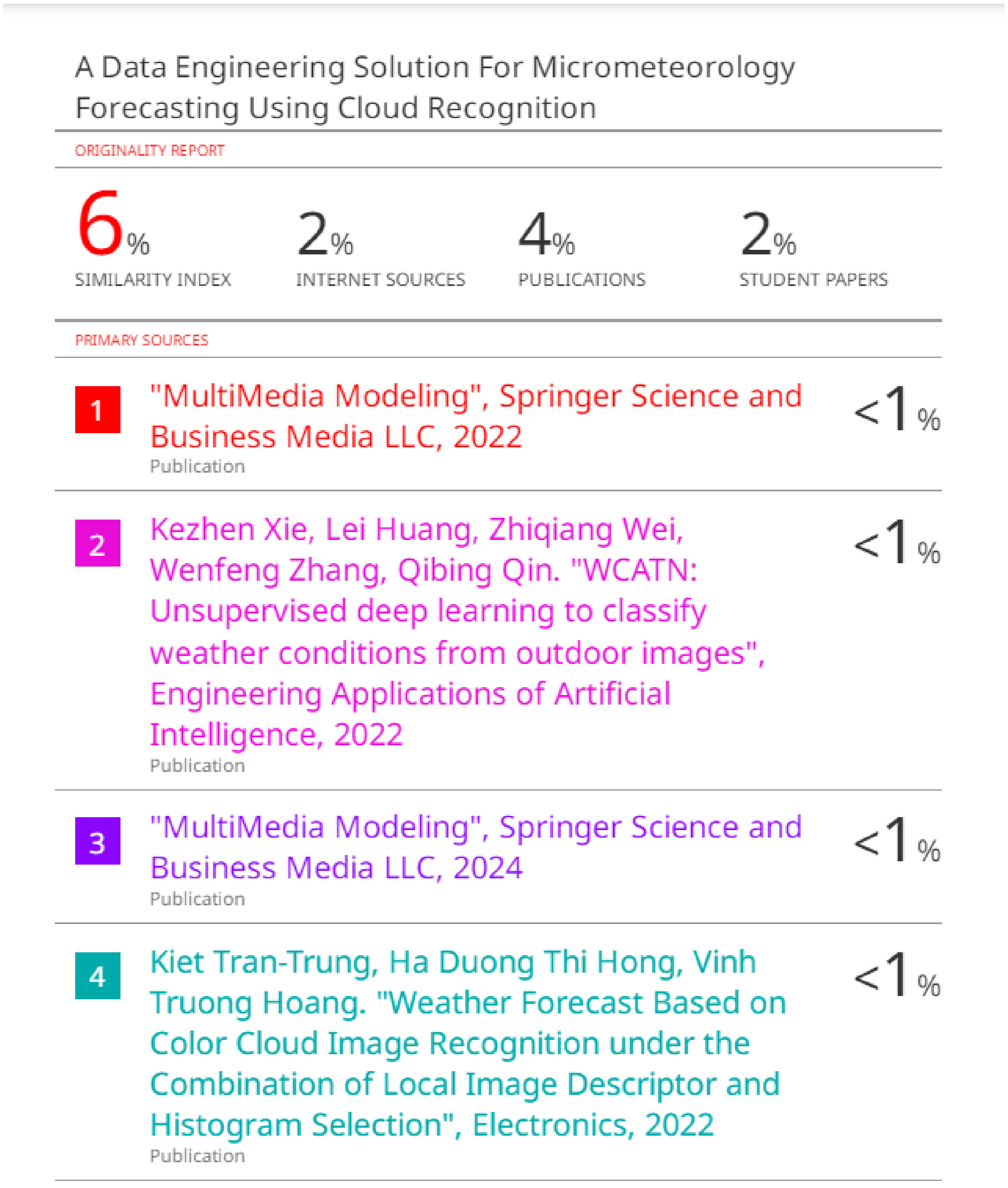
Methodologies

$$\begin{tikzcd}
\text{Setup camera module} \arrow[d] & \text{Output result to users} \\
\text{Data collection} \arrow[d] & \text{Predictive modeling} \arrow[u] \\
\text{Upload data to cloud} \arrow[dr] & \text{Classification} \arrow[u] \\
& \text{Data annotation} \arrow[u]
\end{tikzcd}$$

Conclusion

This project presents a data engineering solution aimed at revolutionizing micrometeorology forecasting through cloud images. By leveraging advanced data engineering techniques, the system enhances rainfall prediction efficiency, overcoming challenges of traditional methods. It marks a departure from time-consuming data collection, offering improved spatial resolution for localized weather predictions.

PLAGIARISM CHECK RESULT



A Data Engineering Solution For Micrometeorology Forecasting Using Cloud Recognition

ORIGINALITY REPORT

6%	2%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"MultiMedia Modeling", Springer Science and Business Media LLC, 2022 Publication	<1%
2	Kezhen Xie, Lei Huang, Zhiqiang Wei, Wenfeng Zhang, Qibing Qin. "WCATN: Unsupervised deep learning to classify weather conditions from outdoor images", Engineering Applications of Artificial Intelligence, 2022 Publication	<1%
3	"MultiMedia Modeling", Springer Science and Business Media LLC, 2024 Publication	<1%
4	Kiet Tran-Trung, Ha Duong Thi Hong, Vinh Truong Hoang. "Weather Forecast Based on Color Cloud Image Recognition under the Combination of Local Image Descriptor and Histogram Selection", Electronics, 2022 Publication	<1%

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	CHAI AI JIA
ID Number(s)	20ACB03286
Programme / Course	CS / UCCC3596 PROJECT II
Title of Final Year Project	A DATA ENGINEERING SOLUTION FOR MICROMETEOROLOGY FORECASTING USING CLOUD RECOGNITION

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 6 % **Similarity by source** Internet Sources: 2 % Publications: 4 % Student Papers: 2 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Alex	
Signature of Supervisor	Signature of Co-Supervisor
Name: Dr. Ooi Boon Yaik	Name:
Date: 26/4/2024	Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB03286
Student Name	CHAI AI JIA
Supervisor Name	Dr. Ooi Boon Yaik

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
N/A	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]
____
(Signature of Student)
Date: 26/4/2024