

**WEATHER RECOGNITION BASED ON STILL IMAGES USING
DEEP LEARNING NEURAL NETWORKS**

BY

PEACE ULOMA EGBUEZE

STUDENT NO: 63947129

Submitted in accordance with the requirements for the degree of

MASTER OF SCIENCE

In the subject

Computer Science

At the

University of South Africa

Supervisor: Professor Zenghui Wang

DATE: 24 JANUARY 2024

DECLARATION

Name: Egbueze peace Uloma
Student number: 63947129
Degree: Master of Science (MSc): Computer Science
Place: University of South Africa, Florida Science Campus

Weather recognition based on still images using deep learning neural networks.

I declare that the above dissertation/thesis is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.



SIGNATURE

24-01-2024

DATE

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my supervisor, Professor Zenghui Wang, from the department of electrical and mining engineering at the University of South Africa (UNISA) who has guided me a lot during the period of this research, and for the precious time he dedicated to correcting every chapter of this research and providing feedback to improve the quality of work.

I am thankful to the staff at the University of South Africa, in the school of computing for the amazing support provided in the form of necessary information on available resources used in this study.

Finally, I would like to extend a special thanks to my family and friends, and especially my parents, Mr. and Mrs. Ulakpa, my siblings Chuks, Fidelis, Wisdom, Desmond, and Justicial for their immense prayers, goodwill messages and encouragement. A special thanks to my friend grace Mugwejendzi for her motivations and inspirations.

DEDICATION

I dedicate this study to my lovely husband Emmanuel, for his constant source of joy, love, and support. To our lovely children, Jeffrey and Jethro for their extreme understanding and patience in affording me the needed space to focus on the research work. Most importantly, a special thanks to the Almighty God for granting me the grace to accomplish this MSc research work.

PUBLICATION

Egbueze Peace and Zenghui Wang, "Weather recognition based on still images using deep learning neural network", Proceedings of the ICDLT 2022 6th International Conference on Deep Learning Technologies (ICDLT), Xi'an, China, July 2022, pp. 8 -13.

ABSTRACT

Weather recognition from still images remains a challenge due to weather diversity and lack of distinct characteristics amongst weather conditions. The building blocks of deep learning involves a lot of hyperparameters, which is difficult to tune manually. Fine-tuning Hyperparameter is a crucial part of building a good deep learning model. This study proposes a simplified ResNet-15 model fine-tuned by two distinct optimisers, SGD, and Adam for the purpose of optimising the momentum, number of dense layers, learning rate, batch size and dropout rate to find the optimal hyperparameters that gives the best performance on the model. The Convolutional layers are used to extract the most related visual features, then the images are classified through the fully connected layers of the Softmax classifier. To evaluate the recommended approach, a comparison of hyperparameter tuning with and without hyperparameter tuning of ResNet-15 during the experiments shows that fine-tuning of ResNet-15 hyperparameter using random search optimisation method gives more accurate result with accuracy of 97.29% than other techniques.

KEY TERMS:

Deep learning; SGD; Resnet-15; Classifier; SoftMax; hyperparameter; optimisers; weather diversity; Adam; fully connected layers.

Table of contents

Preliminaries

Declaration	ii
Acknowledgments.....	iii
Dedication.....	iv
Abstract	v
Table of contents.....	vi
List of figures.....	x
List of tables.....	xii
List of acronyms	xiii
1 Introduction.....	1
1.1 Background.....	1
1.2 Motivation and overview.....	3
1.3 Problem Statement.....	6
1.4 Hypothesis.....	7
1.5 Aims and Objectives.....	8
1.6 Research Questions.....	8
1.7 Contributions.....	9
1.8 Dissertation Structure.....	10
2 Literature Review.....	11
2.1 Chapter overview.....	11
2.2 Traditional Methods of Weather Recognition.....	11
2.3 Machine Learning Models	13
2.4 . Deep Learning Models	14

2.4.1 The taxonomy of several typical weather recognition approaches.....	18
2.4.1.1 Supervised or Discriminative Learning.....	19
2.4.1.2 Convolutional Neural Networks (CNNs)	19
2.4.1.3 Recurrent Neural Network (RNN).....	20
2.4.1.4 Autoencoders.....	20
2.5 CNN Architecture.....	22
2.5.1 The Convolutional Layer	24
2.5.2 The Pooling Layer	24
2.5.3 The Fully Connected Layer.....	25
2.5.4 Activation Function.....	25
2.6 Techniques for Reducing Overfitting	26
2.7 The Different Variants of CNN Approaches	28
2.7.1 LeNet.....	28
2.7.2 AlexNet.....	29
2.7.3 VGG-16	30
2.7.4 GoogleNet.....	30
2.7.5 ResNet-50	31
2.8 Chapter Conclusion	31
3 Research Methodology.....	32
3.1 Chapter Overview.....	32
3.2 Data Design.....	32
3.2.1 Data Collection	33
3.2.2 Image Preprocessing.....	35
3.2.3 Image Scaling.....	35
3.3 Residual Networks.....	36

3.3.1 Residual Network Architecture.....	37
3.4 ResNet Method.....	40
3.5 Hyperparameter Optimisation,,,,,.....	42
3.5.1 Hyperparameter Setup.....	43
3.6 Chapter Conclusion.....	44
4 System Design and Implementation.....	45
4.1 Chapter overview.....	45
4.2 Configuration and setup of Experimental Tools	45
4.2.1 Python Libraries	45
4.2.2 System Configuration	45
4.3 Feature Extraction and Selection.....	46
4.4 Model Design and Selection	46
4.5 Hyperparameter Tuning using Keras Tuner	48
4.6 Optimisation Method	50
4.6.1 Random search optimisation.....	50
4.7 System Development Life Cycle	51
4.8 Chapter Conclusion	52
5 Experimental Results and Analysis.....	53
5.1 Chapter overview.....	53
5.2 Resnet-15 Experimental Method.....	53
5.3 Experimental Results.....	54
5.4 Impact of Hyperparameter Optimisation	58
5.5 Accuracy Comparison to other Methods	60

5.6 Chapter Conclusion	64
6 Conclusion and Feature work.....	65
6.1 Chapter Overview	65
6.2 Conclusion	65
6.3 Limitations	66
6.4 Future Work	66
References.....	68
Appendix A – Sample python codes	76
Appendix A1- Listing one - Image loading Script	76
Appendix A2- Listing two - Image Resizing Script.....	77
Appendix A3- Listing three – split of rain and test data for Feature extraction.....	78
Appendix A4- Listing four –Augmenting image dataset script.....	79
Appendix A5- Listing five –dropout rate script.....	81
Appendix A6- Listing six–final model script.....	83
Appendix A7- Listing seven–recognition on new images script.....	84
Appendix A8- Listing seven–effects of hyperparameter optimisation script.....	85
Appendix A9- Listing seven–recognition on various methods.....	87
Appendix B – Approved Ethical Clearance.....	94
Appendix C- Editing Services.....	100

List of figures

Figure 1.1: Various outdoor images.....	4
Figure 1.2: Certain significant regions for classification.....	5
Figure 2.1: Deep neural network	18
Figure 2.2: The taxonomy of several typical weather recognition approaches	17
Figure 2.3: A demonstration of deep learning-based, auto encoder algorithm	21
Figure 2.4: The illustration of CNN and RNN Approach for Multi-label Weather.....	23
Figure 2.5: The Pipeline of CNN Image Classification	24
Figure 2.6: The Activation Function of ReLu	26
Figure 2.7: The Softmax Activation Function.....	26
Figure.2.8: A deep neural network	27
Figure 2.9: Example of overfitting	27
Figure.2.10: A neural network before dropout	27
Figure 2.11: ANN after applying dropout	27
Figure 2.12: LeNet architecture	29
Figure 2.13: Alex-net architecture	29
Figure 3.1. Images Per Class.....	34
Figure 3.2: The four samples of the Dataset.....	35
Figure 3.3: Data Augmentation Result.....	36
Figure 3.4. A Residual Unit	37
Figure 3.5. A Basic Resnet Representation	38
Figure 3.6. Full Resnet pre-activation.....	39
Figure 3.7. The two Simple modes of residual module	41

(a) Identity Block	41
(b) Convolution block	41
Figure 3.8: Resnet-15 Architecture	41
Figure 3.9 The flowchart of the proposed Resnet-15 method.....	42
Figure 4.1The Confusion matrix of various approaches. (a) AlexNet. (b) VGG16. (c) GoogleNet. (d) ResNet-50. (e) ResNet-18. (f) ResNet-15.....	48
Figure 5.1 Image of the feature maps from Convolutional layers of Resnet-15	54
Figure 5.2: Comparing accuracy of different dropout rates	56
Figure 5.3 (a) Examples of correct results recognition	57
Figure 5.3 (b) Examples of incorrect results recognition.....	57
Figure 5.4 Accuracy curve comparison.....	59
(a) Curves of accuracy for ResNet-15 without hyperparameter optimisation.....	59
(b) Accuracy curves of ResNet-15 with hyperparameter optimisation	59
Figure 5.5 Accuracy curves of several methods (a)Training accuracy (b) Training Loss curve (c) Validation accuracy .(d) Test accuracy.....	61
Figure 5.6 Comparison of average recognition accuracy	63

List of tables

Table 2.1: Typical weather recognition approaches with their strengths and weaknesses	22
Table 3.1: Details of multi-weather image dataset.....	33
Table 3.2: Resnet Structure.....	38
Table 3.3: The difference between Resnet-50 and Resnet-15.....	39
Table 3.4: Hyperparameter configuration range	44
Table 4.1. Optimal combination of hyperparameters.....	49
Table 5.1: Recognition accuracy of two approaches without and with hyperparameter optimisation	59
Table 5.2: Recognition accuracy of various approaches.....	63

List of Acronyms

ANN	Artificial neural network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
BGD	Batch Gradient Descent
DNN	Deep neural Network
FC	Fully Connected
GPU	Graphic Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ReLU	Rectified Linear units.
RNN	Recurrent Neural Network
RSCM	Region Selection and Concurrency Model
SGD	Stochastic Gradient Descent
OPT	Optimisation

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Our daily lives are impacted by the weather condition (Lu et al., 2014). Understanding of the weather conditions plays a crucial role in human activities, such as dressing, traveling, sporting, manufacturing, farming, aviation, production, and solar technologies etc. Therefore, obtaining weather conditions automatically is important to avoid natural phenomenon.

Traditionally, human observation has been the basis for recognising weather conditions. However, it is labor intensive and leads to errors to distinguish between weather conditions using the conventional artificial image approaches. Therefore, the development of highly accurate, effective, and automatic systems for the recognition of weather conditions are urgently needed. Other traditional method of weather recognition models previously used also depended heavily on expensive sensors to recognise the weather conditions, of which, the set up and repairs of the expensive sensors takes so much material resources including manpower, which restricts scalability of analysing local weather conditions for many locations. Cameras usually cost less than sensor devices for weather recognition due to the following reasons:

- ❖ Flexibility: Cameras capture different weather conditions, thus providing visual information about it. In fact, only specialised sensor devices collect information about the weather because it is limited to one function (Smith et al., 2019).
- ❖ Low production cost : The Low-cost during production is attributed to the wide application range of cameras. In contrast, the production cost of sensor devices that are intended for weather recognition increases because more specialized technologies and components are necessary (Johnson et al., 2020).
- ❖ Using the existing Infrastructure: Several areas have installed cameras which are meant for safety purposes such as observing and controlling traffic. Given the circumstance that it would cost more if we were to purchase additional sensors for

this purpose, hence using cameras that has sensors to recognise weather conditions would be more economical. (Williams et al., 2018).

- ❖ Efficient data processing: Data processing can be done efficiently using computer vision algorithms, which help in recognising weather patterns from visual images captured from cameras. Although sensors require intricate calibration and interpretation algorithms, processing is fast nowadays because of current computer abilities. (Brown et al., 2021).
- ❖ Maintenance: Maintaining cameras takes less effort when compared to sensor devices, sensor devices require protection from extreme weather conditions, regular calibration checks, as well as uninterrupted power supply.

In previous years, weather recognition has been based on the use of tools such as, sensors. Although the hardware equipment is normally expensive and depends on experts for maintenance. Thus, the need for a different method to recognise weather condition from still images with the use of computer vision techniques (Elhoseiny et al., 2015). Computer vision techniques has become important in many applications (Han & Cheng, 2018), such as image retrieval, (Qayyum et al., 2017), image restoration (Liu et al., 2011), and the reliability improvement of outdoor surveillance systems (Huang et al., 2015), and vehicle assistant driving systems (Kurihata and Luo, 2009) can also benefit because of weather recognition.

Deep learning neural networks have become more popular for image classification such as computer visions. Deep Neural Networks (DNNs) are multi-layered feed forward neural networks (Goodfellow & Bengio, 2016), mostly, deep convolutional neural networks (CNNs) which learns features automatically from raw images. These are made up of simple processing components known as artificial neurons.

Support vector machine (SVM) method was used for recognising weather conditions from images by extracting features such as contrast, slope, power spectrum, saturation, and noise, (Li et al., 2014; Jindal et al., 2016). This technique was tried by using a small amount of data. It achieved a recognition error rate for rainy and foggy weather of 25%, and 15%

respectively. (Narasimhan et al., 2002). The dataset was very small and not enough, which led to lack of generalisation, and robustness. However, most of the approaches based on machine learning were designed for manual feature extraction and abstraction of weather characteristics. Besides, these approaches were very complex and usually led to weak generalisation capabilities.

Currently, deep learning is fast-growing, and convolutional neural network (CNN) has achieved incredibly in several machine vision tasks such as, image classification, semantic segmentation and object detection, (Tang et al., 2017). Many deep learning models faces gradient problems at the lower layers when training. To resolve this problem, an approach that functions across many domains are needed to train neural networks from end-to-end. CNNs can extract very rich abstracts and deep semantic information from weather images, which makes them perform better than the conventional techniques for weather recognition. Insufficient flow of gradient has become an obstacle for training deeper networks, a different technique is recommended for weather recognition using ResNet-15 which uses the means of identity mappings to remove the vanishing gradients problem and make it possible to train very deep networks.

To build CNNs, the machine learning researcher must manually adjust a few configurations. Hyperparameters are the variables of the network structure and the network parameters that are trained and adjusted in a CNN (Aszemi et al., 2019). The main goals of hyperparameter optimisation are to minimize the loss or to achieve a satisfactory model. Though it is computationally costly to test every possible combination of hyperparameters.

1.2 MOTIVATION AND OVERVIEW

Though weather recognition is of high importance, works related to weather recognition using image processing are quite few, most works done were conducted for vision-based driver assistance systems. Other authors explored recognition from a particular outdoor image, of which they referred to recognition of weather conditions as a single classification task, which seems inappropriate. The inappropriateness can be explained based on two

major reasons. The first reason being ambiguity, that is to say, the class limits among some weather categories are basically uncertain. For instance, as seen from Figure 1.1

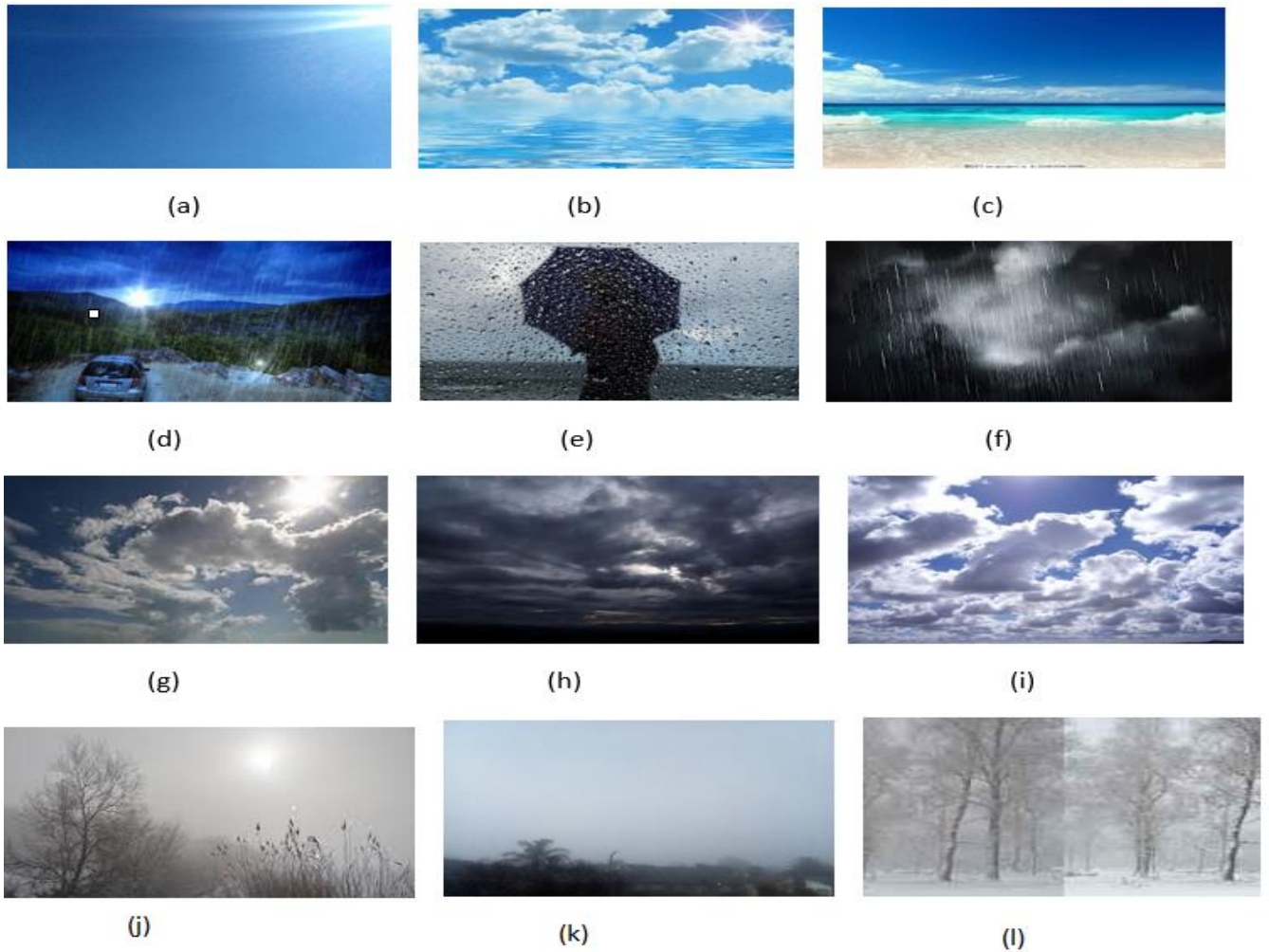


Figure 1.1: Various outdoor images. Some of the images were taken from Boksburg area of Johannesburg South Africa, and some from dataset 2. (a) sunny image but clear cloud. (b) Sunny image with a very few clouds. (c) Comparing with (b), image has more clouds (b). (d) and(e) present both rainy and sunny features. (f) A rainy and cloudy image. (g) Shows a cloudy and sunny feature. (h) Cloudy image with no sun. (i) cloudy image with sunny features. (j) foggy image with a clear sun on its background. (k) A foggy image that looks more overcast. (l) A foggy image with obvious snow on the ground.

The changes from Figure 1.1 (a) to (l) demonstrates that there is a sequence of conditions between a clear sunny weather such as Figure 1 (a) and an obvious cloudy weather as Depicted in Figure 1.1 (h), it is also difficult to determine whether the category in Figure1.1(i) becomes sunny or cloudy when one is referring to immediate weather conditions such as Figures 1.1 (c), (g) and (h). Therefore, the ambiguity of such borderline models gives bases of the complexity of determining the ground-truth categories from the human standpoint, moreover, a very little work done previously presents solutions to the uncertainty challenges. Another shortcoming of tackling weather recognition as a single classification task can be seen as insufficiency, the distinct weather category may not refer to the weather conditions widely for the specific image. An instance, haze is obvious from the visual effect of images in Figures 1.1 (j), (k) and (l). However, when comparing these three images, it can be seen from Figure 1.1(j) appears sunnier while Figure1.1 (k) looks much like overcast, while Figure1.1 (l) appears snowy. Hence, just a haze category may not tell the fluctuations among the three images.

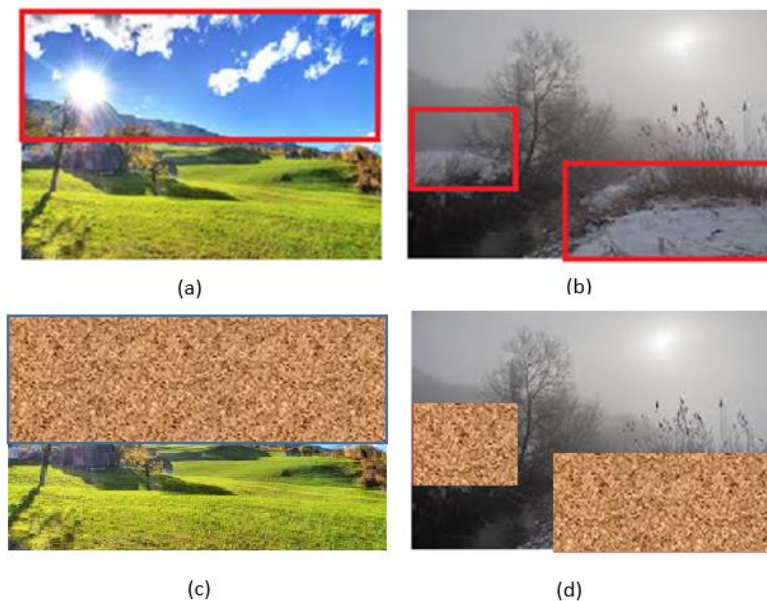


Figure 1.2: Demonstration of some specific regions for image recognition. (a) Sunny condition with sky blue. (b) A snowy weather with obvious snow at the background. (c) And (d) when obstructing the vital regions, becomes a bit complex to assess the weather condition.

Inspired by the above-mentioned two reasons, this work understands recognition task as a challenge. That is, allocating an image according to the weather condition present. This can be accomplished using a convolutional neural network architecture. This insight comes from two parts. The first is that some of the earlier research concentrated on using handcrafted weather features (Lu et al.2014), (Zhang et al. 2016), although the features could not attain the anticipated outcomes for the recognition task. Motivated by this huge success in Convolutional Neural Network (CNN) over the years, this work utilises CNN for feature extraction. Secondly, classes display a good co-existence reliance in the weather field. An instance, where snow and fog commonly happen together, while rain and snow almost never happen at the same time. As can be seen from Figure 1.2, the sky blue is very vital for assessing a sunny day, and snow at the background is important for the snowy weather estimation. (Lu et al., 2014), (Lin et al., 2017) emphasises that such weather cues are severe. Thus, this is required towards making the weather cues discriminative.

In a situation where the accessible data is statistically limited, another suitable approach to use is Transfer Learning, which is becoming more popular (Yosinski et al.,2014). This method uses the knowledge formerly obtained for solving a new task, which may not really be interrelated to the previous task.

The aim of this work is to distinguish the presence of the weather condition from the given image. A first step is to look at the different group of features extracted from weather images, secondly, utilise an image recognition method that can reliably differentiate between certain weather conditions. Weather images were collected from online sources like, Flickr, Wikimedia, Pixabay, Sky finder, dataset2, including other search engines.

1.3 PROBLEM STATEMENT

One of the most important issues in analysing weather conditions from still images is how well we can recognise the conditions and their impacts in areas such as agriculture, transportation, and disaster management.

A system is being developed in this study that will automatically recognise different weather conditions such as cloudy foggy, rainy, sunny, or extreme weather conditions

involving storms or hurricanes by using still images. The system must be capable of processing images taken at various places with high accuracy and speed.

The key problems are:

- ❖ Inconsistency in weather patterns: Weather patterns vary across different regions and can change quite fast. Instead, the system needs to be capable of handling this variability and adapting to various weather conditions efficiently.
- ❖ Data quality and noise: Still images may exhibit noise, objects, or obstructions that can obscure weather recognition accuracy. It is important to develop strategies that will filter such data challenges.
- ❖ Integration with pre-existing Infrastructure: Leveraging the extensive coverage provided by the existing camera networks or satellite images, as well as keeping up the performance and reliability are important in the seamless integration.
- ❖ Multi-class weather: Recognising weather types is a complex task that requires developing algorithms which can classify weather images such as rainy ,cloudy, foggy, and sunny among others.
- ❖ Efficiency and scale: Scale as well as efficiency matter most since the system deals with huge amount of image data where performance in terms of scalability and computational efficiency has become paramount. It is therefore important that algorithms be optimised for speed and resourceful use.

1.4 HYPOTHESIS

If the multiple non-linear layer of deep learning neural network approaches a limit or infinity function, then it is equal to hypothesize that they can approach the limit of the residual function, after going through this hypothesis, what comes to the mind is What is a residual Function?

Residual function also called residual mapping, is the difference between the input and output of the residual block. Thus, residual mapping is the value that is added to the input to estimate the final function of the block. Further explanation to address the hypothesis is detailed in section 3.3 and Figure 3.4 has more information on the hypothesis.

1.5 OVERALL AIMS AND OBJECTIVES

The primary focus of this study is to devise methods of addressing the research question. By understanding, analysing, reducing, evaluating, and applying measures to overcome the constraints and challenges in the optimisation of hyperparameters for deep learning model.

The aims and objectives of this study are.

- 1) **Enhancing Performance:** The main aim is to increase the performance of deep learning by searching for the best values of several hyperparameters as well as getting better accuracy, loss functions reduction and overall model quality.
- 2) **Training time reduction:** The goal of this work is to reduce the training time for deep learning models by selecting efficient hyperparameter configurations. Which will result in higher productivity as well as speed up experimentation time.
- 3) **Improving generalisation:** deep learning can help with challenges of overfitting and underfitting where they do not generalize well to unseen data. The optimisation of the hyperparameters would help improve generalisation, thereby allowing the models to perform well on both training and testing datasets, by reducing the time and efforts required for training
- 4) **Random search optimisation:** Using techniques such as random search optimisation would help to reduce computational costs by efficiently exploring the hyperparameter space. This would help to tackle the key problems with hyperparameter optimisation in deep learning models which comprises of computational costs and overfitting.

1.6 RESEARCH QUESTIONS

Optimising the hyperparameters of deep neural networks to perform weather recognition from images has not been extensively done. To address the problems that deep learning neural works faces when optimising them, the following research questions were asked.

- 1) What is the impact of optimising deep learning hyperparameters on the overall performance and accuracy of the model?
- 2) How does the selection of different optimisation algorithms for tuning hyperparameters affect the training time and convergence of deep learning models?
- 3) How effectively should the optimization of hyperparameters in deep learning models be automated to save time and labor?
- 4) What are the challenges linked with optimising hyperparameters for deep learning and how can they be prevented?

1.7 CONTRIBUTION

By optimising the hyperparameters of deep learning, including adjustment of hyperparameters such as, momentum, dense layer number, batch size, and dropout presents the following contributions

- 1) The model's superior performance is attributed to the adjustment of these hyperparameters. It is worth noting that the learning dynamics and generalisation abilities of the model are affected by each hyperparameter in a different way, thus, selecting appropriate values for momentum, learning rate, batch size, and dropout has yield accuracies through faster convergence speeds as well as avoidance of overfitting.
- 2) The enhancement to convergent acceleration was implemented by adjusting the momentum hyperparameter to ensure a better control as well as search balance over hyper-parameters during the optimization phase; this has consequently reduced training time as well as led to quicker achievement of expected result.
- 3) Tuning the batch size hyperparameter has helped to balance batches that are either too small or too large thereby enhancing generalisation. In the study a good batch size was necessary to make the model generalize well on unseen data.
- 4) The dropout used during the training process helped to prevent the variance of the neurons in the model, making it more robust. Evidently, tuning a hyperparameter for .dropout was one way that contributed to finding how much

regularization should be applied such that the risk from overfitting is reduced, yet preserving the model's accuracy.

1.8 DISSERTATION STRUCTURE

The subsequent part of this dissertation is organised as follows;

Chapter two: Describes the reviewed literature around deep learning together with previous work related to weather recognition involving various CNN and their architectures respectively.

Chapter three: Gives details of the construction methods for the dataset. Also illustrates the data creation and presents the deep learning neural network architecture and the methods used in this study.

Chapter four: Entails the implementation and development of the ResNet algorithm and the design of the neural network to recognise weather conditions. It also involves the overview of hyperparameter optimisation.

Chapter five: This chapter describes the experiments performed, it outlines the overview of training and testing of the neural network. It also gives the details of the results obtained from the experiments and presents the outcomes in contrast to various deep learning techniques

Chapter six: Draws conclusions and summary, then provide recommendations for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 CHAPTER OVERVIEW

This chapter presents the background and related work of the thesis. Section 2.1 presents an outline of past works on traditional methods of weather recognition models, 2.2 covers the machine learning methods of weather, 2.3 entails the deep learning methods of weather recognition, Section 2.4 discusses in depth of deep learning Neural Networks and in Section 2.5, a look at the how to reduce overfitting and the overview of CNN, while 2.6 presents the deep residual networks and 2.7 the chapter conclusion.

2.2 TRADITIONAL METHODS OF WEATHER RECOGNITION

In the past weather recognition was traditionally done by hand-crafted feature extraction. These methods often rely on handcrafted feature extraction that may not generalise well to new data, also, the traditional models are not capable of learning complex and abstract features that are possible with machine learning feature extraction methods. Several vehicle assistant driving systems (VADs) used weather recognition to improve road safety. For instance, the limit in speed of adverse weather condition can be set, automatic opening of the car wipers. Hand-crafted features were common to the work by. (Kurihata et al., 2005, Q. Li et al., 2014) and recommended rain drop as a powerful cue in detecting the existence of drizzly weather, also, developed a rain feature to sense rain drops on the windshield.

(Roser et al., 2008) discovered several regions of interest (ROI) and developed different types of histogram features for rainy conditions. (Yan et al., 2009) utilised gradient amplitude histogram, Hue Saturation Value (HSV) color histogram as well as road information for the classification task among sunny, cloudy, and rainy categories. Moreover, many methods were proposed particularly for fog recognition, (Hautière et al., 2006) used Koschmieders Law to detect the presence of fog and estimated the visibility distance. (Bronte et al., 2009) utilised many techniques, including a Sobel based sunny-foggy detector, edge binarization, though line detection, vanishing point detection and road and sky segmentation. (Gallen et al., 2011) based his work for fog recognition by

discovering back scattered shroud from the car lights. (Pavlic et al., 2012; Pavlic, 2013) converted an image using the rate of recurrence and identified the existence of fog through training different scaled and oriented Gabor filters in the power spectrum. However, the above-mentioned approaches have shown good performance, though, were usually limited to the in-vehicle standpoint and cannot be applied to a broader range of applications.

In recent times, several research have been devoted to weather recognition from common outdoor images. (Shen et al., 2009) proposed a photometric stereo-based approach to estimate weather condition of a given site. (Zhao et al., 2011) pointed out that pixel-wise intensities of dynamic weather conditions (rainy, snowy, etcetera.) changes over time while static weather conditions (sunny, foggy, etcetera.) almost stays unchanged. They suggested a two stage classification scheme which first distinguishes between the two conditions and then utilises several spatio-temporal and chromatic features to further estimate the weather category. (Song et al., 2014) extracted numerous global features to be classified, such as, edge gradient energy, saturation, contrast, power spectral slope, and noise. (Li et al., 2014) similarly utilised several features in Song et al., work and built a decision tree, according to the distance between features. Except for regular global features, (Lu et al., 2014) suggested several cues as well as shadow and sky descriptor for two-class weather recognition and reflection. (Zhang et al., 2015, Zhang et al., 2016) proposed the sunny feature, rainy, snowy, and haze features independently for each weather class as well as two global features. Furthermore, a multiple kernel learning approach is proposed in (Zhang et al., 2015) work to fuse these features. (Derpanis et al., 2012) formulated spatial appearance and temporal dynamics to investigate audiovisual clip which could recognise several weather types.

The ability to generalise on a new dataset has made it complex for important features to be extracted from the images. For this reason, (Lu et al., 2014) proposed a two-class weather classifier which classified images based on five features (Sky, Haze, Contrast, Reflection and Shadow). Their work was extended by the authors, by contacting CNN features in the feature vector. (Zhang & Ma, 2015) relied on manual feature identification and extraction for the improvement of their weather classifier. The authors used global

and local features, to represent the images. In their work, local features represent certain characteristics of weather conditions, this may not be present in images depicting a different kind of weather.

2.3 MACHINE LEARNING MODELS

Machine learning is a type of artificial intelligence (AI) which centers on using data and algorithms to mimic the way human learns, by gradually improving the accuracy rate.

(Roser & Moosmann, 2008) proposed a model to classify weather conditions using a Support Vector Machine (SVM) trained on single color images. Though, the model was limited to recognising three weather conditions, light, heavy rain, and clear weather. (Chu et al., 2017) employed a random forest classifier to classify weather conditions using tagged images of weather conditions, like, (sunny, foggy, and cloudy). Finally, (Zhang et al., 2016) formulated a method to classify weather conditions based on a general framework that aims to extract multiple features such as, sky, rain, snowflakes, shadow, dark channel, saturation, and contrast relying on SVM and k-nearest neighbors. Yet, the suggested models were rather limited to a specific weather classification.

In a study that was published, (Lisha et al., 2018) first proposed the Hyperband method of hyperparameter optimisation. According to the authors study, hyperparameter optimisation is a non-stochastic, finite-armed bandit problem where a given resource, such as features, data samples, or iterations is distributed among randomly selected configurations. Hyperparameters refer to the adjustments made to machine learning models' parameters that are not learnable during the training process. The process of fine-tuning hyperparameters to improve the performance is known as hyperparameter optimisation. To get the best performance out of a machine learning model, hyperparameter optimisation is essential. A manual process of trial and error is frequently used in hyperparameter optimisation, where the developer tries out different configurations and develops new ones based on experience and what has been observed. hyperparameter optimisation has attempted to automate the process from manual hyperparameter optimisation to automatic. The primary motivation for automated

hyperparameter optimisation is to save the developers time because the manual process is very labour intensive. Enhancing performance is another reason for automated hyperparameter optimisation. some studies have demonstrated that hyperparameter optimisation mostly selects better settings than when done manually. (Melis et al., 2017)

2.4 DEEP LEARNING MODELS

Deep learning is a type of machine learning that is based on artificial neural networks (ANN). It can learn complex patterns and connections within a given dataset. DNN consist of three layers of neural network, which includes input, hidden, and output layer. The input data is fed into the input layer, the output layer displays the output data. While the hidden layer is accountable for performing all the calculations and every other hidden task. Few approaches have been used in the computer vision field to do the weather recognition task from images. An advantage of deep learning algorithms is its ability to discover and learn good representations using feature learning. The general methods involve feature abstraction from images which can be used in image recognition algorithms (Bossu, Hautière & Tarel, 2011), or machine learning techniques. Deep learning architectures have become a hopeful tool as regards image classification and analysis, (Goodfellow et al., 2016), Figure 2.1 illustrates the three different layers of a typical DNN

The research area of computer vision changed significantly over recent years, mostly due to the advances made around DNN (Krizhevsky A, Sutskever I, Hinton, 2012). DNN was employed as an effective machine in an artificial neural network for resolving complicated glitches (Bengio, 2009). This network allows automatic feature extraction from raw data, known as feature learning. Deep neural networks can rebuild the original raw data set, learn features with neural networks (NN) instead of selecting features manually that was done traditionally (X. Wang and Q, 2004).

The computer vision techniques that rely on the DNN prototypes, particularly, CNN models have been seen as a promising tool in recognition task by (LeCun et al., 2015).

Many applications based on categorization, segmentation, and localization of pixels from images has turn out to be a general method based on the different components of an urban scene according to (Chaurasia & Culurciello, 2017; Li et al., 2017; Yang et al., 2018; Zhou et al., 2017). Also, several models have been implemented to classify weather from the extracted features based on a convolutional building block of deep learning models. A CNN model combined with sparse decomposition is trained to classify weather conditions (Liu et al., 2017). Furthermore, a CNN that performs binary classification was trained to classify images as either cloudy or sunny (Elhoseiny et al., 2015; Lu et al., 2017). Though, this model remains limited to the given binary classes of weather overlooking the complexity of the addressed issue. Based on the previous methods, (Guerra et al., 2018) proposed a framework relying on super-pixel masks. CNN and SVM classifiers to recognise three weather classes, rain, fog, and snow. Whereas this model shows promising in recognising more weather classes, as it only views weather conditions as a single class, not considering the simultaneous occurrence of two or more classes in an image. To this end, to tackle the combination issues of the presence of many class of weather, (Zhao et al., 2019) implemented a CNN based-model that includes an attention-wise-layer to enable the model to infer more than a class for a given time depending on the characteristics of the input image. However, this model shows multiple weather conditions and the presence of other weather conditions(sunny, cloudy, foggy, rainy, and snowy), yet it overlooks the dynamics of visual conditions as regards time of the day that could influence weather recognition accuracy.

It is important to begin by examining the main properties characterising weather images, which makes it recognizable in images. These graphical effects produced by images seem complicated, the appearance is influenced by numerous dynamics, for example camera type, and the settings of the camera. In the image processing field, there are few works addressing weather recognition from images. Image processing techniques were used by (Sudheer et al., 2000), according to their work, they measured drop sizes from an irrigation spray system, Images were captured by high-speed cameras. A rainy condition being dynamic, is categorised in a group of randomly dispersed drops of water, of various shapes and sizes with high velocities (Garg et al., 2004) the visibility of rain from images is determined by the droplet dispersal and speeds, the brightness of the

environment and background scenes are the camera's essential factors, such as coverage time and field depth, which rises when the droplet is bigger in size and reduces with the illumination of the background scenes (Garg & Nayar, 2005). (Yan et al., 2009), employed another type of classifier to recognise images captured by the vision system in automobiles. They based their approach using a robust algorithm frequently used in pattern recognition called real AdaBoost, which distinguished between sunny, rainy, and cloudy weather conditions. Their dataset contained two thousand-five hundred images captured from video recorder in car systems showing cities, streets, or highways. The features examined were, saturation, brightness, hue, and the amplitude of gradient. In the domain of image processing there are various works that addressed rain recognition, (Bossu et al., 2011), the authors here, collected images by video cameras, in their work they recognised the existence of rain or snow in the images. According to the authors, they used a mixture model to isolate the foreground from the background for them to recognise active conditions such as rain from the foreground, potential drops of rain were also selected.

(Chen et al., 2012) used support vector machine to categorise image features in overcast, sunny and cloudy. Their work involved the pre-processing phase which preceded the real classification. In which, the sky region was removed at the beginning of an image input, according to the work done by (Krizhevsky et al., 2012), two-classifier were used to recognise images from the sun and the cloud, earlier trained on the ImageNet dataset. The image processing method was used by (Sawant et al., 2013), to measure rainfall and the raindrop size. The authors employed the cameras for taking high-quality images which they applied several data conversion. The RGB images were converted to gray ones where threshold technique was applied to isolate the foreground from the background, they obtained two images. The revolution was targeted at envisaging drops of rain on the foreground, to recognise and compute its size.

Few applications of deep learning have been found in the weather recognition field. A popular deep learning algorithm like convolutional neural networks (CNNs), was used by (Wang et al., 2016) to envisage three-dimensional maps of snow concentration in satellite images. This approach was also employed by (Liu et al, 2016, Mahesh et al, 2018 & Kunkel et al, 2018), who employed convolutional neural networks to discriminate features

such as synoptic fronts, atmospheric rivers, and tropical cyclones. (Liu et al., 2014) presented a model based on a deep learning, the authors applied a deep neural network (DNN) in their work to process huge datasets of environmental records of almost 30 years. Lately, the LSTM and deep neural network was developed for precipitation now casting and implemented by (Shi et al., 2015). They trained it on the 2-D radar map time series, on various assessment metrics, their system outperformed the current state-of-the-art precipitation now casting.

A multitask deep fully connected neural network was employed by (Iglesias et al., 2015) for forecasting temperature based on past data. The authors proved that neural network method is more improved than linear and logistic regression and could possibly improve the performance of forecasting extreme heat waves. These studies showed that neural networks are a generative method and can be applied on various weather problems. A GoogleNet architecture was used to recognise four classes of weather such as sunny, rainstorm (Szegedy et al., 2015).

In another research done by (Krizhevsky et al., 2012), the researchers used convolutional neural networks for classifying images, with the help of the computing power required for training and general performance. The outcome of the work showed the capabilities of the classifier through an in-built feature extraction based on supervised learning, as an alternative to the manual feature extraction used in the other applications of machine learning.

(Mohamed Elhoseiny, 2015) developed another network based on Krizhevsky's work by using convolutional neural networks (CNNs). The researcher classified images between two likely groups. The two-class weather work done by Mohamed was further extended by (Lu et al., 2017), by adding convolutional neural network features used by (Elhoseiny). Also, (Zhu et al., 2017), explored the GoogleNet architecture implementation which has proven successful in image task classification, proposed by (Szegedy et al., 2015), the application of the CNNs had extra layers than its equals at the time it was proposed, a better result was achieved in the research done by (Zhu).

Another framework called region selection and concurrency model (RSCM) was recommended by (Di Lin et al., 2017), regional cues were used to estimate the weather

condition. This study will be based on the pipeline used by Elhoseiny's work to solve the weather recognition problem from images, since CNN architecture has a greater advantage in terms of accuracy in the image recognition task. An Artificial neural networks having more than one hidden layer, is called a deep neural network (DNN) as can be seen in the image below.

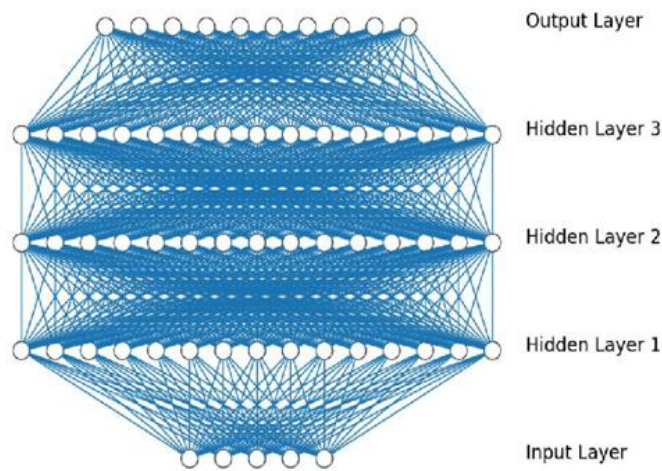


Figure 2.1: Deep neural network (Source: Zixiang Ma, 2019)

2.4.1 THE TAXONOMY OF SEVERAL TYPICAL WEATHER RECOGNITION USING DEEP LEARNING APPROACHES

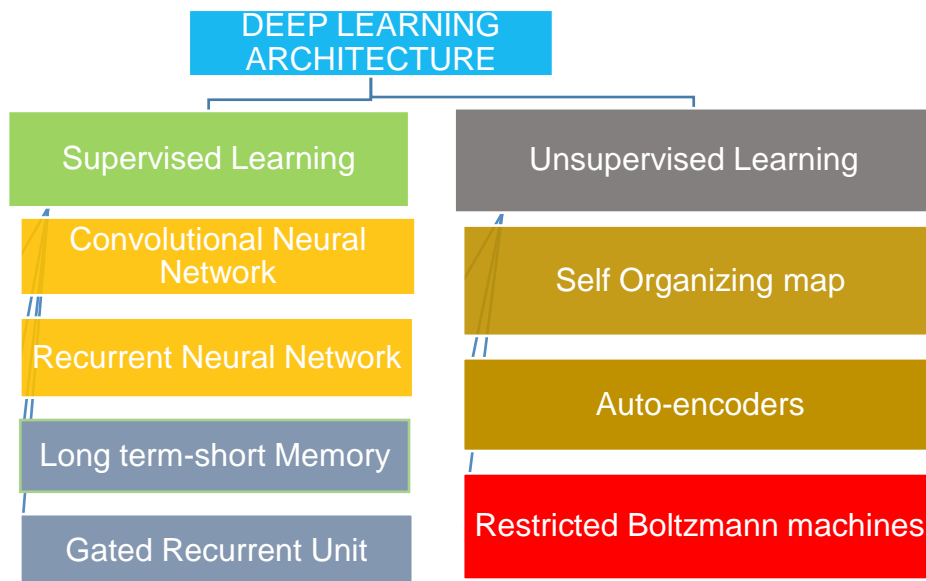


Figure 2.2: the taxonomy of several typical weather recognition approaches

2.4.1.1 SUPERVISED OR DISCRIMINATIVE LEARNING

Supervised learning technique is an approach for creating a machine by using a computer algorithm to train it on input data that has been labelled for a particular output. The training of the model is done until it can detect the essential patterns and relationships between the input data and the output labels, thus, producing accurate labelling results when presented with data it has never seen before. Supervised learning aims to make sense of data to have a background about it. Supervised learning architectures include, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), long term-short memory(LTSM) and Gated recurrent unit (GRU) as seen in Figure 2.2

2.4.1.2 CONVOLUTIONAL NEURAL NETWORKS (CNNs)

CNNs is a family of Deep Neural Networks which are capable of distinguishing and characterising specific points in images and are widely used for analysing real images. Its application areas include image classification, computer vision and image analysis. CNN models consist of three types of layers or building blocks: convolution, pooling, and fully connected layer. The first two layers, convolution, and pooling, performs extraction of features, while a fully connected layer helps in mapping of filtered features or extracted features into the final output. CNN depends mostly on the convolutional layer. CNNs are very suitable for image processing since the features can appear anywhere in the image. A feature extractor is easily used to apply each pixel region of the image. The processing of one layer in a CNN model is passed on to the next layer, which results in more complex features. The training is done using parameters to reduce the differences between the real and predicted outcome by employing output algorithms such as gradient descent. With respect to image processing, using deep learning methods, the convolutional neural network is one of the popular approach among other approaches (Goodfellow et al., 2016). According to (Elhoseiny et al., 2015), convolutional neural networks (CNNs) have been able to classify weather in images as cloudy or sunny weather.

(Kamavisdar et al., 2013). However, the task of weather recognition from images comes with a level of difficulty for an automated system. Since understanding image

characteristics from pixel value is insignificant due to several changes the image could be exposed to.

2.4.1.3 RECURRENT NEURAL NETWORK (RNN)

Recurrent Neural Network (RNN) is another famous neural network, that uses time-series data and put the output from the previous phase as input to the current phase (Dupond S. A,2019). Like CNN, recurrent networks usually learn from the training inputs, and they stand out due to their “MEMORY”, that enables them to effect current input and output through from the previous input. Different from typical DNN, which assumes that input and output does not depend on one another, the output of RNN is dependent on previous elements within a sequence. Though, a typical recurrent network have a vanishing gradient problem, which makes learning from a long data sequence challenging. Long short-term memory (LSTM) is a common form of RNN architecture. An illustration of RNN can be seen in Figure 2.4

2.4.1.4 AUTOENCODERS

Autoencoder is a common unsupervised learning technique in which neural networks are used for learning representations. Usually, autoencoders work very well with high dimensional data, and the reduction in dimension describes how a set of data is represented.

Autoencoder is made up of three parts, which includes, Encoder, code, and decoder. The encoder flattens the input and generates the code, which the decoder then uses to reconstruct the input. Autoencoders are mostly used to learn generative data models. Autoencoders are broadly used in many unsupervised learning tasks, such as, feature extraction and generative modelling.

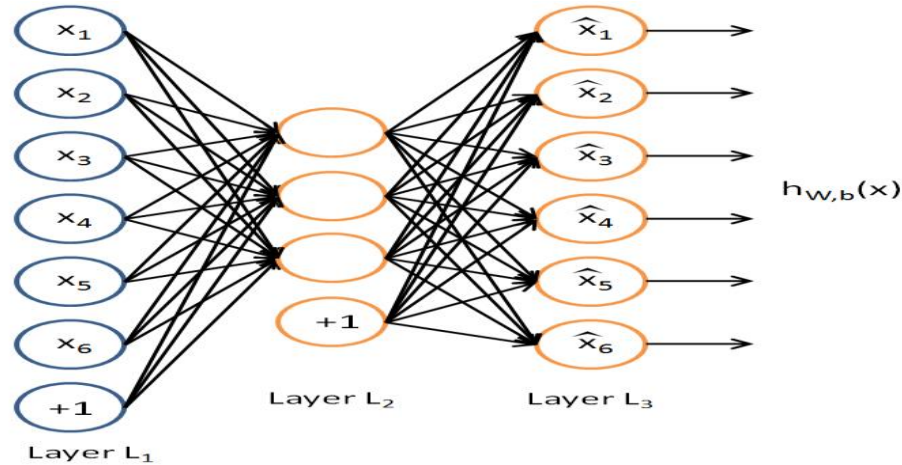


Figure 2.3: A demonstration of deep learning-based, auto encoder algorithm. (Source: Stanford.edu)

An auto-encoder deep learning-based algorithm with three layers as depicted in Figure.1.4. Above, illustrates that Layer L_1 is the input layer, L_2 the hidden, and L_3 the output layer, is used to represent the information x in layer L_1 , so that the output \hat{x} in L_3 can approximate the raw data x . For each layer, an auto-encoder is used to train the features in the layer, then, the layers are combined. Particularly, in each layer of the training process, the vector input must pass via three layers, the hidden layer vectors, L_2 , is taken to be the representations of the input vectors and is used to rebuild the input vectors. The loss function of auto-encoder is shown as.

$$J(W,b) = \left[\frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \| h_{w,b}(x^{(i)}) - x^{(i)} \|^2 \right) \right] \quad (2.1)$$

Here, m is the number of samples for training, the auto-encoder objective is to minimise equation 1.4 to ensure that the output $h_{w,b}(x^{(i)})$ can estimate the raw data $x^{(i)}$ as much as possible.

Table 2.1 Typical weather recognition approaches with their strengths and weaknesses

APPROACH	STRENGTHS	WEAKNESSES
CNN	Effective for image processing high in accuracy rate uses automated feature extraction. handles large datasets	Highly computational Struggles with small dataset.
RNN	Suitable for time series data used with convolutional layers to extend the pixel region	Gradients vanishing problems. Training is a tough task
AUTOEN- CODERS	Earn features automatically from the input data. capable of learning complex and abstract features	Computationally expensive Prone to overfitting

2.5 CNN ARCHITECTURE

The basic parts of a CNN design are as follows:

A convolutional layer which separates and recognises the different points of the image for feature extraction, a pooling layer which is used to reduce the dimension of the input, and a fully connected layer which uses the output from the convolutional layer and recognises the class to which the image belongs. Section 2.4.2 explains the basic parts of a CNN architecture in more detail. Figure 2.3 shows an example of a CNN classification pipeline (Yosinski et al., 2014). When the layers increase in number, the network becomes more specialised in recognising features which is theoretical and interrelated to the actual target task (Clune et al., 2014).

Nowadays, convolutional neural networks have displayed great performance in a range of computer vision tasks, such as image classification (Krizhevsky., 2012) object detection

(Ren et al., 2015), semantic segmentation (Gkioxari et al., 2017), etcetera. Many exceptional architectures of CNNs are proposed as well as AlexNet (Krizhevsky et al., 2012) used a convolutional neural network (CNN) recently for image classification to implement a distinctive architecture of CNNs for image classification. The authors' suggestion pointed out the abilities of an image classifier with an in-built feature extraction based on supervised learning, rather than manually extracting features as previously used in other machine learning applications.

(Simonyan & Zisserman, 2014) employed VGGNet and ResNet (He and Zhang, 2016), which outperformed the conventional methods with a large gap. Motivated by the great success of CNNs, a few works attempted to apply CNNs for weather recognition task. (Elhoseiny et al., 2015) fine-tuned AlexNet directly (Krizhevsky, 2012) in a two-class weather classification dataset image.

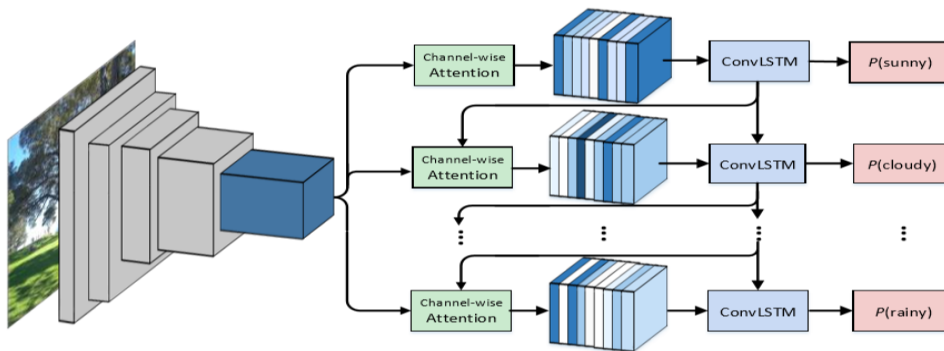


Figure 2.4: The illustration of CNN and RNN approach for multi-label weather recognition. (Source: Bin Zhao, 2019)

Here, CNN was used for extracting the features from the images and the convolutional LSTM was used to predict weather class. At every stage, the channel wise attention model was utilised for recalibration of the feature responses.

In the weather recognition domain, convolutional neural networks were employed by (Elhoseiny et al., 2015) and (Ziqi et al., 2016). They produced a model built on Krizhevsky et al work. For categorising images between two possible classes. (Lu et al., 2017) extended their work of two-class weather recognition by adding CNN features used by Elhoseiny. Likewise, (Zhu et al., 2016) discovered an implementation that has proven successful for the task of image classification. The architecture known as GoogLeNet,

proposed in (Szegedy et al., 2015). Thus, CNNs have additional layers than its contemporaries as at the time of its proposal, it achieved better results in the experiments performed by (Ziqiet al., 2016). (Di Lin et al., 2017) recommended a deep learning framework named region selection and concurrency model (RSCM) which used regional cues to predict the weather condition.

(Lu et al., 2014), then achieved a better result. (Lu et al., 2017) joined the hand-crafted weather features with CNNs extracted features, and further improved the classification performance. Whereas as deliberated in (Lu et al., 2017), there was no sort of limitations in the weather classes. Many weather conditions may appear concurrently. Hence, the above approaches suffered from loss of information when treated as a single class recognition problem. (Li et al., 2017).

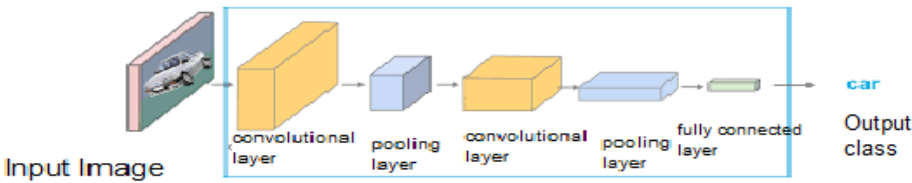


Figure 2.5: The pipeline of CNN Image classification. (Source: Kaggle)

2.5.1 THE CONVOLUTIONAL LAYER

For a convolutional neural network, its main building block is the Convolutional Layer. The layer parameter is a learnable filter, and the three-dimensional matrix in mathematical values, with respect to size. Equation 2.2 shows the output of the i^{th} filter, denoted by y_i^l , for a convolutional layer l with a total number of C filters.

$$y_i^l = \delta \left(\sum_{j=1}^{C^{l-1}} f_{i,j}^l * y_i^{l-1} + b^l \right) \quad (2.2)$$

b^l the bias vector of layer l , $f_{i,j}^l$ the filters of the conv layer l connected to the j^{th} feature map of layer $l - 1$, and δ is the activation function. The parameters $f_{i,j}^l$ refined by the network are updated by back-propagation. That way, the network learns several forms of filter specialised in resolving the task.

2.5.2 THE POOLING LAYER

This layer is next to the convolutional layer, its major function is to reduce the three-dimensional input size. Usually, this layer consists of a filter which slides, through a fixed value of strides, through the input layer to produce the output (Goodfellow et al., 2016). The filter can perform different functions, the common ones include.

- The max pooling: This uses the maximum values of the input within the filter size. It can also return the maximum input within a rectangular area.
- The average Pooling: This takes the average values of the input contained in the filter sizes.

2.5.3 THE FULLY CONNECTED LAYER

Generally, CNN has a minimum of one Fully Connected (FC) layer, usually positioned before the network output, its main purpose is for parameter learning. To assign the input layer to the corresponding output, its output y^l for the FC layer l is presented in the equation below.

$$y^l = s(y^{l-1} * W^l + b^l) \quad (2.3)$$

w^l is the weight, while b^l is the bias vector of layer l , and s is the activation function. Different from the convolutional layers, fully connected layers (FC) does not support parameter sharing. As a result, there is a considerable rise in the parameters to be learnt in a convolutional neural network (CNN).

2.5.4 ACTIVATION FUNCTION

For non-linearity to be introduced into the network, so it could help in learning additional complex functions, activation function is used. Several functions for activation are available, though, the common and general one such as, the ReLU, is seen in Figure 2.6 it is a function which the threshold of the activation is set to zero (0). This is explained in the equation below. While the softmax activation for the classifier is displayed in Figure 2.7.

$$\delta(z) = \max(0, z) \quad (2.4)$$

The Relu activation function can be seen from Figure 2.6. The description is seen from equation 2.5, z is a vector of the input to the output layer, and $j = 1, \dots, k$ indexes the output neurons, with k number of classes. This value associated to each output neuron, equivalent to the probability that the input signal belongs to the class representing it, and this is limited between 0 and 1. Therefore, the sum of all the output value is limited to 1.

$$a(z)_j = \frac{e^{z_j}}{\sum_{k=1}^k e^{z_k}} \quad (2.5)$$

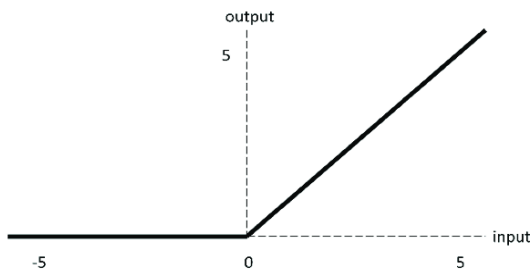


Figure 2.6: The activation Function of ReLu

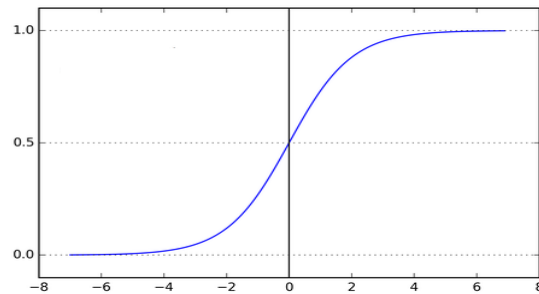


Figure 2.7: The Softmax activation Function(Source: Hossam Sultan, 2019)

2.6 TECHNIQUES FOR REDUCING OVERFITTING

For networks that contain many hidden layers such as DNNs, Figure 2.8 shows an example of deep neural network overfitting is a huge problem. Overfitting occurs when the model fits too well as compared to the training dataset. Thus, it becomes difficult for the model to generalise to new samples that were not in the training dataset. Example of an overfitting network is demonstrated in Figure 2.9

To deal with this problem, various methods have been proposed such as regularization, which is the method that presents the loss function penalty, regularization helps to limit the growth of attaining a steadier model.

L2 regularization: This is the most generally used regularization method. It adds squared mean. This implementation can be seen in equation 2.6.

$$L = L + \lambda * \theta^2 \quad (2.6)$$

Here, θ^2 , the L2 mean of the parameter, and λ as the regularization influence on the input that decides the input of term W^2 in the cost function.

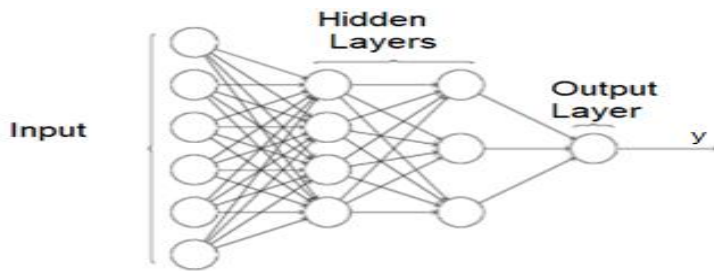


Figure 2.8: A deep neural network (Source: Tena Belinic, 2018)

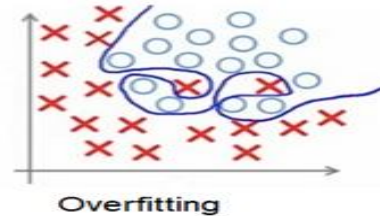


Figure 2.9: Example of overfitting

- ❖ **Dropout:** At every iteration, it randomly selects some nodes and remove them with all their incoming and outgoing connections as seen in the diagram below. Since, the unit and the connections are dropped randomly, given the possibility of model, while processing to inhibit it from adjusting so much. Hence encourages every unseen unit to produce good features with no dependence on others to amend its errors (Srivastava et al., 2014).

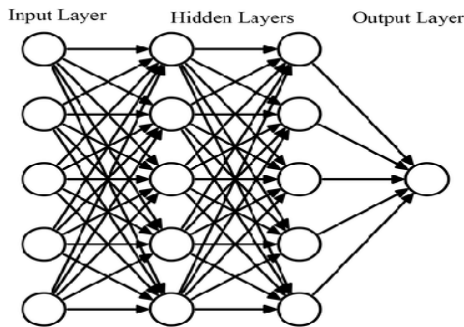


Figure 2.10: A neural network before dropout (Source: Zong-Sheng Wang, 2020)

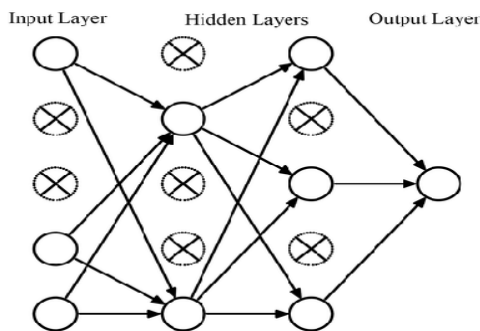


Figure 2.11: ANN after applying dropout.

- ❖ **Batch Normalization:** This method regularizes the model and reduces the necessity in dropout (Srivastava et al., 2014). To function, there is need for the mini-batch size. Thus, inappropriate for Stochastic Gradient Descent algorithm. Equation 2.7 describes the mathematical expression of batch normalization approach.

$$\mu_B = \frac{1}{N} * \sum_{i=1}^N x \tag{2.7}$$

- ❖ **Data Augmentation:** It is the easiest method to reduce overfitting. This method is used for increasing the size of the training dataset by transforming the images such as, rotating, flipping, scaling, and shifting. All transformations done using this technique is known as data augmentation. Which generally provides a big increase for improving the accuracy of the model.
- ❖ **Early stopping:** training a neural network is usually faced with the problem of number of epochs train. The use of large epochs possibly leads to overfitting of the training dataset, while very little epochs could produce an underfit model. Early stopping is a technique that enables you to indicate a randomly huge number of training epochs Then stops the training as soon as the model performance stops to improve on the validation set

2.7 THE DIFFERENT VARIANTS OF CNN APPROACHES

Convolutional neural networks (CNN) are now widely used in resolving computer vision glitches at present. Since it emerged as the frontrunner architecture of the previous year's ImageNet Large Scale Visual Recognition Challenges (ILSVRCs), (Deng et al., 2009). The challenge involves evaluation of processes to detect objects, and image classification including the recognition of a thousand classes of image. Firstly, Alex-net won the CNN ILSVRCs competition. Hence, making it more widespread architecture (Krizhevsky et al., 2012) in 2012.

2.7.1 LENET

LeNet is possibly the least complicated engineering model. Which consist of two convolutional layers and three fully connected layers . The additional sampling layer is the average pooling layer which trains loads as weights. The model uses about 60,000 parameters for training. This model has developed into the typical method of stacking a convolutional layer with activation function, and the pooling layer is implemented by finalising it with at least one fully connected layer.

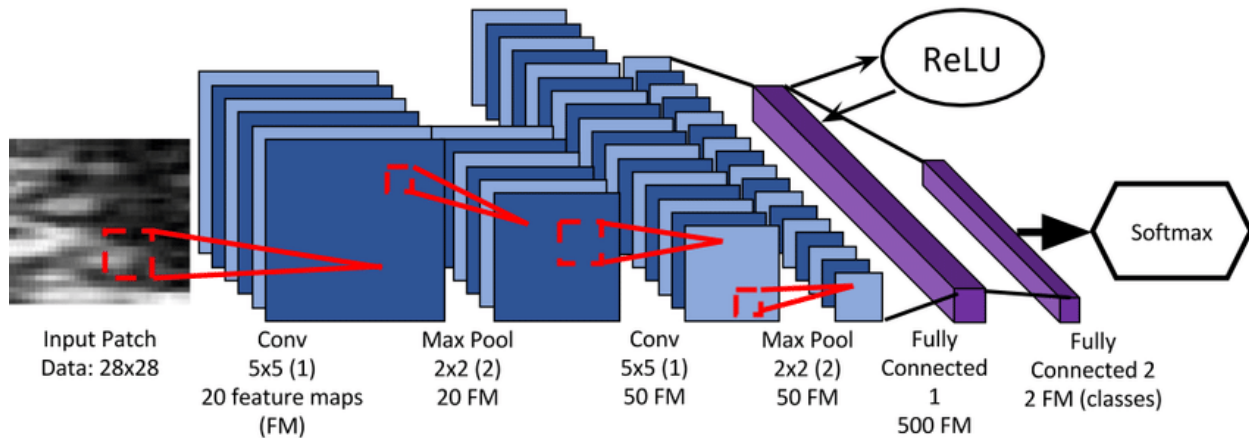


Figure 2.12: LeNet architecture (Source: Gerard Pons, 2017)

2.7.2 ALEXNET

The AlexNet architecture is made of a total of eight layers, with five convolutional layers and three fully connected layers. A few more layers are stacked on the LeNet model to form AlexNet

All the layers including the dropouts' ones are displayed in Figure 2.10, uses ReLU as activation function. Also, it uses SoftMax to classify images. In CNN structure, the GPU solves the calculation problem. AlexNet by (Krizhevsky et al., 2012), has become the building block of utilizing DNN approach. Fig. 2.13 shows AlexNet architecture. Thus, obtaining an error rate of 15.6%, this error rate is needed for classifying any image within the closest five classes (top 5).

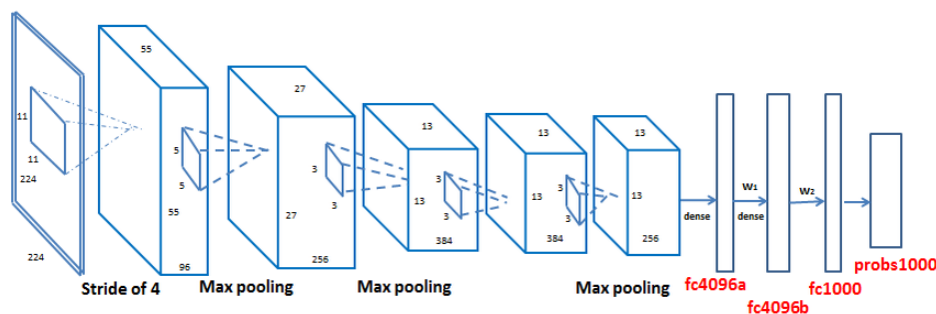


Figure 2.13: Alex-net architecture. (Source: Huafeng Wang, 2015)

In the subsequent years. AlexNet was enhanced by the authors of (Zeiler & Fergus, 2014) changing the factors, and achieved a 11.2% top-5 error rate in 2014,

2.7.3 VGG-16

The model VGG-16 was developed by Visual Geometry Group (VGG) it is the traditionally used convolutional neural network design. Which relies on the study of how to build a perceptive model. VGG-16 uses a small 3 x 3 channels. The model is made up of thirteen convolutional and three fully connected layers and uses the ReLu function like AlexNet . Additional layers are stacked on AlexNet to form the VGG model. It contains about 138 million parameters and uses storage space of about 500MB . The developers also intend to build a deeper variation, VGG-19.

VGGNet (Simonyan & Zisserman, 2014) although, this did not lead to winning the competition, but indicated that, it can decrease the number of factors, needed to increase the depth of the network at the same time. Hence, it attained an excellent performance more than the Alex-net architecture by 7.3%. The VGGNet architecture contained more convolutional layers than AlexNet, it has exactly thirteen convolutional layers, which were small as regards filter dimensions, hence, causes a decrease in parameters, and however, they were capable of learning new high-level features than earlier CNNs. A different approach that was important, won as the champion of the ImageNet challenge 2014 with an error rate of 6.7%, which is GoogleNet (Szegedy et al., 2015).

2.7.4 GOOGLNET

After VGG-16 was developed, Google gave rise to GoogleNet, which was another champion of ILSVRC-2014 with a higher accuracy rate than its predecessors. Different from the previous networks, GoogleNet has a different architecture. Firstly, other networks such as VGG-16 have convolutional layers stacked one over the other but, GoogleNet organised the convolutional and pooling layers in a similar way to extract features using different sizes of kernel. GoogleNet is famous for its use of the Inception module, which consists of multiple similar convolutional layers with different filter sizes, and a pooling layer. The design allows the network to learn features at multiple scales and resolutions, while keeping the computational cost manageable. GoogleNet was built based on the ideas of previous convolutional neural networks, such as LeNet, which was one of the first

successful applications of deep learning in computer vision. However, GoogleNet is much deeper and more complex than LeNet

The aim was to increase the depth of the network and to achieve a higher performance rate as compared to previous winners of the ImageNet classification challenge.

2.7.5 RESNET-50

Based on previous CNN, the Resnet-50 was faced with the problem of gradually expanding layers and thus, led to better performances. Though, with the expansion of the network, accuracy became saturated and later on reduced rapidly. Microsoft Researchers tried to fix the issue with ResNet-50 by using deep residual learning framework, by using residual blocks and identity blocks . The idea is to add a shot cut or a skip connection by allowing a swift flow of information from one layer to another.

2.8 CHAPTER CONCLUSION

So far, the chapter explained the related details about the traditional method of weather recognition, and the machine learning models was also presented. Then various taxonomy of deep learning together with their strengths and weaknesses models were explored. Various work reviewed were very useful for an enriched understanding of the challenges the target task involves and helped in making an informed decision about the approach suitable for the task at hand. Based on the literature reviewed on CNN, they have shown to be very successful in image recognition particularly, deeper architectures such as Resnet.

CHAPTER 3

RESEARCH DESIGN AND METHODOLOGY

3.1 CHAPTER OVERVIEW

This section covers the details of the dataset used such as the data collection and design the details about the preprocessed and augmented images are also presented. To implement augmentation in the training phase, the Augmentor library is used, augmentation is done on the four classes of images for every batch. An introduction to ResNet and the hyperparameters for training and optimisation methods are also given in detail.

3.2 DATA DESIGN

To achieve the aim of training a network which can recognise different weather conditions, thus, it is very necessary to make available the images for training, which should meet specific requirements as follows.

- ❖ **Different Weather Conditions:** In every region, the weather changes. It is essential that the images contain various weather phenomena noticeable at various times. This will ensure that the network learns the difference in the various weather images.
- ❖ **High-Quality Images:** High-Quality Images: It involves high-resolution images that have good clarity as well as detail. This is because low-quality images might introduce noise hence complicating learning for the network.
- ❖ **Variety of Scenes:** The dataset comprises of variety of images taken on various scenic including cities, countryside and hillsides thus allowing the network to easily adjust to different environments
- ❖ **Annotated Images:** The images must be labeled according to the different weather conditions . This annotated data is very important in supervised learning where the network is trained on examples that are already marked.
- ❖ **Data Augmentation:** Data augmentation techniques that has been employed include the following; rotating, scaling, flipping; adding noise etc. to increase the generated training data size hence increasing network capacity for learning.

According to these criteria, a solid dataset has been prepared which will allow effective training and recognition of various weather patterns present in the images.

3.2.1 DATA COLLECTION

A dataset of small amount of weather images from various scenes named “Multi-Weather Dataset2” is used in weather recognition in this study. Table 3.1 shows the details of this dataset. The data set comprises of 5000 images in total. The images have totally different backgrounds and cover majority of the adverse weather grouped into four classes: Rainy days, sunny days, foggy days, and cloudy days. The images in each class are grouped as the training and validation set, using four (4,000) thousand image data set for training and (1,000) thousand for validation and testing. The 80/20 rule principle was applied by allocating 80% of the data for training and 20% for validation/testing.

The dataset is distributed in the following format ;

- ❖ Training set: 80%
- ❖ Validation set: 10%
- ❖ Testing set: 10%

Table 3.1 details of multi-weather dataset2

CATEGORY	TRAINING	TESTING
Sunny	1500	270
Rainy	1000	250
Cloudy	1000	249
Foggy	1500	231
Total	4000	1000

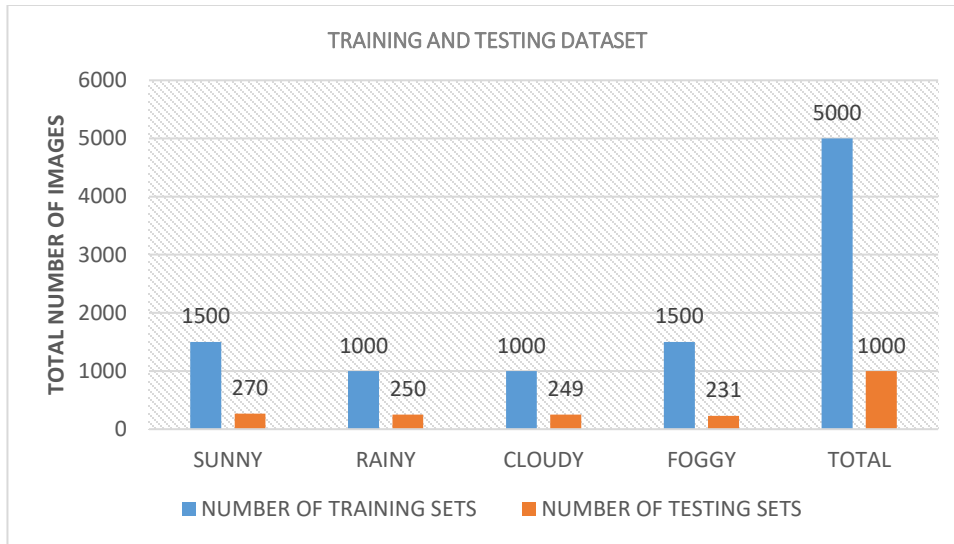


Figure 3.1: images per class

From the image data, the quantity of training images is more than the testing set with a huge amount. Most of the training images make up about 80% of the data. In figure 3.1 some examples of the training images are higher than the testing one as shown figure 3.1 above.

Deep learning methods typically need huge amounts of datasets to prevent overfitting of the network model. As a result of this condition, data augmentation method of some sort was used for the image training in “dataset2” in this study. To begin with, the image block with size (224x224) intercepted unevenly from the original image. Also, the image is cropped, rotated, flipped, translated, and enlarged. Lastly, different kinds of images are generated, as shown in Figure 3.2. Particularly, Image Data Generator function embedded in the Keras API generated the weather images, followed by stochastic conversion for each of the training epochs, parameters such as rotation range, width shift, and shear range were set. This method is adopted for the model not to have two matching images, since it is important for the prevention of overfitting, therefore, enhancing the generalisation, as well as the robustness of the network. To validate the effectiveness of hyperparameter optimisation for the weather recognition task, a comparison experimentation without hyperparameter optimisation or with hyperparameter optimisation is demonstrated in chapter 5.

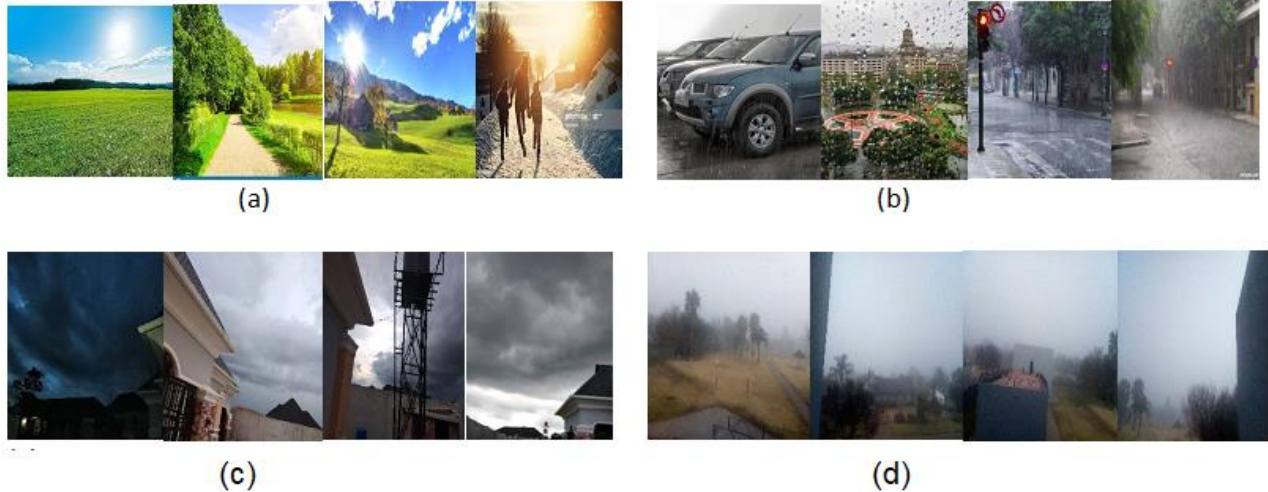


Figure 3.2: The four samples of the Dataset. (a) Sunny. (b) Rainy. (c) Cloudy. (d) Foggy.

3.2.2 IMAGE PREPROCESSING

In this section, the preprocessing techniques used for the input images are described. The images used for the training phase were scaled, primarily the aspect ratio of the images were distorted and randomly cropped, which covers between 10%-90% of the misleading image, was taken. The cropped edges of the image was rescaled to size 224 and the Color jitter was applied, such as random brightness, contrast and saturation of the images. Then, Color normalization was applied by deducting from each image the mean dimension, then calculated all through the entire set. Lastly, the images are horizontally flipped on 80% of the images chosen randomly.

3.2.3 IMAGE SCALING

The main limitation in building the model in this work was the necessity to resize the different images in the dataset to an equal dimension. The dimensions of the images was set to (256 x 256 pixels) therefore, the aspect ratio of the images was maintained, (in width and height). This research used the Augmentor library in keras python for the pre-processing of the image as explained in section 3.1.1. The Augmentor library was used for analysis the images in the dataset directories, by resizing and saving to the required dimension, in a new directory to form the new dataset. Normally, pre-processing the

image is important to speeding up the training process, thus, improving image features by eliminating undesirable distortions.

data augmentation technique is effectively used by adding new data points to the original image which is transformed by cropping and flipping the original image, as depicted in Figure 3.3.

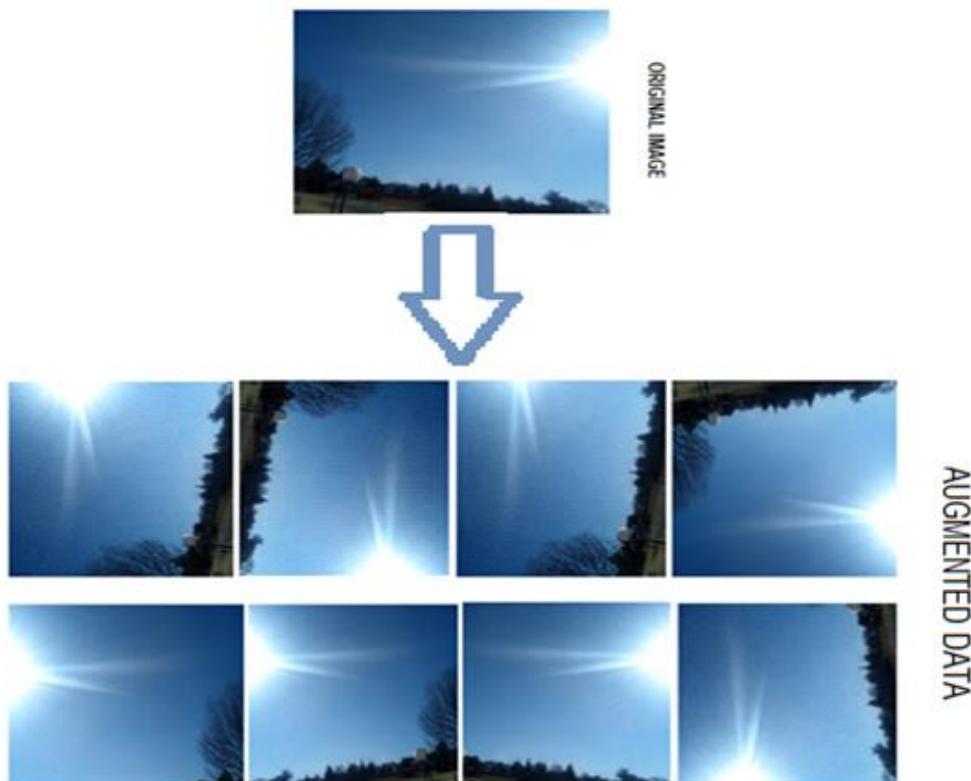


Figure 3.3: Data augmentation result.

3.3 RESIDUAL NETWORKS

In all the reviewed CNN architectures in Section 2.5, ResNet has shown to be the best in performance in the domain of image classification. The results of ResNet is attributed to the manner it is organised, which enabled it to attain a deep architecture of a lesser number of parameters (Zhang et al., 2015). ResNet uses the concept of residual learning (Ren et al., 2015). These novelists suggested a deep residual learning background whose method is driven by the idea of the VGGNet (Simonyan & Zisserman, 2014).The

innovation in their model was the additional shortcut connections of each layer, depicted in Figure 3.4. The architecture allowed the layers to learn a residual mapping that is, properly representing the desired fundamental mapping between few fixed layers as seen in Figure 3.4.

Here, $H(x)$, allows the fixed layers to fit into a different portion, $F(x) := H(x) - x$, this is written as $H(x) := F(x) + x$. The latter formula can be realized by using shortcut connections to skip some other layers.

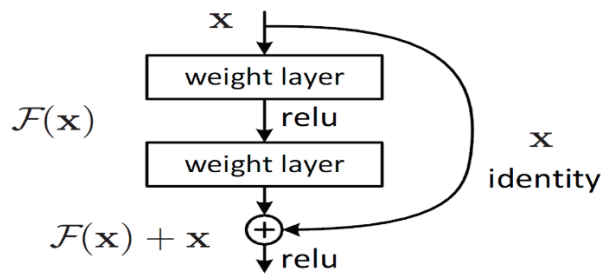


Figure 3.4: A residual Unit (Source: Connor Shorten, 2019)

The main idea of ResNet (Zhang et al., 2015) being the residual block, which addresses the vanishing gradient problem by introducing skip connections or shortcuts, allowing the network to learn residual mappings instead of directly learning the desired mappings. which is shown in Fig. 3.4 is a technique used to combine input and output by learning the difference between input and output. The middle layer can easily be optimised for deep neural networks, and the accuracy is improved due to the increased network depth.

3.3.1 RESIDUAL NETWORK ARCHITECTURE

The Resnet architecture has different units, known as residual units. Made up of blocks, which are simulated so many times, through the entire network. The depth of the architecture depends on the amount of reiteration of the blocks within a Resnet unit. Figure 3.5 depicts the basic architecture, the main foundation comprising of the Resnet units can be seen. They are composed of series of convolutional layers, typically made up of $(3) \times (3)$ filter. Resnet has additional layers, including, the Pooling layer, placed from the start of the design, the average layer is positioned lastly. Both are used for

sharing the size of their input while the fully connected layer (FC) and the Softmax function is used for classification.

This research has adopted the simplified model, ResNet-15 based on residual network 50 (Resnet 50), inspired as per the little number of factors for training in comparison with the other variations. In Table 3.2 the structure of the architecture ResNet-15 is described. For this architecture, each ResNet unit is made up of two blocks. The blocks perform Identity mapping through their shortcut referred to as "Identity", while the one performing the convolution task is called "Studyion".

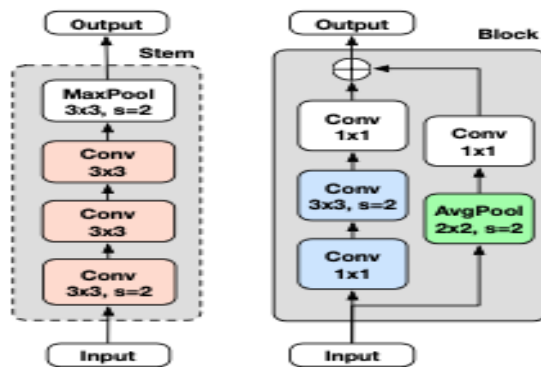


Figure 3.5: A basic Resnet Representation. (Source: Akihiro FUJII, 2020)

Table 3.2 The Resnet structure

MODULES	STRUCTURE	OUTPUT DIMENSION
1	Identity Identity	56x56x64
2	Identity Studyion	28x28x128
3	Identity Studyion	14x14x256
4	Identity Studyion	7X7X512

The designs are improved further in the subsequent work, (He et al., 2016). In which the authors only aimed for enhancing the performance, also understanding better the conduct of the network when trivial changes must be made.



(a) Original

(b) Full pre-activation

Figure 3.6: Full Resnet pre-activation. (Source: JalFaizy Shaikh, 2017)

The main differences between ResNet – 50 and ResNet – 15 are as follows.

Table 3.3 The differences between ResNet-50 and ResNet-15

RENET-50	RESNET-15
Adds the second non-linearity after the addition operation is done between the x and $F(x)$. as seen in figure 3.4	Removes the last non-linearity, hence, paves the path of the input to output by using identity connection.
Uses Batch Normalization and ReLU activation for the input before multiplying with the weight matrix .as seen in figure 3.6	Performs the convolution together with Batch Normalization and ReLU activation
The addition operation output goes from ReLU activation, and it is then transferred to the next block as the input as seen in figure 3.6	Uses the second non-linearity as an identity mapping by performing the addition output operation between the identity mapping and the residual mapping, then passes it to the next block for further processing

3.4 RESNET METHOD

Here the first stage is to decide which CNN approach to use. A ResNet model is an enhanced version of CNN that minimizes distortion that happens in deeper and more complex networks by adding shortcuts between layers. bottleneck blocks are also utilized allow a faster training of ResNet. Because of these factors. This research has decided to choose optimising ResNet to improve its accuracy. ResNet-50 has a conv layer size of (7×7) , a max pooling layer size of (3×3) , and series of residual modules. The residual modules has two simple modes, which are shown in Figure 3.6 (a) whose input and output are of the same dimension, for them to be connected. Figure 3.6(b), where the ConvBlock input and output are of different dimensions, for this reason, they cannot be joined in together, also, the purpose was to alter the dimension of the feature vector by the convolutional layer of size (1×1) . Lastly, the classification and recognition of the weather from images are done through the fully connected layer and the Softmax classifier.

Built based on ResNet-50, ResNet-15 is simplified and enhanced version. First, the size of the conv layer (7×7) and the max pooling layer with size (3×3) in ResNet-50 is reserved. Lastly, four Convolutional blocks (CB) of the first level in four group of residual module is reserved too, also, the stride parameter of the first group of the residual module changes from 1 to 2, while the other identity block (IB) are removed. At this point, the average pooling layer changes into a fully connected layer with 512 dimensions, while the dropout layer is then added after the fully connected layer. Lastly, the Softmax classifier remains unchanged. ResNet-15 architecture is shown in figure 3.5 the rectangular box and curved arrow in the figure represents the four groups of residual modules. Considering that, the number of parameters of the network model should be equivalent to the size of the dataset, in this study, the number of convolutional kernel is properly adjusted to reduce the number of parameters of the network model.

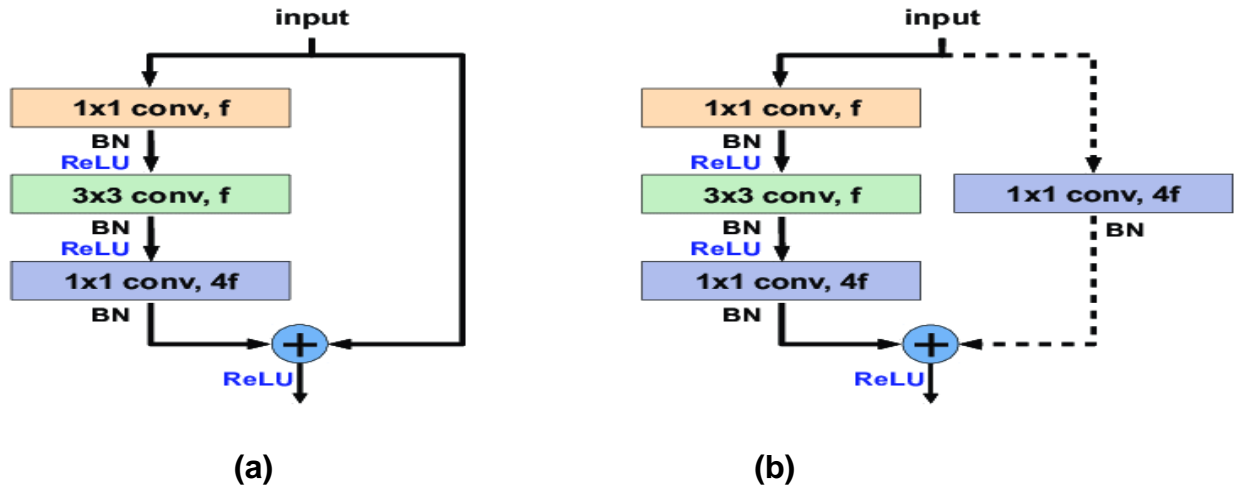


Figure 3.7: The two simple modes of residual module. (a) Identity block. (b) Convolutional block. (Source: Leandro Aparecido Passos Júnior, 2018)

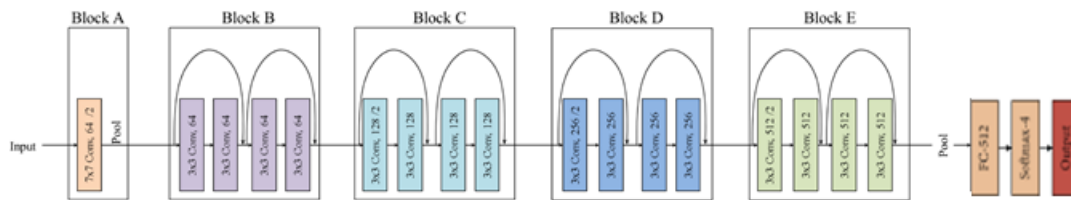


Figure 3.8: ResNet-15 Architecture. (Source: Sai Kumar Basaveswara, 2019)

Below are the major benefits of ResNet over the earlier winners of ILSVRC, such as AlexNet and VGGNet, as measured in terms of the following.

- 1.) **ACCURACY:** Due to its accuracy, it obtained the best performance ever achieved in ILSVRC.
- 2.) **SPEED:** training speed is highly accelerated with Resnet.
- 3.) **DEPTH OF NETWORK:** Increases the network depth without additional parameters. Thus, enables the model to learn more complex and abstract features, without necessarily increasing the training dataset size respectively.
- 4.) **VANISHING POINT:** It reduces the influence caused by the vanishing gradient problem, which usually leads to saturation and the reduction in the accuracy, whenever the network increases in depth.

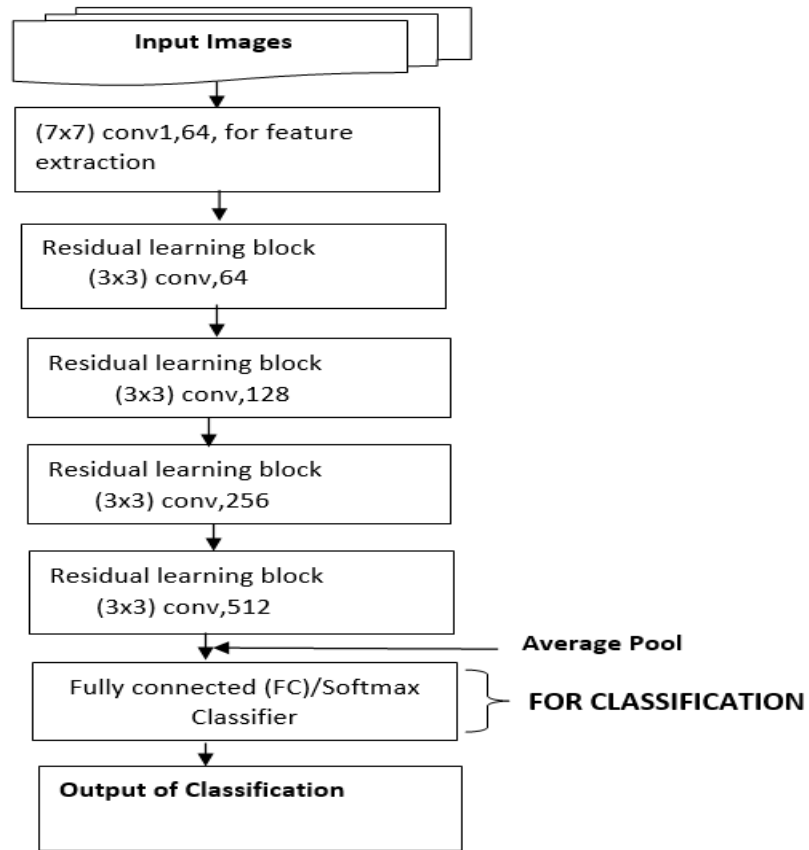


Figure 3.9: The flowchart of the proposed Resnet-15 method

3.5 HYPERPARAMETER OPTIMISATION

Hyperparameter optimisation is the process of finding the best combination of hyperparameters for a deep learning model to achieve optimal performance. Hyperparameters are parameters that are set before the learning process begins and influences the behavior and performance of the model.

The ResNet-15 configurations were optimised in this research, by fine-tuning their hyperparameters to obtain the best architecture for classifying the dataset. Choosing the appropriate model hyperparameter values improves the accuracy of the network model. Instead of using the complete training set, this research used small batches of training data to compute the gradient and update the weight matrix in Stochastic Gradient Descent (SGD), which is a straightforward modification to the

standard gradient descent algorithm. This change makes update noisier, but it also enables the gradient to move one step further for each batch as opposed to one step for each epoch—which eventually speeds up convergence without compromising loss or recognition accuracy. When it comes to training deep neural networks, SGD is possibly the most significant optimiser for hyperparameter optimisation . SGD is also easy to implement.

ResNet-50, AlexNet, ResNet-18, GoogleNet, VGG16 and ResNet-15 were evaluated to determine the accuracy using the selected dataset. Next, the ResNet model was optimised using the random search optimisation method by using keras tuner library, with 40 epochs for the model. The next step involved training the ResNet model to optimise the following hyperparameters.

- ❖ Number of dense layers: This is the number of layers in the fully connected layer.
- ❖ Dropout rate: This is a regularization technique that prevents the network from over-fitting, during training, some neurons in the hidden layer are dropped at random.
- ❖ Momentum: Used to speed up gradient descent algorithm, to achieve quicker convergence.
- ❖ Learning rate: A crucial hyperparameter that controls the rate of the step in each iteration. It will take a while to converge if the learning rate is too low, and it may diverge if it is too high.
- ❖ Batch size: the number of images processed at once, reduction in speed occurs when the mini-batch size is too large, and slow convergence occurs when it is too small.

3.5.1 HYPERPARAMETER SETUP

This research utilised a Resnet architecture implemented by keras library with python, and the hyperparameters and the ranges used for training the architecture are shown below.

The configurations of hyperparameters ranges used for the optimisers are displayed in table 3.4

Table 3.4: Hyperparameter configuration range

Hyperparameters	Ranges
Number of dense layers	Min:32,max:512,step:32
Batch size	16, 32, 64
Learning rate	0.01, 0.001, 0.0001
momentum	0.1, 0.9
Dropout	0.3-0.9

3.6 CHAPTER CONCLUSION

This chapter emphasizes the steps taken for the progress of the research experiment. The programming language suitable for this research work is explained and the python libraries and framework used is described. The data collection that meets some specific requirements was covered.

The flowchart diagram of the methodology which shows the general flow for each step taken during the experiment was presented. The image preprocessing to help reduce inconsistencies in the dataset was presented, which helped to reduce computational time of the experiment. Finally, the training method and the hyperparameter ranges were obtained.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 CHAPTER OVERVIEW

This section describes in detail the configuration and setup of experimental tools used, and the choice of the framework for the proposed system and its related information. Python programming language is the best choice for this research work due to its flexibility, it is extendable and has a huge number of libraries. Section 4.2.1 explains the python libraries used .

4.2 CONFIGURATION AND SETUP OF EXPERIMENTAL TOOLS

4.2.1 PYTHON LIBRARIES

- 1) **Augmentor library:** To implement augmentation in the training phase, the Augmentor library is used, augmentation is done on the four classes of images for every batch. Images produced by augmentations are removed automatically by cropping, flipping, rotation, random erase and resizing the image.
- 2) **NumPy:** Used for scientific computing, it worked effectively on multidimensional data arrays and matrices.
- 3) **Matplotlib:** the matplotlib's pyplot API is used for creating the charts and graphs
- 4) **Keras tuner:** A library called Keras Tuner was used to fine-tune the deep learning neural network hyperparameters, which helped in selecting the best values for the models' implementation in keras TensorFlow.

4.2.2 SYSTEM CONFIGURATION

- ❖ All experiments were done using the following system configurations.
- ❖ Dell Core i5
- ❖ 2.4GHz CPU
- ❖ 8GB memory (RAM)
- ❖ Hard-disk size: 500GB

4.3 FEATURE EXTRACTION AND SELECTION

Choosing the appropriate features or point of interest from image related task is crucial for any effective computer vision problem. Machine learning becomes more accurate and efficient with the help of feature extraction. In this study, the feature extraction and selection was done through image preprocessing techniques to reduce unwanted data, enhance learning rate and improve model accuracy. The preprocessing technique was used to identify the features in the images, such as shape, or edges, then resize the images to 224x224, and color jitter such as erratic contrast, brightness, and saturation was added. Then, each image's mean dimension was subtracted to apply color normalization.

4.4 MODEL DESIGN AND SELECTION

The design of the model was implemented by keras library with python, and the parameters used for training the architecture are shown below.

- ❖ Regularization method: L2 regularization

L2 regularization was applied during the training phase of the network, which helped to generalise the model better by reducing the complexity of the learned parameters, thus improving the model's ability to generalise to unseen data.

- ❖ Loss function: Cross-entropy

Cross-entropy loss was used for training the classification models. Which is more effective because it was combined with Softmax activation in the output layer for the multi-class classification task.

- ❖ Method of optimisation: Random search and hyperband

Random search and hyperband algorithm optimisation methods were used to search through a large hyperparameter space, including learning rates, batch sizes, momentum, dropout number of dense layers. Random search optimisation method was used to explore the space randomly, while the hyperband algorithm combines random search with adaptive resource allocation to focus on the best configurations.

❖ Activation: Relu and sigmoid

ReLU (Rectified Linear Unit) is a popular activation function which was used in the hidden layers of the deep neural network. It helped to mitigate the vanishing gradient problem and accelerates convergence during training.

Sigmoid activation was used in the output layer for the classification task.

❖ Optimiser: SGD and Adam

Stochastic Gradient Descent (SGD) is a fundamental optimisation algorithm that was used to update the parameters based on the gradients of the loss with respect to those parameters.

Adam (Adaptive Moment Estimation) is an adaptive optimisation algorithm which was combined with momentum methods, to ensure robustness and effectiveness of the network.

The initial learning rate (lr) is 0.01 and updated during every epoch. The training momentum is 0.9, and the weight decay set to 0.0001 and the batch size is fixed at 64, then the network training was done for 40 epochs. For the training of the network model, image input samples with dimensions were used.

Model selection was done on the 5000 images for the four weather classes, rainy, sunny, foggy, and cloudy. Using a cross validation procedure, (80%) is used as the training dataset and (20%) is used as the Test set. During the testing phase, the network is tested on unseen data. The learning curve is plotted, which is made up of the network accuracy over the reiterations of the training and the testing phase, the loss value of the validation over the iterations to check its tendency was also used for the model selection.

To validate the approach used in this study, the ResNet-15 is built based on python. ResNet-15 is mainly trained on the training set of "DATASET-2". The validation set of images are fed into the training model for recognition purposes, then, the classification result is displayed into one of the four weather conditions, such as rainy, sunny, foggy, and cloudy. Lastly, in line with the recognition results of the images, the accuracy of recognition is computed, and the confusion matrix of weather recognition is established, as can be seen in Figure 4.1(f). The recognition accuracy of each class is represented by

the values on the slope of the confusion matrix respectively. As depicted in Figure 4.1(f), the recognition accuracy of the weather conditions is 97.89% rainy, 94.61% sunny, 95.73% cloudy, and 96.92% foggy.

Lastly, to know which image category the network performs best, the confusion matrix is calculated. as seen in figure 4.1.

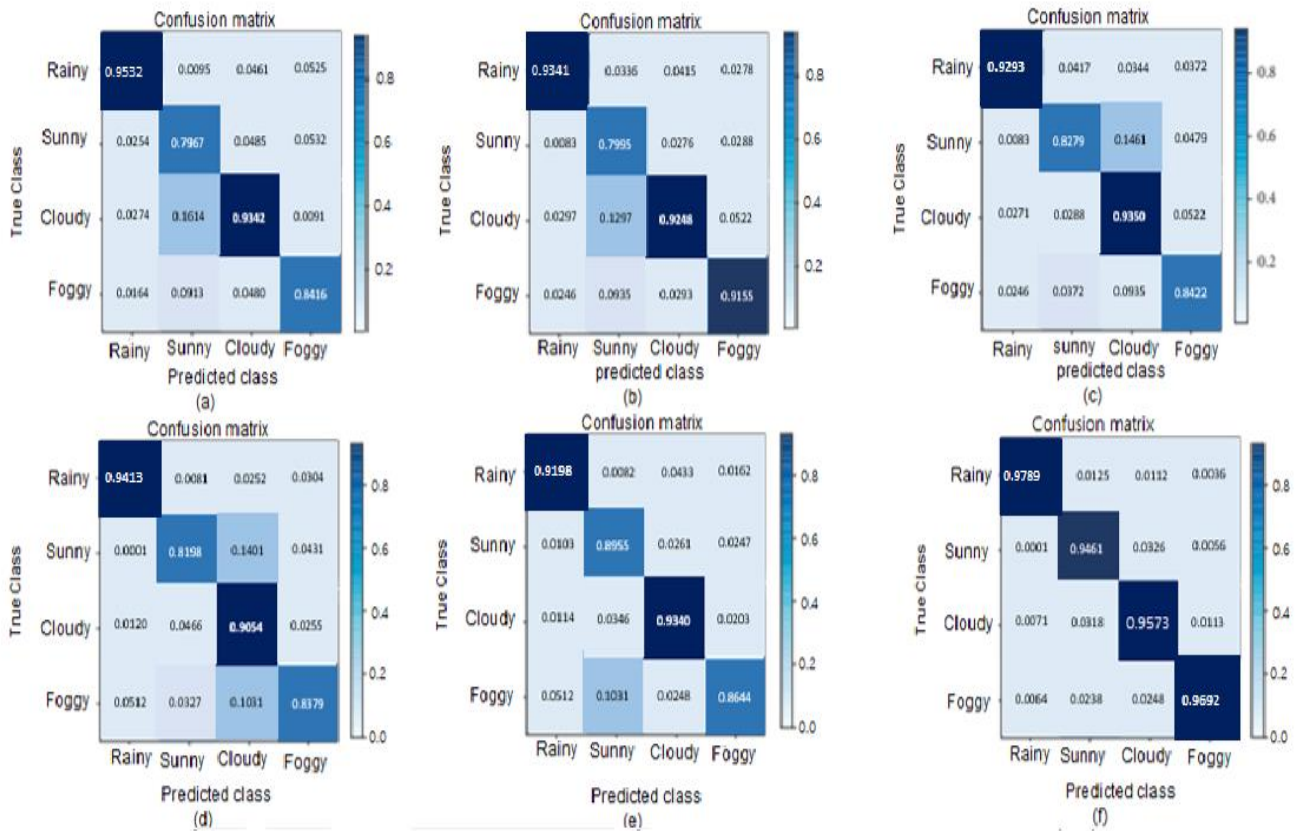


Figure 4.1: The Confusion matrix of various approaches. (a) AlexNet. (b) VGG16. (c) GoogleNet. (d) ResNet-50. (e) ResNet-18. (f) ResNet-15.

4.5 HYPERPARAMETER TUNING USING KERAS TUNER

Firstly, the necessary libraries and modules are imported, including TensorFlow, and the Keras Tuner library. Then the dataset2 was loaded and pre-processed to normalize the pixel values between 0 and 1. The build_model function was defined, which takes in a hyperparameter object (hp) as an argument. Inside the function, the model architecture

was defined and the hyperparameters to be tuned. The hyperparameters tuned are the number of units in the first dense layer, batch size, dropout, momentum, and the learning rate for the optimiser. The model was compiled with the hyperparameters specified in the `build_model` function. Then, an instance of the Random Search tuner was created, with the model-building function and setting the objective is to maximise validation accuracy. The tuner performed the hyperparameter search using `tuner.Search()`, with the training data, and 40 number of epochs, with a 0.2 validation split.

The best hyperparameters were retrieved using `tuner.get_best_hyperparameters()` function. Then, the model was built with the best hyperparameters using `tuner.hypermodel.build()` function. The model was trained using `model.fit()` on the training data with 40 number of epochs and validation split of 0.2. The model's performance was evaluated on the test set using `model.evaluate()` function, and the test accuracy recorded. The Hyperband optimisation algorithm was used to rapidly converge the model. In searching for the optimal hyperparameters, the early stop was applied to monitor the model. In this study a search space for each hyperparameter using `hp.uniform` and `hp.choice` functions from the Keras tuner library is defined.

The objective function evaluates the model with a specific set of hyperparameters, trains it on the training data, and evaluates its performance on the validation data. The loss value is then returned to be minimised. Additionally, the params dictionary containing the tuned hyperparameters is also returned. The best chosen hyperparameters are then recorded as seen in table 4.1

Table 4.1. Optimal combination of hyperparameters selected by ResNet-15 tuned with random search optimisation

Hyperparameters	Optimum value
Number of dense layers	32
Batch size	32
Learning rate	0.001
momentum	0.9
Dropout	0.4

As seen from table 4.1, the model performs better at 32 number of dense layers in the network to prevent its capacity to learn complex patterns from data, as compared to the initial range of values in table 3.4. At batch size 32, the number of samples processed by the model in each training iteration was better. A larger batch size could have led to faster training times, as the model updates its weights less frequently. However, smaller batch sizes such as 32, yields better generalisation, as it allowed the model to explore more diverse examples in each iteration. At 0.001 learning rate, it was able to determine the size of the step the model took during gradient descent while updating its weights. A higher learning rate could lead to faster convergence. While a lower learning rate result in slower convergence, the optimum learning rate at 0.001 was able to converge to a more stable solution. At 0.9 momentum, this parameter was used to accelerate the gradient descent in the relevant direction, which helped the model to navigate through flat regions more efficiently. With the momentum at 0.9, the model was also able to converge faster. While dropout regularization technique was used to prevent overfitting by randomly setting a fraction of input units to zero during training. This helped the model learn more robust features and reducing dependence on specific neurons. The dropout rate was used to determine the fraction of units that were dropped during training. An optimum dropout rate of 0.4 was more effective in preventing overfitting while also achieving faster convergence.

4.6. OPTIMISATION METHOD

4.6.1 RANDOM SEARCH OPTIMISATION

RANDOM SEARCH: A basic optimisation method that creates a grid of points by conducting independent trials for each. The points are chosen randomly, and a range of search values are defined for each hyperparameter, as opposed to a set of points. For hyperparameter optimisation, randomized trials are easier to implement and more effective than grid-search. In this study, the Random Search tuner was used to randomly search the hyperparameter space for 30 maximum number of trials. The hyperparameters being tuned are the number of units in the first dense layer, batch size, dropout, momentum, and the learning rate for the optimiser. Unlike grid search, random search

selects hyperparameter values randomly from a predefined distribution. By sampling a diverse range of values, it can be more efficient than grid search, especially when a few hyperparameters have a significant impact on model performance.

HYPERBAND ALGORITHM: A hyperparameter optimisation algorithm used in this study to swiftly converge the model through early stopping.

- ❖ **Momentum:** This method helps to speed up gradient descent algorithm, to achieve quicker convergence. The momentum increases in dimension whose gradient point is equal but decreases updates for dimensions whose gradient changes direction. In equation 4.2 and 4.3, the mathematical expressions for Momentum update rule are given.

$$v_t = m * v_{t-1} - \eta * \nabla_{\Theta} L(\Theta_{t-1}) \quad (4.2)$$

$$\Theta_t = \Theta_{t-1} + v_t \quad (4.3)$$

v , The variable for momentum at time t , initialised to zero from the start, and $m \in [0, 1]$ is the coefficient, needed for reducing the speed and fluctuations of the systems.

ADAM: A stochastic optimisation method whose most useful nature of optimisation is its adaptive learning rate. It was used in this research to calculate the adaptive learning rates for different parameters.

4.7 SYSTEM DEVELOPMENT LIFE CYCLE

The system development life cycle of a machine learning model is quite different from the conventional software development life cycle. During this research, the following **MLSDLC** (Machine learning software development life cycle) was followed.

- 1) Scope planning: This phase involves evaluating the scope of machine learning and how to improve the current system
- 2) Data preparation: This phase involves the data collection and design, and image preprocessing

- 3) Model design: This phase involves the model architecture, experiments involving the data training, data validation training and test data training. Which finally led to the selection of the best model
- 4) Evaluating the model: After the final version of the model, various metrics were tested on the test set to detect errors in the recognition and results are compared
Model deployment: This phase ensures that specifications of the hardware requirements are met before being used by the new system.
- 5) Monitoring: After evaluating the model, constant monitoring is needed to improve the new system.

4.8 CHAPTER CONCLUSION

This chapter has covered in detail the experimental setup and system configurations, including the choice for the selection of the framework used in this research. The feature selection and extraction was given in detail, while the model design and selection is also explained. The hyperparameter optimisation was presented, detailing the various steps taken to optimise the hyperparameters, then finally the machine learning development life cycle followed during the research, was also presented.

CHAPTER 5

EXPERIMENTAL RESULTS AND ANALYSIS

5.1 CHAPTER OVERVIEW

This chapter deals with the experiments done in the research. Thus, to evaluate the recommended approach, a medium size dataset was obtained which was augmented. Lastly, the metrics for evaluation, comparison methods and experimental results are presented respectively.

As already described in chapter 3, the architecture was built based on the ResNet-50 method to optimise ResNet-15, the two architectures were acted upon, by applying the enhancements such as, added dropout layers, eliminating some residual parts, also on the hyper parameters, such as, the value of the regularization strength. The subsequent section gives more details of the experimental settings, The results obtained by the two approaches for without and with hyperparameter optimisation are given in Section 5.4, section 5.5 gives the comparison of various approaches or techniques to optimise the ResNet architecture.

5.2 RESNET-15 EXPERIMENTAL METHOD

Firstly, the images are fed into the model and the convolutional layers of the network model are used for extracting the features from the images. Figure 5.1 demonstrates the pictorial extraction of feature maps from each group of the convolutional layers. The deeper the network, the less the pixel number of feature map showing further abstract features. Here, the weather features extracted from the former layer becomes the shortest route to the next layer through the four sets of residual modules to avoid the loss of significant features in the transmission process using the deep convolutional layers. Lastly, the weather images are classified and recognised through the fully connected layer and Softmax classifier. Figure 5.1 below shows the image of the feature maps from four convolutional layers of the model and the recognition results respectively.

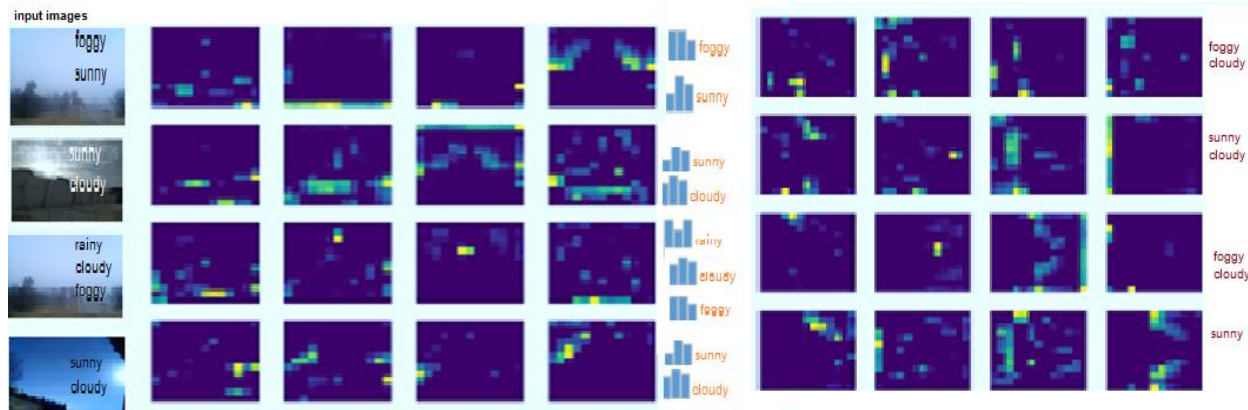


Figure 5.1: Image of the feature maps from four convolutional layers of ResNet-15 together with the recognition results.

5.3 EXPERIMENTAL RESULTS.

To validate the approach used in this study, the ResNet-15 is built based on python. ResNet-15 is mainly trained on the training set of “DATASET-2” then after training the model was saved and the validation set of images are fed into the training model for recognition purposes, then, the classification result is displayed into one of the four weather conditions, such as rainy, sunny, foggy, and cloudy. Lastly, in line with the recognition results of one thousand (1000) images, then the accuracy of recognition is computed, also, the confusion matrix of weather recognition is established, as can be seen in Figure 4.1(f). The recognition accuracy of each class is represented by the values on the slope of the confusion matrix respectively. As depicted in Figure 4.1(f), the recognition accuracy of the weather conditions is 97.89% rainy, 94.61% sunny, 95.73% cloudy, and 96.92% foggy

Without adding any additional parameters, the accuracy increased as the network depth increased and the rate of convergence of the residual network model was optimised. Hence, adding residual modules resolved the issues since deepening the network caused the gradient to vanish and the most important features that the convolutional layer had extracted were lost. Furthermore, computation got harder as depth increased. Thus, it became necessary to determine the difference between the

computational complexity and the model's performance. To achieve this with ResNet-15, experiments were run by stacking four or five residual modules. The ResNet-15 network configuration consists of four primary sets of residual modules, while the max-pooling layer focuses on extracting the feature maps and the average pooling layer extracts the background features, the pooling layer is used to extract significant features from the entire image. In this case, the procedure maintains crucial features in addition to merely reducing the amount of data processing.

Several combinations of maxpooling and average pooling were used in this study, which was conducted as experiments for comparison. The highest recognition accuracy was attained by the experimental results that combined the max-pooling layer added after the first convolutional layer and a fully connected layer with dimension (512) added before the Softmax classifier. To prevent overfitting, a dropout layer is also added in the middle of the fully connected layer and the Softmax classifier. This layer randomly discards a small amount of the fully connected layer's image. The trained model is compared with different dropout rates ranging from 0.3 to 0.9. The experimental results are displayed in Figure 5.2, where the model's recognition accuracy was at its highest when the dropout rate was set to 0.3. Using dropout as a regularization method has proven to improve model accuracy and combat overfitting. To prevent the loss of deeper levels of weather features from the convolutional layers, four more group of residual modules are added to the ResNet-15. Features from the images of the previous layer from to form the shortest route to the next layer, through the residual modules. Hence, its speeds up the rate of convergence of the network model. Thereby enhancing the recognition accuracy and at the same time, solves the vanishing gradient problem which was triggered by the increase in depth of the network.

ACCURACY OF COMPARISON

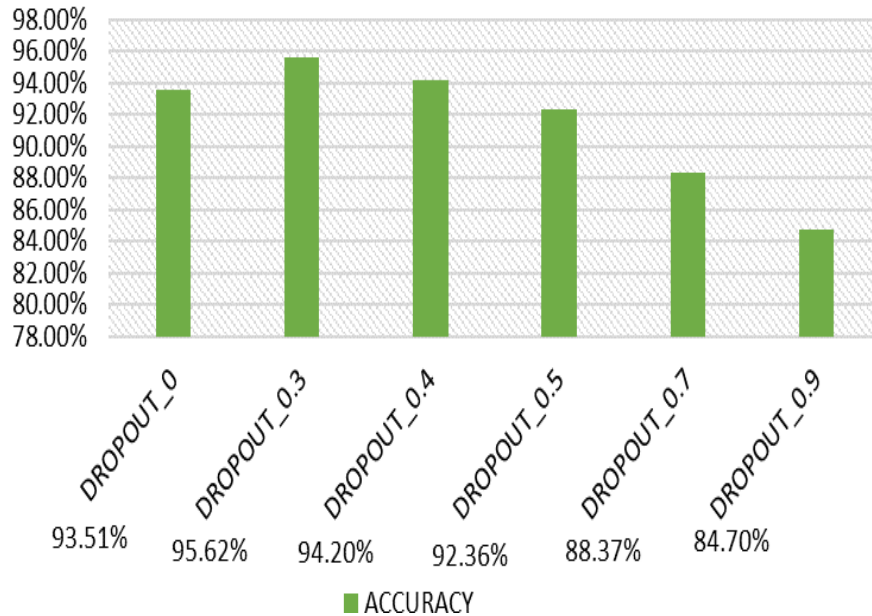


Figure 5.2: Comparing accuracy of different dropout rates.

Hence, the average accuracy of the weather recognition is 96.28%. As a result of that, ResNet-15 performed greatly to achieve the task of weather recognition from still images. Figure 5.3(a) shows the weather types that are accurately identified, the recognizable weather features such as heavy fog, level-ground, dark cloud, also, the sky blue enables weather images to be successfully recognised for a foggy day, rainy day, cloudy day, and sunny day, respectively. Additionally, fog reduces brightness, while rain will make the roads level-up, in most cases, not favorable for the traffic and can lead to road accidents and traffic-jams. Also, it becomes important to recognise severe weather conditions in real time, so that severe road accidents can be avoided effectively, and driving efficacy could be boosted.

Nevertheless, because of the difficulty in recognising the weather, it is not all the weather images that can be accurately recognised, since some related information in the weather images are quite difficult. The cause could be ascribed to “ambiguity” since there is no clear limit among the different weather classes. Additionally, at times weather recognition becomes a difficult process, due to other components in most images. This might be a multi-class task (Zhao et al., 2018, Lin et al., 2017), this can be summed up as

incompleteness. For that reason, different images cannot be recognised accurately as presented in Figure 5.3(b). For example, a hazy sky and background features fit into foggy weather, however, at times they are finally mis-recognised as snowy weather due to snow white cover on the street. At times, regardless of snow cover on the street, the presence of sky blue makes the image mis-recognised for sunny weather. Occasionally, the image is mis-recognised as foggy weather because of the difficult background features such as streaks of water, dark clouds, and grayish sky. Various images are very difficult to recognise correctly even when done manually, also the classes of the dataset may contain errors. In addition, the cleanliness of a dataset is also very significant as a characteristic for image classification. This implies that it should not be less than the size of the datasets (Mishkin et al., 2017). Hence, it ensures the important measures for enhancing the accuracy of weather recognition, by picking out the errors in the dataset labels, also, by removing the weather images with ambiguous labels.



(a)
Figure 5.3 Examples of correct recognition results

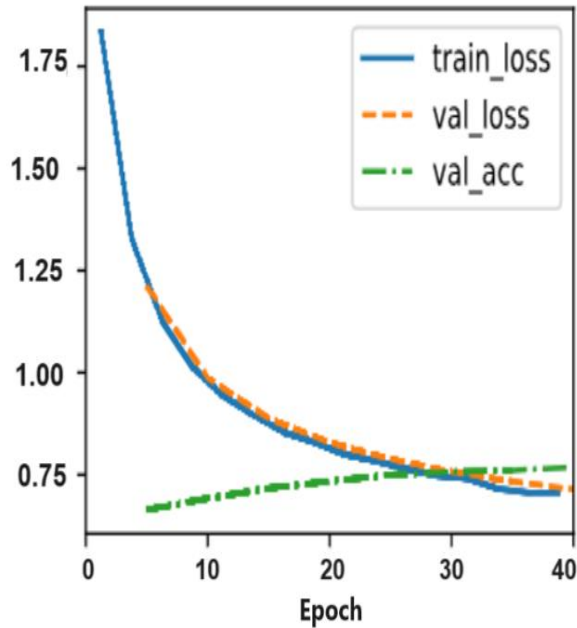


(b)
figure 5.3 Examples of incorrect recognition results

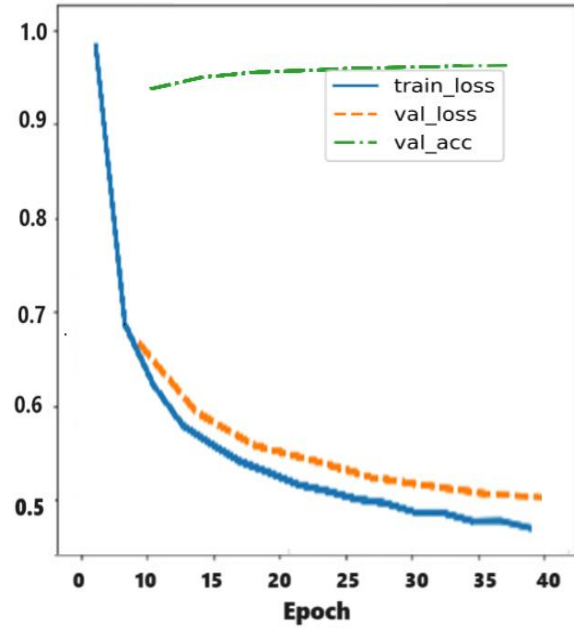
Figure 5.3(a): Examples of correct recognition results and **(b)** examples of incorrect recognition results.

5.4 IMPACT OF HYPERPARAMETER OPTIMISATION

In this study, to validate the efficiency of hyperparameter optimisation for weather recognition task, the “Dataset-2” training images with and without no hyperparameter tuning were input into the ResNet-15 model. Then, validation is done on the test images, the curves for accuracy and loss on the test set is displayed from Figure 5.4(a), from the curves, the blue curves represent the train loss on the train data, while the orange dashed curves represent the validation loss, and the green dashed curve represents the validation accuracy without hyperparameter tuning. In Figure 5.4(b), from the curves, the blue curves represent the train loss on the train data, while the orange dashed curves represent the validation loss, and the green dashed curve represents the validation accuracy with hyperparameter tuning. Table 5.1. Shows the experimental results of the two methods. We can observe from the graphs below that after the first epoch, the training loss reaches 1.25. After 10 epochs, accuracy reaches approximately 72%. We can compare the outcomes of not optimising the hyperparameters to those with hyperparameter optimisation using this result as a baseline.. The optimiser was successful in reaching a loss of about 0.66 after one epoch. Approximately 96% accuracy was reached after 5 epochs. We can observe from this experiment that optimising the hyperparameters performs better in this case more quickly and accurately. With this experiment, you can quickly see that optimising the hyperparameter is an effective way to develop a good deep learning model. From the results above, it is obvious that there is an increase of about 24% for ResNet-15 with hyperparameter tuning. It can be observed from the curves, that with hyperparameter tuning, the model is a perfect fit. In ResNet-15 , at approximately epoch 40. Thus, the test or validation set accuracy stopped increasing. In this study, the accuracy and training loss of a Deep Learning model can both be improved by optimising the hyperparameters.



(a)



(b)

Figure 5.4: Accuracy curve comparison. (a) Curves of accuracy for ResNet-15 without hyperparameter optimisation. (b) Accuracy curves of ResNet-15 with hyperparameter optimisation.

Table 5.1: Recognition accuracy of two approaches without and with hyperparameter optimisation.

Resnet-15	without hyperparameter Opt	with hyperparameter Opt
Val acc	72%	96%
Train loss	1.16	0.66
Val loss	1.17	0.66

To validate the efficiency of the residual module for weather recognition task, the residual modules were used to form a Fifteen-layer (15-layer) convolutional neural network known as *15-layers conv network*. The accuracy curves of the validation or test data is presented in Figure 5.4. The recognition accuracy of 15-layer conv network is about 96% with

hyperparameter optimisation, and that of the recognition accuracy of ResNet-15 without hyperparameter optimisation is 72%, which is enhanced by about 24% .

5.5 ACCURACY COMPARISON TO OTHER METHODS

Before training the models, image preprocessing was done by resizing images to a consistent size, normalizing pixel values, and splitting the dataset into training, validation, and testing sets according to the 80/20 rule. Each model was trained using the training set with the appropriate hyperparameters such as learning rate, dropout, batch size, number of dense layers, momentum, and the optimiser . Data augmentation techniques such as random rotations, flips, and crops to was applied to the images to enhance model generalisation. After each training epoch, the models' performance was evaluated using the validation set to monitor metrics such as accuracy, and loss to assess how well each model generalizes to unseen data and to detect overfitting. Hyperparameters fine-tuning was done based on validation set performance to adjust learning rates, regularization techniques, and model architectures as needed to improve model accuracy and generalisation.

When training and hyperparameter tuning was complete, the final evaluation of the models' performance was done using the testing set, to calculate metrics like accuracy, and confusion matrix to assess how well each model performs on completely unseen data. The performance Comparison of ResNet-50, AlexNet, ResNet-18, GoogleNet, VGG16, and ResNet-15 based on their accuracy, computational efficiency, and robustness was done. To analyse any patterns or insights gained from model comparisons to understand which architecture performs best for the given dataset and task. By following these approaches, the models were systematically evaluated to determine the accuracy of deep learning models using the selected dataset2, leading to informed decisions on model selection, hyperparameter tuning, and optimisation.

ResNet-15 hyperparameters are optimised through random search optimisation and tested on a test dataset. The results show that ResNet-15 Random search has a higher recognition accuracy than ResNet50, Resnet18, AlexNet, GoogleNet, and VGG16 when applied to the selected dataset. To assess the performance of weather recognition using ResNet-15, the recognition performance of ResNet-15 on “Dataset-2”

is being compared to the other approaches such as, AlexNet, VGG16, GoogleNet, ResNet-50, and ResNet-18. Figure 5.5 illustrates the accuracy curves of several approaches, as can be seen from the curves, the various colored curves signifies different approaches. As depicted from the curves, ResNet-15 which is the recommended approach in this study outperformed the other approaches in terms of accuracy. Therefore, it can be said that random search hyperparameter tuning for the residual network produces an accuracy rate that is higher than that of some earlier versions of residual networks. Results for Dataset 2 were found to be better with ResNet 15 than with ResNet 50 when comparing the ResNet random search optimisation and other techniques. The training took place for 40 epochs. And it obvious that Resnet-15 surpasses all the other deep learning techniques. Not only does the training converge very fast, but also the loss function is much lower than the other deep learning techniques.

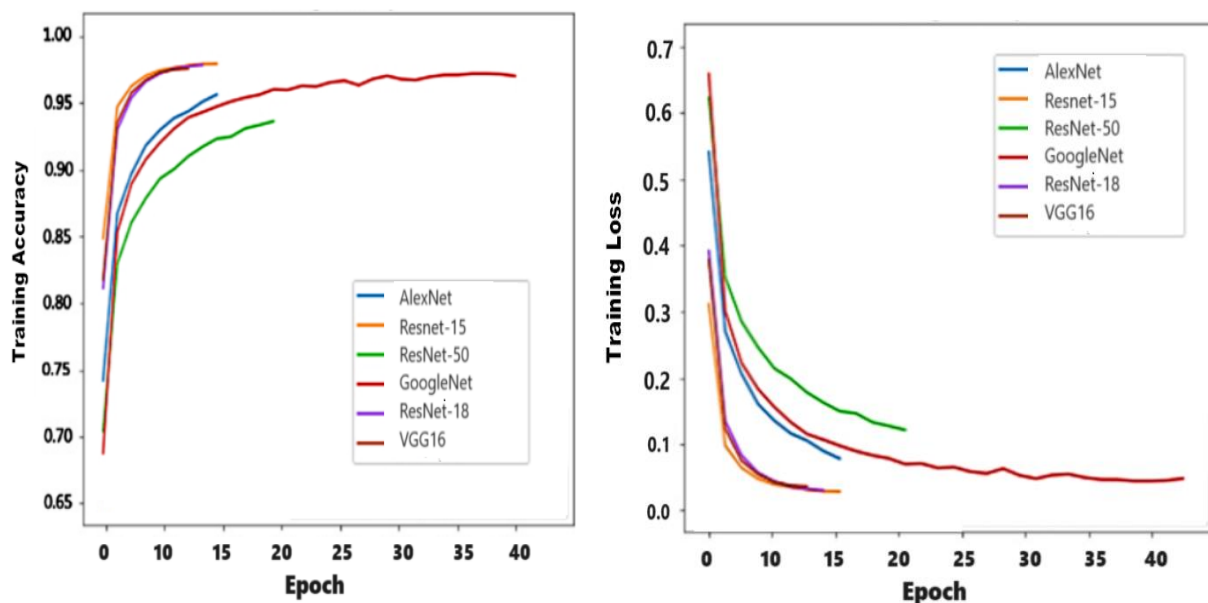
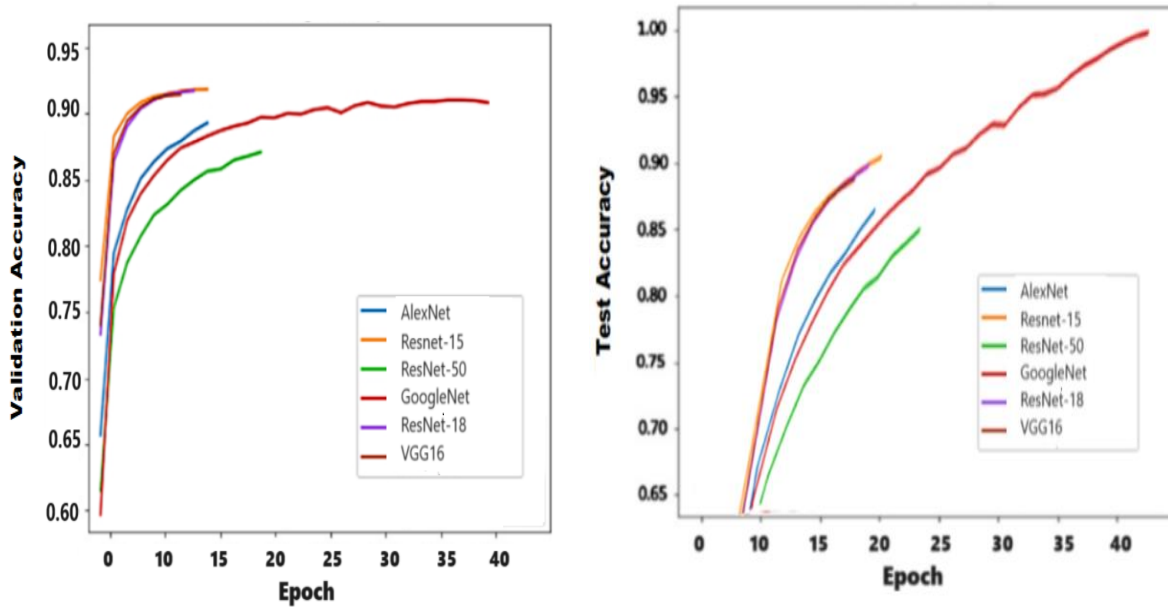


Figure 5.5: Accuracy curves of several methods (a) Training accuracy (b) Training Loss curve



(c) Validation accuracy .(d) Test accuracy

ResNet- was tested on the selected dataset-2 to obtain the accuracy on the test data, it obtained 98% training accuracy, 92% validation accuracy and a test accuracy of 93%. Hyperband optimisation method was done on the ResNet with 40 maximum number of trials, and 40. The model was then trained using the hyperparameter configuration displayed in Table 3.4. Selecting the best combination of hyperparameters chosen by ResNet-15 network was tuned. From the recognition accuracies of different approaches, a chart is drawn as displayed in figure 5.5, it can be observed that ResNet-15 recommended in this study obtained the highest recognition accuracy, the next in accuracy to Resnet-15 is the ResNet18, VGG16, GoogleNet, AlexNet, and ResNet50 respectively. The confusion matrices of various network models are depicted in Figure 4.1.

Table 5.2: Recognition accuracy of various approaches

Approach	Train (%)	Validation(%)	Test (%)
AlexNet	92.03	84.00	85.01
VGG16	97.01	86.00	87.06
GoogleNet	92.00	86.04	84.25
ResNet-50	90.01	79.00	80.05
ResNet-18	97.00	88.00	89.00
ResNet-15	98.09	92.01	93.03

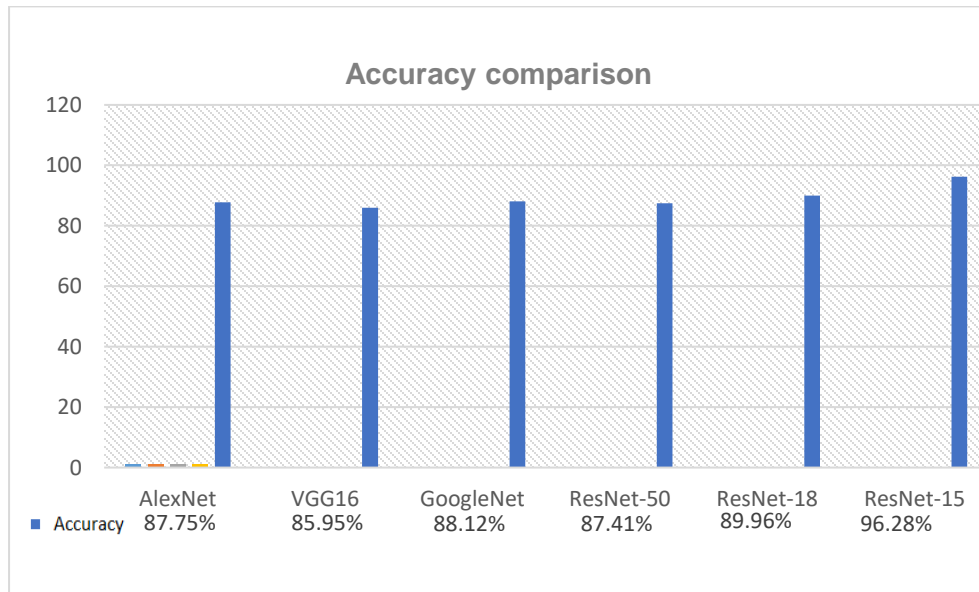


Figure 5.6: Comparison of average recognition accuracy

Table 5.2 shows the experimental results of different approaches based on the training accuracy, validation accuracy, and test accuracy. From the experimental results, Resnet-15 shows the highest level of accuracy, thus, the model is highly superior to other approaches in recognition speed and model size and can broadly be useful to a wide range of fields.

5.6 CHAPTER CONCLUSION

This study mainly aimed at the task of weather recognition using a ResNet-15 technique, to recognise the presence of the weather condition from images, the optimised hyperparameter was able to do the feature extraction and the fully connected layer was capable of recognising the images. Evaluating from an intensive experiment shows that the ResNet-15 model attains a desired result on the Dataset-2. The simplified model recognition accuracy achieved 97.09%, Thus, this approach is much better as compared to the conventional network model such as ResNet-50 in terms of recognition accuracy, recognition speed, and size of the model. Hence, the recommended model in this study ResNet-15 can generally meet-up with the necessities of real-world application, also, it can be extensively used in other fields or domains of life. The dataset is currently grouped into four classes of weather conditions. Experiments also shows that the effects of hyperparameter optimisation has a great impact on the model's accuracy, performance, and robustness.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CHAPTER OVERVIEW

This chapter is a summary of the entire research done to accomplish the research goals and objectives, it also presents the limitations of the research and provides recommendation for future work.

6.2 CONCLUSION

In conclusion, tuning hyperparameters in line with the objectives of optimisation plays a crucial role in enhancing the performance, efficiency, and generalizability of deep Learning models.

The following contributions are provided by this research.

- ❖ Improving the model's overall performance through enhanced accuracy and reducing loss accuracy, which has also improved the quality of recognitions is one of the benefits associated with optimising the hyperparameters of the model. It is necessary to get improved outcomes and come up with well-informed decisions from results achieved in the model's output.
- ❖ Also reducing the training time has led to faster experimentation, it has also become important for this purpose, to tune the hyperparameters of the model.
- ❖ Hyperparameter optimisation has enhanced the model's generalisation capability. by finding the right combination of hyperparameters, hence preventing overfitting , leading to improved performance on unseen data. In addition to the above, data augmentation experiments done on Resnet-15 , resulted in more capable models, during the experiments, the augmentation method created variants of the images, by improving the capability of fitting the models to generalise what has been learnt to new images. It significantly reduced the overfitting problem
- ❖ The process became more efficient by automating search for hyperparameters; it also became more effective using random searches, it was able to select the best

hyperparameters. Automating hyperparameter tuning made it easier to explore hyperparameter search space reducing manual intervention while improving productivity. These optimal hyperparameters were chosen using random search optimisation method for ResNet-15 that fits the selected dataset obtained 98% training accuracy, 92% validation accuracy, and 93% test accuracy

Conclusively, this research presents a different perception into the optimisation of deep learning models for resolving the challenges around weather recognition from still images. Since random search optimisation has not been widely used with ResNet for weather recognition application, this study leads to new optimised ResNet-15 model.

6.3 LIMITATIONS

In this study there are limitations, some methods which might have facilitated in attaining good results were not experimented. An example, this study did not use the freezing layer method used in a transfer learning, which in the case of this study was not used. At the point of training the network it was understood that it was not useful due to the high computational time involved.

One more limitation of this study was recognised in the dataset. The dataset presented various distorted images that could have contributed to reduce the performance such as unclear labelled images, noisy images, and murky images. Lastly, this study focused on ResNet-15 using hyperparameter optimisation and without hyperparameter optimisation. Hence, accuracy of the recognitions are compared with or without hyperparameter optimisation.

6.4 FUTURE WORK

It would be intriguing to investigate additional optimisation methods for image-based weather recognition in the future. From the result of this study, I would make a recommendation for future work, I recommend refining the dataset content by disposing images that do not have useful information to the task, such as the noisy ones. It is also sensible to examine the impact of the results when the size of classes to recognise increases. For example, based on the dataset, the number of images for each class that

represents the best ones could be identified. Perhaps, this can enhance performances. Then, the class extension as regards the weather conditions characterized by each class could be lessened, therefore it makes it easier for the recognition of such classes, this can prevent the occurrence of different features showing the exact weather condition in a single class.

Perhaps a different type of input such as video can be applied, such that some dynamic images can be captured properly and possibly easier to recognise. Thus, since this study used feature visualization, it would be worthwhile to gain an additional understanding by exploring other means rather than feature visualization, to know how likely to identify the features which permits discrimination between the different classes.

Lastly, for future work, the severe weather conditions could be split into more sections. Particularly, heavy, light rain, and moderate rain. Besides, the model is only used to recognise the weather conditions from different backgrounds during the daytime. I suggest the approach could generally be used to recognise weather conditions at nighttime through future research and development.

REFERENCES

- Ajayi, Gbeminiyi (2018), "Multi-class Weather Dataset for Image Classification", Mendeley Data,
- Akihiro. F. (2020). Assemble-ResNet that is 5 times faster with the same accuracy as EfficientNet B6 AutoAugment. Published in Analytics Vidhya.
- Aszemi. N.M and P.D.D. (2019), Dominic, Hyperparameter optimisation in convolutional neural network using genetic algorithms, Int J Adv Computer Sci Appl 10 269–278. <https://doi.org/10.14569/ijacsa.2019.0100638>
- Bengio. Y. (2009) Learning deep architectures for AI , Foundations, and trends R In Machine Learning 2.
- Bossu. J, Hautière. N and Tarel. J.P. (2011) Rain or snow detection in image sequences through use of a histogram of orientation of streaks, international journal of computer vision 93, 348.
- Bronte. S, Bergasa. L. M , Alcantarilla, P. F. (2009) Fog detection system based on computer vision techniques, 12th International IEEE Conference on Intelligent Transportation Systems, pp. 1–6.
- Brown, D. (2021) "We show that camera data can be efficiently processed using computer vision techniques for weather pattern recognition, reducing the overall cost of weather monitoring systems."
- Cai. Y, Huafeng. W, Pan. H, Weifeng. L , Zhang, Y. (2015). Deep Learning for Image Retrieval: What Works and What Doesn't. DO - 10.1109/ICDMW.2015.121.
- Cewu Lu, Lin Di, Jiaya Jia, and Chi-Keung Tang,(2014) "Two-class weather classification," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3718–3725.
- Chen .H , Ding. G, Zhao. S, Han. J. (2018) Temporal-difference learning with sampling baseline for image captioning, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence.

- Christian S, Wei. L, Yangqing. J, Pierre , Scott. R, Dragomir . A, Dumitru. E, Vincent. V, Andrew. R.(2015) “Going deeper with convolutions,” Cvpr.
- Connor. S,(2019). Introduction to ResNet. Published in towards data science.
- Davison. A, Ganau. S, Marti. J, Sentís. M, Pons. G, Yap. M, Zwigelaar. R.(2017). Automated Breast Ultrasound Lesions Detection Using Convolutional Neural Networks. IEEE Journal of Biomedical and Health Informatics. DO - 10.1109/JBHI.2017.2731873.
- Deng. J, Dong. W, Socher. R, Li. L.J, Li. K and Fei-Fei. L. (2009) ImageNet: A large-scale hierarchical image database, in Computer Vision and Pattern Recognition, 2009.CVPR2009.IEEEConference on (IEEE, pp. 248–255.
- Derpanis. K. G. Lecce. M. Daniilidis. K. Wildes. R. P. (2012) Dynamic scene understanding: The role of orientation features in space and time in scene classification, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1306–1313.
- Di Lin, Cewu Lu, Hui Huang, and Jiaya Jia. (2017) “Rscm: Region selection and concurrency model for multi-class weather recognition,” IEEE Transactions on Image Processing, vol. 26, no. 9, pp. 4154–4167.
- Dupond S. A thorough review on the current advance of neural network structures. Annu Rev Control. 2019;14:200–30.
- Elhoseiny. M. Huang. S, Elgammal. A. (2015) Weather classification with deep convolutional neural networks, in: Image Processing (ICIP), 2015 IEEE International Conference on, pp. 3349–3353.
- Gallen. R, Cord. A, Hautire. N, Aubert. D. (2011) Towards night fog detection through use of in-vehicle multipurpose cameras, in: Intelligent Vehicles Symposium (IV), IEEE, 2011, pp. 399–404.
- Garg. K, and Nayar. S. K. (2004) Detection and removal of rain from videos, in Computer Vision and Pattern Recognition, CVPR Proceedings of the 2004 IEEE Computer Society Conference on, Vol. 1 (IEEE, 2004) pp. 1–528.

- Garg. K, and Nayar. S. K. (2005). When does a camera see rain? in Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, Vol. 2, pp. 1067–1074.
- Goodfellow, I, Bengio, Y and Courville, A. (2016) Deep learning MIT Press.
- Gu. J, Wang, Z, Kuen, J, Ma, L, Shahroudy. A, Shuai, B, Liu, T, Wang, X and Wang. G.(2015) Recent advances in convolutional neural networks, arXiv preprint arXiv:1512.07108.
- G´abor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. arXiv preprint arXiv:1707.05589, 2017
- Garcia, E. (2017) "Our analysis indicates that cameras have lower maintenance costs compared to sensor devices for weather monitoring, contributing to their cost-effectiveness."
- Hauti´ere. N, Tarel. J.P, Lavenant. J, Aubert. D. (2006)Automatic fog detection and estimation of visibility distance through use of an on-board camera, Machine Vision, and Applications 17 (1) 8–20.
- He, K, Zhang, X, Ren, S and Sun, J.(2015) Deep residual learning for image recognition, arXiv preprint arXiv:1512.03385 (2015). Used twice.
- He. K, Zhang. X, Ren. S and Sun. J.(2016) Identity mappings in deep residual networks, in European Conference on Computer Vision pp. 630–645.
- He. K, Zhang. X, Ren. S and Sun. J. (2016)"Deep residual learning for image recognition," in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, IEEE, Las Vegas, NV, USA.
- He. K,Gkioxari. G, Dollr.P, Girshick. R.(2017) Maskr-CNN, in IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988.
- JalFaizy. S. (2017). "Advanced deep learning architectures". Published in Analytics Vidhya

- Jindal. A, Dua. A , Kaur. K, Singh. M, Kumar. N and Mishra. S.(2016) “Decision tree and SVM-based data analytics for the detection in smart grid, ”IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1005–1016.
- Johnson, B, (2020) “Our cost analysis reveals that cameras are more cost-effective for weather monitoring compared to specialized sensor devices due to economies of scale and lower production costs.”
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.(2015) "Deep Residual Learning for Image Recognition" Computer Vision and Pattern Recognition (cs.CV).
- Kamavisdar. P, Saluja, S and Agrawal, S. (2013) “A survey on image classification approaches and techniques, International Journal of Advanced Research in Computer and Communication Engineering 2, 1005”.
- Karayiannis. N. And Venetsanopoulos, A. N. (2013) “Artificial neural networks: learning algorithms, performance evaluation, and applications”, Vol. 209 Springer Science & Business Media.
- Kingma, D. P, Ba. J and Adam: “A method for stochastic optimisation”, CoRR abs/1412.6980.
- Krizhevsky. A, Sutskever, I and Hinton, G.E . (2012)“ImageNet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105.
- Krizhevsky. A, Sutskever, I and Hinton, G.E. (2017) “ImageNet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84–90.
- Kurihata.H,Takahashi.T,Mekada.Y,Ide.I,Murase.H,Tamatsu.Y,Miyahara.T.(2005), “Rainy weather recognition from in-vehicle camera images for driver assistance”, in: IEEE Proceedings. Intelligent Vehicles Symposium, pp. 205–210.
- Kurihata.H,Takahashi.T,Mekada.Y,Ide.I,Murase.H,Tamatsu.Y,Miyahara.T.(2006) “Rain drop detection from in-vehicle video camera images for rainfall judgment”, in: First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06), Vol. 2, pp. 544–547.

- Leandro .P. (2018). “ Parkinson Disease Identification Using Residual Networks and Optimum-Path Forest”. DO - 10.1109/SACI.2018.8441012
- Lee. J, Kim. S, Chang. S ,Zong-Sheng. W.(2020) “Efficient Chaotic Imperialist Competitive Algorithm with Dropout Strategy for Global Optimisation”. DO - 10.3390/sym12040635. VL - 12.
- Li. Q, Kong, Y and Xia, S.M. (2014) “A method of weather recognition based on outdoor images,” in Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP), IEEE, Lisbon, Portugal, pp. 510–516.
- Lin. D, Lu, C, Huang, H and Jia, J.(2017) “RSCM: region selection and concurrency model for multi-class weather recognition,” IEEE Transactions on Image Processing, vol. 26, no. 9, pp. 4154–4167.
- Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. “Hyperband a novel bandit-based approach to hyperparameter optimisation”. arXiv preprint arXiv:1603.06560, 18(1):6765–6816, 2016.
- Lu, C, Lin, D, Jia, J and Tang, C. K. (2014) “Two-class weather classification”, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '14, 2014, pp. 3718–3725.
- Lu, C, Lin, D, Jia, J and Tang, C. K. (2017) “Two-class weather classification,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2510–2524.
- Michael, N.(2005) “Artificial intelligence a guide to intelligent systems”, ISBN 321204662.
- Mishkin, D, Sergievskiy, N and Matas, J. (2017) “Systematic evaluation of convolution neural network advances on the ImageNet,” Computer Vision and Image Understanding, vol. 161, pp. 11–19.
- Narasimhan, S. G. Wang, C and Nayar, S. K. (2002) “All the images of an outdoor scene,” in Computer Vision-ECCV 2002, pp. 148–162, Springer, Berlin, Germany.
- Pavlic, M, Belzner, H, Rigoll, G, Ili, S.(2012) “Image based fog detection in vehicles, in: Intelligent Vehicles Symposium” (IV), IEEE, 2012, pp. 1132–1137.

- Pavlic, M, Rigoll, G, Ilic, S. (2013) “Classification of images in fog and fog-free scenes for use in vehicles”, in IEEE Intelligent Vehicles Symposium (IV), Gold Coast City, Australia, June 23-26, pp. 481–486.
- Qian, X, Patton, E.W. Swaney, J, Xing, Q and Zeng, T. (2018) “Machine learning on cataracts classification using Squeeze Net”, “in Proceedings of the 2018 4th International Conference on Universal Village (UV), pp.1–3, IEEE, Boston, MA, USA.
- Ren, S, He, K, Girshick, R, Sun, J. (2015) “Faster R-CNN: Towards real-time object detection with region proposal networks”, in: Advances in Neural Information Processing Systems (NIPS).
- Roser, M, Moosmann, F.(2008) “Classification of weather situations on single color images”, in: Intelligent Vehicles Symposium, IEEE, 2008, pp. 798–803.
- Ruder, S. (2016). “An overview of gradient descent optimisation algorithms”, arXiv preprint arXiv:1609.04747.
- Sai Kumar .B. (2019). “CNN Architectures, a Deep dive. Published in towards Data Science”
- Sawant, S and Ghonge, P. (2013) “Estimation of rain drop analysis using image processing”, International Journal of Science and Research (IJSR).
- Shen, L, Tan, P. (2009) “Photometric stereo and weather estimation using internet images”, in Computer Vision and Pattern Recognition, IEEE Conference on, pp. 1850–1857.
- Simonyan, K, Zisserman, A. (2014) “Very deep convolutional networks for large-scale image recognition”, arXiv preprint arXiv:1409.1556.
- Simonyan, K, Zisserman, A. “Very deep convolutional networks for large-scale image recognition”, CoRR abs/1409.1556.
- Smith, A, (2019) “Cameras offer a versatile solution for weather recognition in smart agriculture due to their ability to capture visual data on cloud cover, precipitation, and other atmospheric phenomena.”
- Song, H, Chen, Y, Gao, Y.(2015) “Weather Condition Recognition Based on Feature Extraction and K-NN,” Springer Berlin Heidelberg, 2014, pp. 199–210.

- Srivastava, N, Hinton, G.E, Krizhevsky, A, Sutskever, I and Salakhutdinov, R. (2014) "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.
- Sudheer, K and Panda, R.(2000) "Digital image processing for determining drop sizes from irrigation spray nozzles, Agricultural Water Management". 45, 159.
- Sultan. H, (2019). "Multi-Classification of Brain Tumor Images Using Deep Neural Network." DO - 10.1109/ACCESS.2019.2919122. IEEE Access, VL - PP.
- Szegedy, Liu, W, Jia, Y, Sermanet, P, Reed, S, Anguelov, D, Erhan, D, Vanhoucke, V and Rabinovich, A.(2015) "Going deeper with convolutions" ,in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp. 1–9.
- Tang, P, Wang, H, and Kwong, S. (2017) G-MS2F: "GoogleNet based multi-stage feature fusion of deep CNN for scene recognition," Neurocomputing, vol. 225, pp. 188–197.
- Tena .B. (2018). "Your first dive into deep learning.
- Williams, C. (2018) "Our study demonstrates the cost savings of integrating weather recognition cameras with existing surveillance systems, minimizing additional infrastructure costs."
- Xia, J., Xuan, D., Tan, L. and Xing, L., 2020. ResNet15: weather recognition on traffic road with deep convolutional neural network. Advances in Meteorology, 2020, pp.1-11.
- Xiaoqiang. L, Xuelong. L, Zhao. B, Zhigang. W.(2019) "A CNN-RNN Architecture for Multi-Label Weather Recognition." Computer Vision and Pattern Recognition (cs.CV); Artificial Intelligence (cs.AI). arXiv:1904.10709.
- Yan, X Luo, Y, Zheng, X. (2009) "Weather recognition based on images captured by vision system in vehicle," in: Proceedings of the 6th International Symposium on Neural Networks: Advances in Neural Networks-Part III,ISNN2009, Springer Verlag, pp. 390–398.

- Yosinski, J, Clune, J, Bengio, Y and Lipson , H.(2004) “How transferable are features in deep neural networks? in Advances in neural information processing systems pp. 3320–3328.
- Zeiler, M. D and Fergus, R.(2014) “Visualizing and understanding convolutional networks, “in European conference on computer vision Springer, pp. 818–833.
- Zhao, B, Li, X. Lu, X and Wang, Z. (2018) “A CNN-RNN architecture for multi-label weather recognition,” Neurocomputing, vol. 322, pp. 47–57, 2018.
- Zhang, Z, Ma, H. (2015) “multi-class weather classification on single images,” in: Image Processing (ICIP), IEEE International Conference on, pp.4396–4400.
- Zhang, Z and Huadong Ma. (2015) “multi-class weather classification on single images,” in Image Processing (ICIP), IEEE International Conference on. IEEE, pp. 4396–4400.
- Zhang, Z, Ma, H. Fu, C, Zhang. (2016) “Scene-free multi-class weather classification on single images,” Neurocomputing 207 365 – 373.
- Zhao, X, Liu, P, Liu, J, Tang, X. (2011) “A time, space and color-based classification of different weather conditions”, in: Visual Communications and Image Processing (VCIP), IEEE, pp. 1–4.
- Zhu, Z ,Zhuo, L. Qu, P, Zhou, K and Zhang, J.(2016) “Extreme weather recognition using convolutional neural networks”, in Multimedia (ISM),IEEE International Symposium on IEEE, pp. 621–625.
- Ziqi Zhu, Zhuo, Li, Panling Qu, Kailong Zhou, and Jing Zhang.(2016) “Extreme weather recognition using convolutional neural networks,” in Multimedia (ISM),IEEE International Symposium on. IEEE, pp. 621– 625.
- Zixiang .M. (2019). “A Wi-Fi RSSI ranking fingerprint positioning system and its application to indoor activities of daily living recognition.” International Journal of Distributed Sensor Networks, VL - 15.

Appendix A: Sample python source codes

A1: Listing one

DETAILED DESCRIPTION: This code section describes the main procedure used for loading and plotting the first eight photos of rainy image in a single figure. The code runs the example created in a figure showing the first eight photos of rain, shine, cloudy and foggy images respectively in the dataset.

```
# plot rainy photos from the rain dataset
from matplotlib import pyplot
from matplotlib.image import imread
# define location of dataset
folder = 'train/'
# plot first few images
for i in range(8):
# define subplot
pyplot.subplot(350 + 1 + i)
# define filename
filename = folder + 'rain.' + str(i) + '.jpg'
# load image pixels
image = imread(filename)
# plot raw pixel data
pyplot.imshow(image)
# show the figure
pyplot.show()

# plot shine photos from the shine dataset
from matplotlib import pyplot
from matplotlib.image import imread
# define location of dataset
folder = 'train/'
# plot first few images
for i in range(8):
# define subplot
pyplot.subplot(350 + 1 + i)
# define filename
filename = folder + 'shine.' + str(i) + '.jpg'
# load image pixels
image = imread(filename)
# plot raw pixel data
pyplot.imshow(image)
# show the figure
pyplot.show()

# plot cloudy photos from the cloudy dataset
from matplotlib import pyplot
from matplotlib.image import imread
```

```

# define location of dataset
folder = 'train/'
# plot first few images
for i in range(8):
# define subplot
pyplot.subplot(350 + 1 + i)
# define filename
filename = folder + 'cloudy.' + str(i) + '.jpg'
# load image pixels
image = imread(filename)
# plot raw pixel data
pyplot.imshow(image)
# show the figure
pyplot.show()

```

```

# plot foggy photos from the cloudy dataset
from matplotlib import pyplot
from matplotlib.image import imread
# define location of dataset
folder = 'train/'
# plot first few images
for i in range(8):
# define subplot
pyplot.subplot(350 + 1 + i)
# define filename
filename = folder + 'foggy.' + str(i) + '.jpg'
# load image pixels
image = imread(filename)
# plot raw pixel data
pyplot.imshow(image)
# show the figure
pyplot.show()

```

Appendix A: Sample python source codes

A2: Listing Two, image resizing.

DETAILED DESCRIPTION: The code in this section below uses the Keras image processing API to load all photos in the training dataset and reshapes them to 256x256 square photos. The label is also determined for each photo based on the filenames.

```

# load rainy, sunny, cloudy, and foggy dataset, reshape and save to a new
file
from os import listdir
from numpy import asarray
from numpy import save

```

```

from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
# define location of dataset
folder = 'train/'
photos, labels = list(), list()
# Itemize files in the directory
for file in listdir(folder):
# define class
output = 0.0
if file.starts with('shine'):
output = 1.0
# Load image
photo = load_img(folder + file, target_size=(256, 256))
# convert to numpy array
photo = img_to_array(photo)
# store
photos.append(photo)
labels.append(output)
# convert to a numpy arrays
photos = as array(photos)
labels = as array(labels)
print(photos.shape, labels.shape)
# save the reshaped photos
save(rainy_sunny_cloudy_and_foggy_photos.npy', photos)

save('rain_vs_shine_cloudy_vs_foggy_labels.npy', labels)

```

Appendix A: Sample python source codes

A3: Listing Three

DETAILED DESCRIPTION: The code in this section below randomly holds back 20% of the images into the test and 80% into train dataset. This is done consistently by fixing the seed for the pseudorandom number generator so that it can get the same split of data each time the code is run.

```

# Organize the dataset into a very useful structure
from os import makedirs
from os import listdir
from shutil import copyfile
from random import seed
from random import random
# create directories
dataset_home = 'dataset_ rainy_sunny_cloudy_and_foggy/'
subdirs = [ 'train/', 'test/' ]
for subdir in subdirs:
# create label subdirectories
labldirs = ['rainy/', 'sunny/']
for labldir in labldirs:

```

```

newdir = dataset_home + subdir + labldir
mkdirs(newdir, exist_ok= True)
# seed random number generator
seed(1)
# define ratio of pictures to use for validation
val_ratio = 0.20
# copy training dataset images into subdirectories
src_directory = 'train/'
for file in listdir(src_directory):
src = src_directory + '/' + file
dst_dir = 'train/'
If random() < val_ratio:
dst_dir = 'test/'
If file.startswith('shine'):
dst = dataset_home + dst_dir + 'sunny/' + file
copyfile(src, dst)
el if file.startswith('rain'):
dst = dataset_home + dst_dir + 'rainy/' + file
copyfile(src, dst)

```

Appendix A: Sample python source codes

A4: Listing Four

DETAILED DESCRIPTION: The code in this section uses the Image Data Generator in keras to train the dataset which is augmented with small (20%) random horizontal and vertical shifts and random horizontal flips that create a mirror image of a photo. Photos in both the train and test steps will have their pixel values scaled in the same way.

```

# baseline model with data augmentation for the rain,shine,cloudy and foggy
dataset
import sys
from matplotlib import pyplot
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimisers import SGD
from keras.preprocessing.image import ImageDataGenerator

# define the CNN model
def define_model():
model = Sequential()

```

```

model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(256, 256,
3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(Dense(4, activation='softmax'))
# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])
return model

# plot diagnostic learning curves
def summarize_diagnostics(history):
# plot loss
pyplot.subplot(214)
pyplot.title('Loss')
pyplot.plot(history.history['loss'], color='blue', label='train')
pyplot.plot(history.history['val_loss'], color='orange', label='validation')
pyplot.xlabel('Epochs')
pyplot.ylabel('loss')
pyplot.legend(loc='lower right')
pyplot.show()

# plot accuracy
pyplot.subplot(215)
pyplot.title('Classification Accuracy')
pyplot.plot(history.history['accuracy'], color='blue', label='train')
pyplot.plot(history.history['val_accuracy'], color='orange',
label='validation')
pyplot.xlabel('Epochs')
pyplot.ylabel('Accuracy')
pyplot.legend(loc='lower right')
pyplot.show()
# save plot to file
filename = sys.argv[0].split('/')[-1]
pyplot.savefig(filename + '_plot.png')
pyplot.close()

# define model
model = define_model()
# create data generators

```

```

train_datagen = ImageDataGenerator(rescale=1.0/255.0,
width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1.0/255.0)
# prepare iterators
train_it = train_datagen.flow_from_directory('dataset_
rainy_sunny_cloudy_and_foggy/',
train/',
class_mode='categorical', batch_size=64, target_size=(256, 256))
test_it = test_datagen.flow_from_directory('dataset_rainy_vs_sunny/test/',
class_mode='categorical', batch_size=64, target_size=(256, 256))
# fit model
history = model.fit_generator(train_it, steps_per_epoch=len(train_it),
validation_data=test_it, validation_steps=len(test_it), epochs=100,
verbose=0)
# evaluate model
_, acc = model.evaluate_generator(test_it, steps=len(test_it), verbose=0)
print('> %.3f ' % (acc * 100.0))
# learning curves
summarize_diagnostics(history)

```

Appendix A: Sample python source codes

A5: Listing Five

DETAILED DESCRIPTION: This code in this section defines the baseline model with the addition of Dropout. In this case, a dropout of 20% is applied after each ResNet15 block, with a larger dropout rate of 50% applied after the fully connected layer in the classifier part of the model.

```

# baseline model with dropout for the rain and shine dataset
import sys
from matplotlib import pyplot
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimisers import SGD
from keras.preprocessing.image import ImageDataGenerator

# define the CNN model
def define_model():
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(256, 256,
3)))

```

```

model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.3))

model.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(0.5))
    model.add(Dense(4, activation='softmax'))
# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])
return model

# plot diagnostic learning curves
def summarise_diagnostics(history):
# plot loss
pyplot.subplot(214)
pyplot.title('Cross Entropy Loss')
pyplot.plot(history.history['loss'], color='blue', label='train')
pyplot.plot(history.history['val_loss'], color='orange', label='validation')
    pyplot.xlabel('Epochs')
    pyplot.ylabel('loss')
    pyplot.legend(loc='lower right')
    pyplot.show()
# plot accuracy
pyplot.subplot(215)
pyplot.title('Classification Accuracy')
pyplot.plot(history.history['accuracy'], color='blue', label='train')
pyplot.plot(history.history['val_accuracy'], color='orange',
label='validation')
    pyplot.xlabel('Epochs')
    pyplot.ylabel('Accuracy')
    pyplot.legend(loc='lower right')
    pyplot.show()
# save plot to file
filename = sys.argv[0].split('/')[0]
pyplot.savefig(filename + '_plot.png')
pyplot.close()

# define model
model = define_model()
# create data generators

```

```

train_datagen = ImageDataGenerator(rescale=1.0/255.0,
width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1.0/255.0)
# prepare iterators
train_it = train_datagen.flow_from_directory('dataset_rainy_vs_sunny/train/',
class_mode='categorical', batch_size=64, target_size=(256, 256))
test_it = test_datagen.flow_from_directory('dataset_
rainy_sunny_cloudy_and_foggy/',
test/',
class_mode='categorical', batch_size=64, target_size=(256, 256))
# fit model
history = model.fit_generator(train_it, steps_per_epoch=len(train_it),
validation_data=test_it, validation_steps=len(test_it), epochs=100,
verbose=0)
# evaluate model
_, acc = model.evaluate_generator(test_it, steps=len(test_it), verbose=0)
print('> %.3f ' % (acc * 100.0))
# learning curves
summarize_diagnostics(history)

```

Appendix A: Sample python source codes

A6: Listing six

DETAILED DESCRIPTION: The code here describes how the final model is typically fit on all available data, such as the combination of all train and test datasets.

```

# save the final model to file
from keras.applications.ResNet15 import ResNet15
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimisers import SGD
from keras.preprocessing.image import ImageDataGenerator

# define the CNN model
def define_model():
# load model
model = ResNet15(include_top=False, input_shape=(224, 224, 3))
# mark loaded layers as not trainable
for layer in model.layers:
layer.trainable = False
# add new classifier layers
flat1 = Flatten()(model.layers[-1].output)
class1 = Dense(128, activation='relu',
kernel_initializer='he_uniform')(flat1)
output = Dense(4, activation='softmax')

```



```

# define new model
model = Model(inputs=model.inputs, outputs=output)
# compile model
opt = SGD(lr=0.001, momentum=0.9)
model.compile(optimiser=opt, loss='categorical_crossentropy',
metrics=['accuracy'])
return model

# define model
model = define_model()
# create data generator
datagen = ImageDataGenerator(featurewise_center=True)
# specify dataset2 mean values for centering
datagen.mean = [133.68, 118.879, 104.939]
# prepare iterator
train_it = datagen.flow_from_directory
('finalise_rainy_sunny_cloudy_and_foggy/',
class_mode='categorical', batch_size=64, target_size=(224, 224))
# fit model
model.fit_generator(train_it, steps_per_epoch=len(train_it), epochs=20,
verbose=0)

```

Appendix A: Sample python source codes

A7: Listing seven

DETAILED DESCRIPTION: The code here describes how to use the saved model to make a recognition on new images. The model assumes that new images are color and they have been segmented so that one image contains at least one rainy or sunny, and cloudy or foggy.

```

# make a recognition for a new image.
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model

# load and prepare the image
def load_image(filename):
# load the image
img = load_img(filename, target_size=(224, 224))
# convert to array
img = img_to_array(img)
# reshape into a single sample with 3 channels
img = img.reshape(1, 224, 224, 3)

```

```

# center pixel data
img = img.astype('float32')
img = img - [133.68, 118.879, 104.939]
return img

# load an image and predict the class
def run_example():
# load the image
img = load_image ('dataset2_image.jpg')
# predict the class
result = model.predict(img)
print(result[0])

```

Appendix A: Sample python source codes

A8: Listing Eight

DETAILED DESCRIPTION: The code describes the experiment showing the impact of no hyperparameter optimisation and with hyperparameter optimisation.

EFFECTS WITHOUT HYPERPARAMETER USING SGD OPTIMISATION

```

import matplotlib.pyplot as plt
model = tf.keras.models.Sequential(
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(4, activation='softmax'))
sgd = tf.keras.optimisers.SGD(lr=0.01, decay=1e-6, momentum=0.0,
nesterov=True)
model.compile(optimizer=sgd,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(x_train, y_train,
epochs=40)
model.evaluate(x_test, y_test, verbose=2)# Plot training & validation
accuracy values
plt.plot(history.history['acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train loss', 'Val loss', 'Val acc'], loc='upper right')

```

```

plt.show()# Plot train loss, Validation loss & validation accuracy values
plt.plot(history.history['loss'])

# plot accuracy
pyplot.subplot(215)
pyplot.title('Classification Accuracy')
pyplot.plot(history.history['Val acc'], color='green',
pyplot.plot(history.history['train loss, color='blue'],
pyplot.plot(history.history['val loss, color='oranges'],
    pyplot.xlabel('Epochs')
    pyplot.legend(loc='upper right')
    pyplot.show()

```

EFFECTS WITH HYPERPARAMETER USING SGD OPTIMISATION

```

import matplotlib.pyplot as plt
model = tf.keras.models.Sequential(
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(4, activation='softmax'))
sgd = tf.keras.optimisers.SGD(lr=0.01, decay=1e-6, momentum=0.9,
nesterov=True)
model.compile(optimizer=sgd,
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
history = model.fit(x_train, y_train,
epochs=40)
model.evaluate(x_validation, y_validation, verbose=2)# Plot train loss,
validation accuracy & validation loss values
plt.plot(history.history['acc','loss'])
plt.xlabel('Epoch')
plt.legend(['Train loss', 'Val loss', Val acc], loc='upper right')
plt.show()# Plot train loss, validation loss & validation accuracy values
# plot accuracy
pyplot.subplot(215)
pyplot.title('Classification Accuracy')
pyplot.plot(history.history['Val acc'], color='green',
pyplot.plot(history.history['train loss, color='blue'],
pyplot.plot(history.history['val loss, color='orange'],
    pyplot.xlabel('Epochs')
    pyplot.legend(loc='lower right')
    pyplot.show()

plt.xlabel('Epoch')

```

```
plt.legend(['Train loss', 'Val loss', 'val acc'], loc='upper right')
plt.show()
```

Appendix A: Sample python source codes

A9: Listing Nine

DETAILED DESCRIPTION: The code describes the development of the model using keras tuner for tuning the hyperparameters.

HYPERPARAMETER OPTIMISATION WITH KERAS TUNER

#Prepare the dataset

```
From tensorflow import keras # importing keras
(x_train, y_train), (x_test, y_test) =
keras.datasets.dataset2.load_data() # loading the data using keras
datasets api
x_train = x_train.astype('float32') / 255.0 # normalize the training
images
x_test = x_test.astype('float32') / 255.0 # normalize the testing
images
```

Develop the baseline model

```
model1 = keras.Sequential()
model1.add(keras.layers.Flatten(input_shape=(28, 28))) # flattening 28
x 28
model1.add(keras.layers.Dense(units=512, activation='relu',
name='dense_1')) # you have 512 neurons with relu activation
model1.add(keras.layers.Dropout(0.3)) # added a dropout layer with the
rate of 0.3
model1.add(keras.layers.Dense(4, activation='softmax')) # output layer,
total of 4 classes
```

Compile and train the model

```
model1.compile(optimiser=keras.optimisers.Adam(learning_rate=0.001),
               loss=keras.losses.CategoricalCrossentropy(),
               metrics=['accuracy'])
model1.fit(x_train, y_train, epochs=40, validation_split=0.2)
```

#Evaluate the model

```
model1_eval = model.evaluate(img_test, label_test, return_dict=True)
# plot diagnostic learning curves
def summarize_diagnostics(history):
# plot accuracy
pyplot.subplot(215)
pyplot.title('Training Accuracy')
pyplot.plot(history.history['Accuracy'],
pyplot.plot(history.history['Alexnet'], color='blue',
pyplot.plot(history.history['ResNet-15'], color='orange',
pyplot.plot(history.history['ResNet-50'], color='green',
pyplot.plot(history.history['gooleNet'], color='tangerine',
pyplot.plot(history.history['ResNet-18'], color='purple',
pyplot.plot(history.history['VGG16'], color='brown',
label='train')
    pyplot.xlabel('Epoch')
    pyplot.ylabel('Training Accuracy')
    pyplot.legend(loc='lower right')
    pyplot.show()
# plot loss
pyplot.subplot(215)
pyplot.title('Training loss')
pyplot.plot(history.history['loss'],
pyplot.plot(history.history['Alexnet'], color='blue',
pyplot.plot(history.history['ResNet-15'], color='orange',
pyplot.plot(history.history['ResNet-50'], color='green',
pyplot.plot(history.history['gooleNet'], color='tangerine',
pyplot.plot(history.history['ResNet-18'], color='purple',
pyplot.plot(history.history['VGG16'], color='brown',

label='Training loss')
pyplot.plot(history.history['Training_loss'],
label='loss')
    pyplot.xlabel('Epochs')
    pyplot.ylabel('Training loss')
    pyplot.legend(loc='upper right')
    pyplot.show()

# plot accuracy
pyplot.subplot(215)
pyplot.title('Validation Accuracy')
pyplot.plot(history.history['accuracy'],
```

```

pyplot.plot(history.history['Alexnet'], color='blue',
pyplot.plot(history.history['ResNet-15'], color='orange',
pyplot.plot(history.history['ResNet-50'], color='green',
pyplot.plot(history.history['gooleNet'], color='tangerine',
pyplot.plot(history.history['ResNet-18'], color='purple',
pyplot.plot(history.history['VGG16'], color='brown',

label='Validation')
pyplot.plot(history.history['val_accuracy'],
label='Validation')
    pyplot.xlabel('Epochs')
    pyplot.ylabel('Validation Accuracy')
    pyplot.legend(loc='lower right')
    pyplot.show()

# plot accuracy
pyplot.subplot(215)
pyplot.title('Test Accuracy')
pyplot.plot(history.history['accuracy'],
pyplot.plot(history.history['Alexnet'], color='blue',
pyplot.plot(history.history['ResNet-15'], color='orange',
pyplot.plot(history.history['ResNet-50'], color='green',
pyplot.plot(history.history['gooleNet'], color='tangerine',
pyplot.plot(history.history['ResNet-18'], color='purple',
pyplot.plot(history.history['VGG16'], color='brown',

label='Test')
pyplot.plot(history.history['val_accuracy'],
label='Test')
    pyplot.xlabel('Epochs')
    pyplot.ylabel('Test Accuracy')
    pyplot.legend(loc='lower right')
    pyplot.show()

# save plot to file
filename = sys.keras[0].split('/')[ -1]
pyplot.savefig(filename + '_plot.png')
pyplot.close()

```

Tuning the model using Keras Tuner

```

import tensorflow as tf
from tensorflow import keras

```

```

import keras_tuner as kt
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
# Load and preprocess the dataset2
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.dataset2.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0 # Normalize pixel
values to between 0 and 1

# Define the model-building function for Keras Tuner
def build_model(hp)
    model = keras.Sequential()

    # Tune the number of units in the first dense layer
    hp_units = hp.Int('units', min_value=32, max_value=512, step=32)
    model.add(layers.Flatten(input_shape=(28, 28)))
    model.add(layers.Dense(units=hp_units, activation='relu'))
    model.add(layers.Dropout(0.3))

model.add(layers.Dense(10, activation='softmax'))

    # Tune the learning rate for the optimiser
    hp_learning_rate = hp.Choice('learning_rate', values=[0.01, 0.001,
0.0001])

model.compile(optimiser=keras.optimisers.Adam(learning_rate=hp_learning
_rate),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

    return model

tuner =
kt.Hyperband(model_builder,objective='val_accuracy',max_epochs=3,
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=5)
tuner.search(x_train, y_train, epochs=2, validation_split=0.2,
callbacks=[stop_early])

# Instantiate the RandomSearch tuner and perform hyperparameter tuning
tuner = RandomSearch(
    build_model,
    objective='val_accuracy',
    max_trials=30,
    directory='my_tuning_directory',

```

```

    study_name='dataset2_tuning')

tuner.search(x_train, y_train, epochs=5, validation_split=0.2)

# Get the best hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

# Build the model with the best hyperparameters
model = tuner.hyperband.build(best_hps)

# Train the model
model.fit(x_train, y_train, epochs=10, validation_split=0.2)

# Get the best hyperparameters

stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=5)
# Perform hypertuning
tuner.search(x_train, y_train, epochs=10, validation_split=0.2,
callbacks=[stop_early])
best_hp=tuner.get_best_hyperparameters()[0]
best_hps=tuner.get_best_hyperparameters(num_trials=1)[0]
print(f"BEST num neurons for Dense Layer : {best_hps.get('units')}")
print(f"BEST learning_rate : {best_hps.get('learning_rate')}")
print(f"BEST dropout rate : {best_hps.get('dropout rate')}")
print(f"BEST momentum : {best_hps.get('momentum')}")
print(f"BEST batch size : {best_hps.get('batch size')}")

# Build the model with the best hyperparameters
model = tuner.hypermodel.build(best_hps)

# Train the model
model.fit(x_train, y_train, epochs=10, validation_split=0.2)

# Evaluate the model on the test set
train_accuracy, train_loss, validation_accuracy, test_accuracy =
model.evaluate(x_test, y_test, verbose=2)
print(Test Accuracy: {test_accuracy*100:.2f}%)

#Rebuild and Train the Model with the best hyperparameters
# Build the model with the best hyperparameters
h_model = tuner.hypermodel.build(best_hps)
h_model.summary()
h_model.fit(x_train, x_test, epochs=10, validation_split=0.2)

```


#evaluate the model,

```
h_eval_dict = h_model.evaluate(img_test, label_test, return_dict=True)
tuner.search(x_train, y_train, epochs=40, validation_split=0.2,
callbacks=[stop_early])
```

```
best_hp=tuner.get_best_hyperparameters()[0]
```

#Rebuild and Train the Model with the best hyperparameters

```
# Build the model with the best hyperparameters
```

```
h_model = tuner.hypermodel.build(best_hps)
```

```
h_model.summary()
```

```
h_model.fit(x_train, x_test, epochs=40, validation_split=0.2)
```

#evaluate the model,

```
h_eval_dict = h_model.evaluate(img_test, label_test, return_dict=True)
```

```
#tune the hyperparameters of momentum, batch size, and dropout using
```

```
Keras tuner library:
```

```
import numpy as np
```

```
from keras.models import Sequential
```

```
from keras.layers import Dense, Dropout
```

```
from keras.optimisers import SGD
```

```
# Define the search space for hyperparameters
```

```
space = {
```

```
'momentum': hp.uniform('momentum', 0.1, 0.9),
```

```
'batch_size': hp.choice('batch_size', [16, 32, 64]),
```

```
'dropout': hp.uniform('dropout', 0.0, 0.9)
```

```
}
```

```
# Define the objective function to minimize (training loss)
```

```
def objective(params):
```

```
model = Sequential()
```

```
model.add(Dense(64, input_dim=100, activation='relu'))
```

```
model.add(Dropout(params['dropout']))
```

```
model.add(Dense(1, activation='sigmoid'))
```

```
optimiser = SGD(lr=0.01, momentum=params['momentum'])
```

```
model.compile(loss='binary_crossentropy', optimiser=optimiser,
```

```
metrics=['accuracy'])
```

```
# Train the model with the given hyperparameters
```

```
history = model.fit(X_train, y_train, epochs=10,
```

```
batch_size=params['batch_size'], validation_split=0.2, verbose=0)
```

```
# Evaluate the model on validation data
loss, accuracy = model.evaluate(X_val, y_val, verbose=0)

# Return the tuned hyperparameters and the validation loss
return {'loss': loss, 'status': STATUS_OK, 'params': params}
# Use hyperband (HB) algorithm for optimisation
best_hyperparams = hb(objective, space, algo=hb.suggest, max_evals=10,
trials=trials)print(best_hyperparams)
```

**UNISA COLLEGE OF SCIENCE, ENGINEERING AND TECHNOLOGY'S
(CSET) RESEARCH AND ETHICS COMMITTEE**

09 September 2019

Ref #: 059/PUE/2019/CSET_SOC
Name: Mrs Peace Uloma Egbueze
Student #: 63947129

Dear Mrs Peace Uloma Egbueze

**Decision: Ethics Approval for 3 years
(No Humans involved)**

Researchers: Mrs Peace Uloma Egbueze, 63947129@mylife.unisa.ac.za, +27 84 025 9187;
+27 71 342 3198

Project Leader(s): Prof Zenghui Wang, wangz@unisa.ac.za, +27 11 471 3513

Working title of Research:

Weather recognition based on still images using deep learning neural network

Qualification: MSc in Computing

Thank you for the application for research ethics clearance by the Unisa College of Science, Engineering and Technology's (CSET) Research and Ethics Committee for the above mentioned research. Ethics approval is granted for a period of three years, from 09 September 2019 to 09 September 2022.

1. The researcher will ensure that the research project adheres to the values and principles expressed in the UNISA Policy on Research Ethics.
2. Any adverse circumstance arising in the undertaking of the research project that is relevant to the ethicality of the study, as well as changes in the methodology, should be communicated in writing to the Unisa College of Science, Engineering and Technology's (CSET) Research and Ethics Committee. An amended application could



University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

be requested if there are substantial changes from the existing proposal, especially if those changes affect any of the study-related risks for the research participants.

3. The researcher(s) will conduct the study according to the methods and procedures set out in the approved application.
4. Any changes that can affect the study-related risks for the research participants, particularly in terms of assurances made with regards to the protection of participants' privacy and the confidentiality of the data, should be reported to the Committee in writing, accompanied by a progress report.
5. The researcher will ensure that the research project adheres to any applicable national legislation, professional codes of conduct, institutional guidelines and scientific standards relevant to the specific field of study. Adherence to the following South African legislation is important, if applicable: Protection of Personal Information Act, no 4 of 2013; Children's act no 38 of 2005 and the National Health Act, no 61 of 2003.
6. Only de-identified research data may be used for secondary research purposes in future on condition that the research objectives are similar to those of the original research. Secondary use of identifiable human research data requires additional ethics clearance.
7. Submission of a completed research ethics progress report will constitute an application for renewal of Ethics Research Committee approval.

Note:

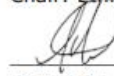
The reference number 059/PUE/2019/CSET_SOC should be clearly indicated on all forms of communication with the intended research participants, as well as with the Unisa College of Science, Engineering and Technology's (CSET) Research and Ethics Committee.

Yours sincerely



Dr. B Chimbo

Chair: Ethics Sub-Committee SoC, College of Science, Engineering and Technology (CSET)



2019/09/12

Dr MG Katumba

Director: School of Computing, CSET



Prof B Mamba

Executive Dean: CSET



Approved - decision template – updated Aug 2016

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA, 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

**UNISA COLLEGE OF SCIENCE, ENGINEERING AND TECHNOLOGY'S
(CSET) ETHICS REVIEW COMMITTEE**

2023/04/26

Dear Mrs Peace Uloma Egbueze

ERC Reference # :
059/PUE/2019/CSET_SOC Egbueze
RENEWAL

Name: Mrs Peace Uloma Egbueze

Student #: 63947129

Staff #: N/A

**Decision: Ethics Approval from
2023/04/26 to 2026/04/26
Non-human/non-animal**

**This certificate is an amendment
to certificate number
059/PUE/2019/CSET_SOC**

Researcher(s): Mrs Peace Uloma Egbueze, 63947129@mylife.unisa.ac.za, +27 84 025 9187;
+27 71 342 3198

Project Leader(s): Prof Zengghui Wang, wangz@unisa.ac.za, +27 11 471 3513

Working title of research:

Weather recognition based on still images using deep learning neural network

Qualification: MSc in Computing

Thank you for the application for research ethics clearance by the Unisa College of Science, Engineering and Technology's (CSET) Ethics Review Committee for the above-mentioned research. Ethics approval is granted for 3 years.

*The **negligible risk application** was expedited by the College of Science, Engineering and Technology's (CSET) Ethics Review Committee on 2023/04/26 in compliance with the Unisa Policy on Research Ethics and the Standard Operating Procedure on Research Ethics Risk Assessment. The decision will be tabled at the next Committee meeting for ratification.*



University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

The proposed research may now commence with the provisions that:

1. The researcher will ensure that the research project adheres to the relevant guidelines set out in the Unisa COVID-19 position statement on research ethics attached.
2. The researcher(s) will ensure that the research project adheres to the values and principles expressed in the UNISA Policy on Research Ethics.
3. Any adverse circumstance arising in the undertaking of the research project that is relevant to the ethicality of the study should be communicated in writing to the College of Science, Engineering and Technology's (CSET) Ethics Review Committee.
4. The researcher(s) will conduct the study according to the methods and procedures set out in the approved application.
5. Any changes that can affect the study-related risks for the research participants, particularly in terms of assurances made with regards to the protection of participants' privacy and the confidentiality of the data, should be reported to the Committee in writing, accompanied by a progress report.
6. The researcher will ensure that the research project adheres to any applicable national legislation, professional codes of conduct, institutional guidelines and scientific standards relevant to the specific field of study. Adherence to the following South African legislation is important, if applicable: Protection of Personal Information Act, no 4 of 2013; Children's act no 38 of 2005 and the National Health Act, no 61 of 2003.
7. Only de-identified research data may be used for secondary research purposes in future on condition that the research objectives are similar to those of the original research. Secondary use of identifiable human research data requires additional ethics clearance.
8. No field work activities may continue after the expiry date 2026/04/26. Submission of a completed research ethics progress report will constitute an application for renewal of Ethics Research Committee approval.

Note

The reference number 059/PUE/2019/CSET_SOC Egbueze RENEWAL should be clearly indicated on all forms of communication with the intended research participants, as well as with the Committee.

Yours sincerely,



Dr D Bisschoff (Signed by Prof JH Kroeze on behalf of Dr Bisschoff)

Chair of School of Computing Ethics Review Subcommittee

College of Science, Engineering and Technology (CSET)

E-mail: dbischof@unisa.ac.za

Tel: (011) 471-2109



Dr. K.T. Masombuka

Acting Director: School of Computing

College of Science Engineering and

Technology (CSET)

E-mail: Masomkt@unisa.ac.za

Tel: 0116709123

pp:



Prof. B Mamba

Executive Dean

College of Science Engineering and

Technology (CSET)

E-mail: mambabb@unisa.ac.za

Tel: (011) 670 9230



URERC 25.04.17- Decision template (V2) - Approve

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za

***VME ENGLISH LANGUAGE
SCIENTIFIC EDITING
SERVICES***

I, VINCENT MANUEL hereby confirm that I have proof read and edited the MSc in Computing

Titled
WEATHER RECOGNITION BASED ON STILL IMAGES USING DEEP
LEARNING NEURAL NETWORK

By
PEACE ULOMA EGBUEZE

The windows "Tracking" System was used to reflect my comments and suggested corrections are given for the author to action.

During the process of the proof reading and editing, the following changes were recommended: punctuations, grammatical and sentence construction and how to improve on coherence of the document. In addition consistency in use of abbreviations, uniformity, referencing style (in-text and reference list) were given by the editor. Although greatest care was taken in editing this document, the final responsibility for the product rests with the author.



Editor's signature

05-01-2024

Date

Dr. V. Manuel
Language and Writing and Editing Consultant
Intermediate Certificate in Business English & Communication (CRPP Qualification, UK)
Further Education in Teacher's Certificate (FCEZ, Zambia)
Post grad Diploma in Mass Communications (Zambia)

ENGLISH EDITING SERVICES:



146 Commissioner Street, JHB, South Africa
Vinceymanuel@webmail.co.za

Cell: 0840943421