

**ALGERIAN REPUBLIC DEMOCRATIC AND POPULAR MINISTRY OF  
HIGH EDUCATION AND SCIENTIFIC RESEARCHES**

University of Mohamed Khider Biskra  
Faculty of Exact Science and Science of Life and  
Nature Department of Computer Science



Dissertation For Master Degree Graduation In Computer Science  
Field Computer Graphics & Computer Vision

**Title:**

**Weather Conditions Image  
Classification**

Using Deep Learning

**Realized by:**

Naassi Mohamed Salah

Defended the 14/09/2020, in front of the jury composed of:

Academic year 2019-2020

# Dedication

I dedicate my dissertation work to my teachers in life my parents and my lovely grand mother whose lessons still in my ears since my childhood, my only brother who is always in my side.

I also dedicate this dissertation to my many friends: Zaki, Omar, Oussama, Okba, Amine, Iliasse, Abdelhak, Louai, Cherif, Nacer, khaled, zakaria who are always supported me. I will always appreciate all they have done.

I dedicate this work to the distinguished friend Hatem who helped and accompanied me throughout my five years of university.

# Acknowledgement

First of all, I have to thank Allah for giving me strength and ability to complete this research work. Throughout the writing of this dissertation I have received a lot of support and assistance.

I would like to thank my supervisor, Doctor Mokhtari Bilel, whose expertise was invaluable in the final achievement of this research. Your insightful feedback made me focus and guide my work to a higher level possible.

My thanks go to the member jury for their efforts to evaluate this work despite the hard time that the world is going through.

# Abstract

Weather Recognition directly or indirectly impacts our daily behaviors, it can decide whether if we need to walk, take a bus, riding a bike, traveling, or even not go outside at all. Despite the importance of this point, it is still not widely studied by deep learning and computer vision. Weather conditions are often addressed with limited number of classes or in specific visual condition. In this work, we take advantage of deep learning that automatically extract information from images using one of the strongest algorithms in computer vision tasks Convolutional neural networks without any pre-defined constraints or specific features in the processed images. The proposed model shows strong performance in extracting this information from user-defined images in five weather conditions from different kind of scenes like nature, street-level, and surveillance camera.

**Key-words:** Deep learning, Weather recognition, Features, Convolution neural networks.

# Contents

<b>1</b>	<b>Image Classification in Computer Vision</b>	<b>13</b>
1.1	Introduction .....	13
1.2	Image basics and color Terminology .....	13
1.2.1	Digital Image .....	13
1.2.1.1	What is a pixel? .....	13
1.2.1.2	Image types .....	14
1.2.2	Human visual system .....	15
1.2.2.1	Description of the visual system .....	15
1.2.2.2	Human Eye Anatomy .....	16
1.2.2.3	How Do We See Color? .....	17
1.2.3	A quick look about Color .....	19
1.2.3.1	What is a color? .....	19
1.2.3.2	Color vision .....	19
1.2.3.3	What is the visible light spectrum? .....	19
1.2.3.4	Additive, subtractive models .....	21
1.3	Computer Vision .....	22
1.3.1	Computer vision what does it mean? .....	22
1.3.1.1	Computer vision and human vision .....	22
1.3.1.2	Applications .....	23
1.3.2	Computer vision vs. image processing .....	26
1.4	Image Classification .....	26
1.4.1	Definition .....	26
1.4.2	Features and classifiers .....	26
1.4.2.1	Features extraction (hand-engineered).....	27
1.4.2.2	Features extraction (automatic-engineered).....	28
1.4.3	Classification process .....	28
1.4.3.1	Image collection .....	28
1.4.3.2	Image preprocessing .....	28
1.4.3.3	Training .....	28
1.4.3.4	Classifying.....	28
1.4.3.5	Evaluation.....	29
1.5	Weather conditions image Classification .....	29
1.5.1	Introduction.....	29
1.5.2	Description.....	29
1.5.3	Features examples.....	29

1.5.4	Possible applications .....	30
1.6	Conclusion .....	31
<b>2</b>	<b>Deep Learning</b>	<b>32</b>
2.1	Introduction.....	32
2.2	Introduction to neural networks .....	32
2.2.1	Definition .....	32
2.2.2	Principal architectures .....	32
2.2.2.1	Single-Layer Feedforward Networks.....	32
2.2.2.2	Multi-Layer Feedforward Networks.....	33
2.2.3	Learning process (types) .....	34
2.2.2.1	Supervised learning .....	34
2.2.2.2	Unsupervised learning.....	34
2.2.2.3	Reinforcement learning .....	35
2.2.2.4	Semi-supervised learning.....	35
2.3	Deep learning.....	36
2.3.1	Introduction.....	36
2.3.2	Definition.....	36
2.3.3	The deep learning revolution .....	36
2.3.4	Deep learning models .....	37
2.3.4.1	Autoencoders.....	37
2.3.4.2	Deep Belief Network (DBN) .....	37
2.3.4.3	Generative Adversarial Networks (GANs) .....	38
2.3.4.4	Recurrent Neural Network (RNNs) .....	39
2.3.4.5	Long Short-Term Memory (LSTM) .....	40
2.3.4.6	Recursive Neural Networks (RNNs) .....	40
2.3.4.7	Convolutional Neural Networks (CNNs) .....	41
2.3.5	Generalization in deep neural networks.....	50
2.3.5.1	Definition.....	50
2.3.5.2	Cross-validation.....	50
2.3.5.3	Dropout.....	51
2.3.5.4	L1 and L2 Regularizations .....	51
2.3.6	Deep learning applications.....	52
2.3.6.1	Entertainment.....	52
2.3.6.2	Fraud detection.....	52
2.3.6.3	Healthcare.....	52
2.3.6.4	Virtual assistants.....	53
2.3.7	Related works.....	53
2.3.7.1	Music genre classification.....	53

2.3.7.2	Lung cancer detection and classification.....	53
2.3.7.3	Photo-realistic facial Details synthesis from single image.....	54
2.3.7.4	Deepfakes creation and detection.....	54
2.3.7.5	Human activity recognition.....	54
2.3.7.6	Weather image recognition.....	55
2.4	Conclusion.....	55
<b>3</b>	<b>System Architecture, Implementation and Results</b>	<b>56</b>
3.1	Introduction.....	56
3.2	System Architecture.....	56
3.2.1	Introduction.....	56
3.2.2	System Architecture.....	56
3.2.2.1	Back-end.....	58
3.2.2.2	Front-end .....	70
3.3	Environments and Implementation.....	70
3.3.1	Environments and tools .....	70
3.3.1.1	Back-end Environments and tools .....	71
3.3.1.2	Front-end Environments and tools.....	75
3.3.2	Back-end development .....	76
3.3.2.1	Import libraries .....	76
3.3.2.2	Dataset loading.....	77
3.3.2.3	Dataset pre-processing.....	78
3.3.2.4	Model training .....	79
3.3.2.5	Model evaluation.....	80
3.3.3	Front-end development.....	80
3.4	Results and discussion.....	85
3.4.1	Results.....	85
3.4.2	Results discussion and comparison.....	88
3.4.3	Limitation.....	89
3.4.4	Future Work .....	89
3.5	Conclusion .....	89

# List of Tables

3.1	Image2Weather Dataset [115]. . . . .	59
3.2	RESIDE Dataset [119]. . . . .	62
3.3	Snow100k Dataset [107]. . . . .	63
3.4	New Dataset. . . . .	64
3.5	Final Dataset. . . . .	65
3.6	One Hot Encoding. . . . .	67
3.7	Weather conditions classification tests. . . . .	85
3.8	Comparison weather recognition works. . . . .	88



# List of Figures

1.1	Pixel square in image example [2] .....	14
1.2	Binary image [3] .....	14
1.3	Grayscale image [4] .....	14
1.4	Color image [3] .....	15
1.5	Human visual system [5].....	16
1.6	The human eye anatomy [6] .....	17
1.7	How human visualize color [7].....	18
1.8	Visible light spectrum [10].....	20
1.9	RGB color mode [13] .....	21
1.10	CMYK color mode [13].....	22
1.11	Human vision system [22].....	23
1.12	Computer vision system [22].....	23
1.13	Cancer detected/ not detected [21] .....	24
1.14	Traffic sign example [23].....	24
1.15	Object detection [16].....	25
1.16	Image creation [15].....	25
1.17	Features example [25].....	27
1.18	Hand-engineered features extraction .....	27
1.19	Automatic-engineered features extraction .....	28
2.1	Single-Layer Feedforward Networks [47] .....	33
2.2	Multi-Layer Feedforward Networks [48] .....	33
2.3	Supervised learning scheme [49] .....	34
2.4	Unsupervised learning scheme [49] .....	34
2.5	Reinforcement learning scheme [52] .....	35
2.6	Autoencoders architecture [59] .....	37
2.7	Deep Belief Network architecture [63] .....	38
2.8	GAN architecture [65].....	39

2.9	Simple RNN architecture [69].....	39
2.10	Simple LSTM architecture [69].....	40
2.11	Recursive neural networks architecture [72].....	41
2.12	CNN architecture example [90].....	41
2.13	CNN Input shape [77].....	42
2.14	Filter of size 3x3 [79].....	43
2.15	Convolution operation [80].....	43
2.16	Feature maps [82].....	44
2.17	ReLU activation function [86] .....	44
2.18	Applying ReLU after convolution operation [87].....	45
2.19	Convolution operation example with difference strides [83].....	45
2.20	Convolution operation example with padding=2 [15].....	46
2.21	Illustration of max pooling with a pooling area of size 2x2 and stride of 2 [88].....	46
2.22	Fully connected layer [89]. .....	47
2.23	Simple CNN architecture with main layers [75].....	47
2.24	LetNet architecture [91] .....	48
2.25	AlexNet architecture [91] .....	48
2.26	GoogLeNet architecture [92].....	49
2.27	VGG-16 architecture [91] .....	49
2.28	ResNet architecture [92].....	50
2.29	Cross-validation idea [96] .....	51
2.30	Dropout some neurons randomly with certain probability [97].....	51
2.31	L1 and L2 equations [98].....	52
3.1	System General Architecture .....	57
3.2	Back-end Architecture.....	58
3.3	Re-sampling dataset [116] .....	60
3.4	FRIDA dataset [117] .....	61

3.5	LIVE Image Defogging Database [118].....	61
3.6	Samples of RESIDE dataset [119].....	62
3.7	Samples of snow100k dataset [107] .....	63
3.8	Samples of the new dataset .....	64
3.9	Samples of the final dataset .....	65
3.10	Pre-processing pipeline.....	66
3.11	Normalization formula [120] .....	68
3.12	Confusion matrix example .....	69
3.13	Prediction example .....	70
3.14	Front-end system .....	70
3.15	Python logo .....	71
3.16	Pycharm logo .....	71
3.17	TensorFlow 2.0 logo .....	72
3.18	Keras logo .....	72
3.19	OpenCV logo .....	73
3.20	Matplotlib logo .....	73
3.21	Numpy logo .....	73
3.22	Scikit-learn logo .....	74
3.23	Flask logo .....	74
3.24	Google Colab Logo .....	75
3.25	JavaScript logo .....	75
3.26	HTML5 logo .....	75
3.27	CSS3 logo .....	76
3.28	Sublime text logo .....	76
3.29	Front-end architecture .....	81
3.30	App Page 1 .....	82

3.31 App Page 2A .....	82
3.32 App Page 2B .....	83
3.33 App Page 3.....	83
3.34 App Page 4.....	84
3.35 Accuracy and Loss graphs.....	87
3.36 Confusion matrix.....	87

# General Introduction

In the past decade human life has begun to be more connected to digital technologies and this association is developing daily, so why not? Every day we hear and see new technology. Digital technologies are permeating our lives; we hardly see a home or a child without a tech device. The success of digital systems depends on the progress made in computer vision which has been a subject of increasing interest and rigorous research for decades now, the purpose of computer vision is to program a computer to "understand" a scene or features in an image. Computer vision is considered as an eye of the game changer in technology revolution: artificial intelligence (AI).

Artificial intelligence now is touching most of the big new technologies; we can say that AI is leader for anything is new right now. With the great achievements feats by AI with computer vision technology the future of computer vision appears to be more promised than ever. Behind the scenes of the success of computer vision systems stand one of the most exciting subfields of AI namely deep learning which take control over computer vision new applications powered by one of deep neural networks: Convolutional neural networks (CNN) which show greater capabilities for image recognition, due to that CNN has become the mainstream deep model in computer vision.

Between the applications of computer vision we find image recognition. Weather recognition from a single image is a challenging subject to study. The challenge to solve this problem is that weather conditions are various in nature and there is an absence of discriminate features to distinguish from weather condition to another. Weather classification in images is a process to categorize images based on the conditions of weather; severe weather conditions will have a great impact on our daily lives, urban traffic and many other things due to these reasons an automatic recognition of weather condition system can help due to his important value in agriculture, traffic condition warning, driver assistant system, surveillance cameras, and other aspects.

In this project, we focus on weather conditions image classification using deep learning algorithm CNN, in order to make our system accurate on extract weather information from images. We propose some CNN architectures that detect the weather class from the image for five weather conditions: cloudy, foggy, rainy, snowy, and sunny.

Beside a general introduction and a conclusion, our master thesis is organized in 3 chapters. In the first chapter, we present the subject of image classification in computer vision. For the second chapter, we yield the concept of deep learning and its models and applications. Finally, the third chapter presents our system architecture, our CNN architecture implementation and the discussion of the achieved results.

# Chapter 1

## Image Classification In Computer Vision

### 1.1 Introduction

Today, the field of computer vision flourishing so fast, and taken great leaps thanks to advances of deep learning and artificial neural networks. These advances have paved the way for boosting the use of computer vision in existing domains and introducing it to new ones. In many cases, computer vision algorithms have become a very important component of the applications we use every day.

This chapter discusses how computer vision inspires from our visual system, also talking about one of its application which is image classification task, and the relationship between computer vision and image processing.

### 1.2 Image basics and color terminology

#### 1.2.1 Digital image

To understand what's a digital image is, we have to first realize that what we see when we look at a "digital image" is actually a physical image reconstructed from a digital image. The digital image itself is really a data structure within the computer, containing a number or code for each pixel in the image. This code determines the color of that pixel. Each pixel can be thought of as a discrete sample of a continuous real image. It is helpful to think about the common ways that a digital image is created. Some of the main ways are via a digital camera, a page or slide scanner, a 3D rendering program, or a paint or drawing package. The simplest process to understand is the one used by the digital camera [1].

##### 1.2.1.1 What is a pixel ?

A pixel, or picture element is a physical point in a raster image, or the smallest addressable element in an all points addressable display device; so it is the smallest controllable element of a picture represented on the screen.

Each pixel is a sample of an original image; more samples typically provide more accurate representations of the original. The intensity of each pixel is variable. In color imaging systems, a color is typically represented by three or four component intensities such as red, green, and blue, or cyan, magenta, yellow, and black [2].

In a digital image, all the coordinates on 2-d function and the corresponding values are finite. Each value available in every location is considered as a pixel. In other words, a pixel is the smallest part of an image. So a digital image can be thought as 2-d array of pixels

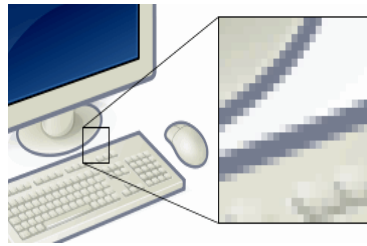


Figure 1.1: Pixel square in image example [2]

### 1.2.1.2 Image types

#### A- Binary image:

A binary image has only two possible gray values or intensities 0 and 255, there are no intermediate values. Binary images are used as masks for indicating the pixels of interest in many image processing tasks. Figure 1.2 gives an example of binary image [3].



Figure 1.2: Binary image [3]

#### B- Grayscale image:

Grayscale image has range of values from 0 to 255. Each pixel location can have any value between 0 and 255. If you watch old films around the 1950s, you are watching grayscale images (films are nothing but videos which are collection of individual images in proper sequence) [3].



Figure 1.3: Grayscale image [4]

## C- Color image:

Both binary image and grayscale image are 2-dimensional arrays, where at every location, you have one value to represent the pixel. Remember to represent a color image; we need more than one value for each pixel. But how many values we need to represent a color? Typically you need 3 values for each pixel to represent any color. This came from the idea that any color can be formed by combining 3 basic colors Red, Blue and Green. Ex: you get yellow by mixing red and green, violet can be formed by combining red and blue etc., This is actually called RGB color space. There are many other ways to create color images which we will discuss in further. Figure 1.4 is an example of a color image [3].



Figure 1.4: Color image [3]

## 1.2.2 Human visual system

### 1.2.2.1 Description of the visual system

The human visual system constantly adapts to different luminance levels when viewing natural scenes. We present a model of the visual adaptation, which supports displaying the high dynamic range content on the low dynamic range displays. In this solution, an eye tracker captures the location of the observer's gaze. Temporary adaptation luminance is then determined as the impact of the light area surrounding the gaze point. Finally, the high dynamic range video frame is tone mapped and displayed on the screen in real time. We use a model of local adaptation, which predicts how the adaptation signal is integrated in the retina, including both time-course and spacial extent of the visual adaptation. The applied tone mapping technique uses a global compression curve, the shape of which is adapted to the local luminance value. This approach mimics a natural phenomenon of the visual adaptation occurring in human eyes [5].

The human visual system comprises three main parts: the eye, the lateral geniculate nucleus (LGN), and the part of the cortex brain that processes the visual information – the visual cortex. Fig. 1.5 shows a schematic structure of the visual pathway [5].



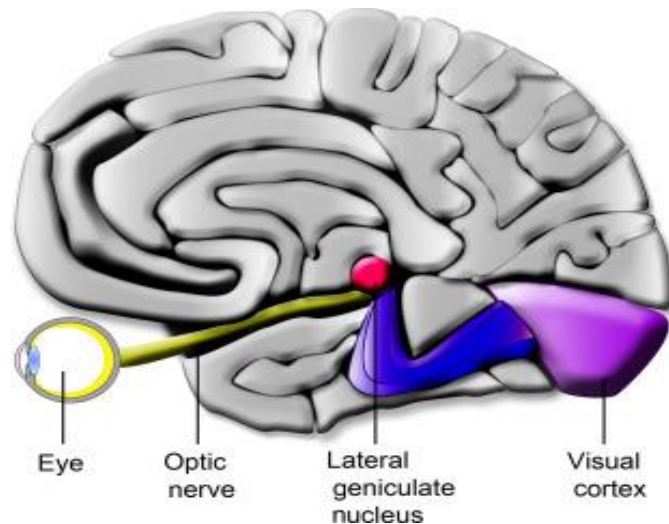


Figure 1.5: Human visual system [5]

### 1.2.2.2 The human eye anatomy

Figure 1.6 illustrates a cross sectional view of the human eye with various ocular structures indicated. Basically, the human eye functions in a sequential manner. Firstly, the lights perceived will pass through the cornea and pupil to the lens. Secondly, the lens will focus the lights onto the retina. Thirdly, the captured light is converted into signals. Finally the signals are transmitted to the brain, through the optic nerve, where the signals are perceived as images. The retina is a thin layer located on the inside wall at the back of human eye, between the choroid and the vitreous body.

- The *retina* is composed of photoreceptors (rods and cones) and neural tissues that receives light, converts it into neural signals, and sends the signals to the optic nerve.
- The *optic* disc is where the retinal blood vessels converge and communicate perceived signals to the brain through the optic nerve. Its horizontal and vertical “diameters” are approximately 1.7 mm and 1.9 mm respectively.
- The *macula* is a small area at the centre of the retina where high concentration of photoreceptors can be found. In a healthy retinal image, the macula appears as a darkened circular region [6].

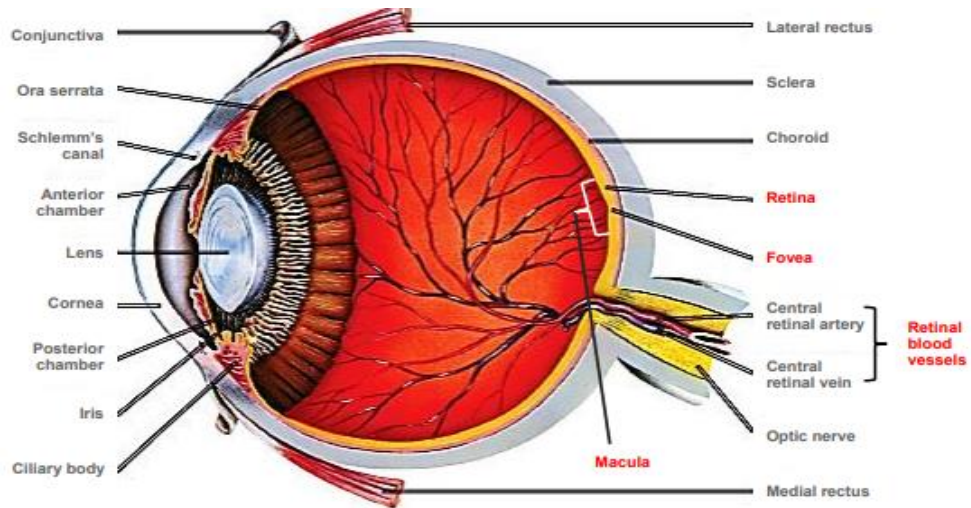


Figure 1.6: The human eye anatomy [6]

### 1.2.2.3 How do we see color?

When light hits an object, the object absorbs some of the light and reflects the rest of it. Which wavelengths are reflected or absorbed depends on the properties of the object. For a ripe banana, wavelengths of about 570 to 580 nanometers bounce back. These are the wavelengths of yellow light. When you look at a banana, the wavelengths of reflected light determine what color you see. The light waves reflect off the banana's peel and hit the light-sensitive retina at the back of your eye. That's where cones come in.

Cones are one type of photoreceptor, the tiny cells in the retina that respond to light. Most of us have 6 to 7 million cones, and almost all of them are concentrated on a 0.3 millimeter spot on the retina called the fovea centralis [7].

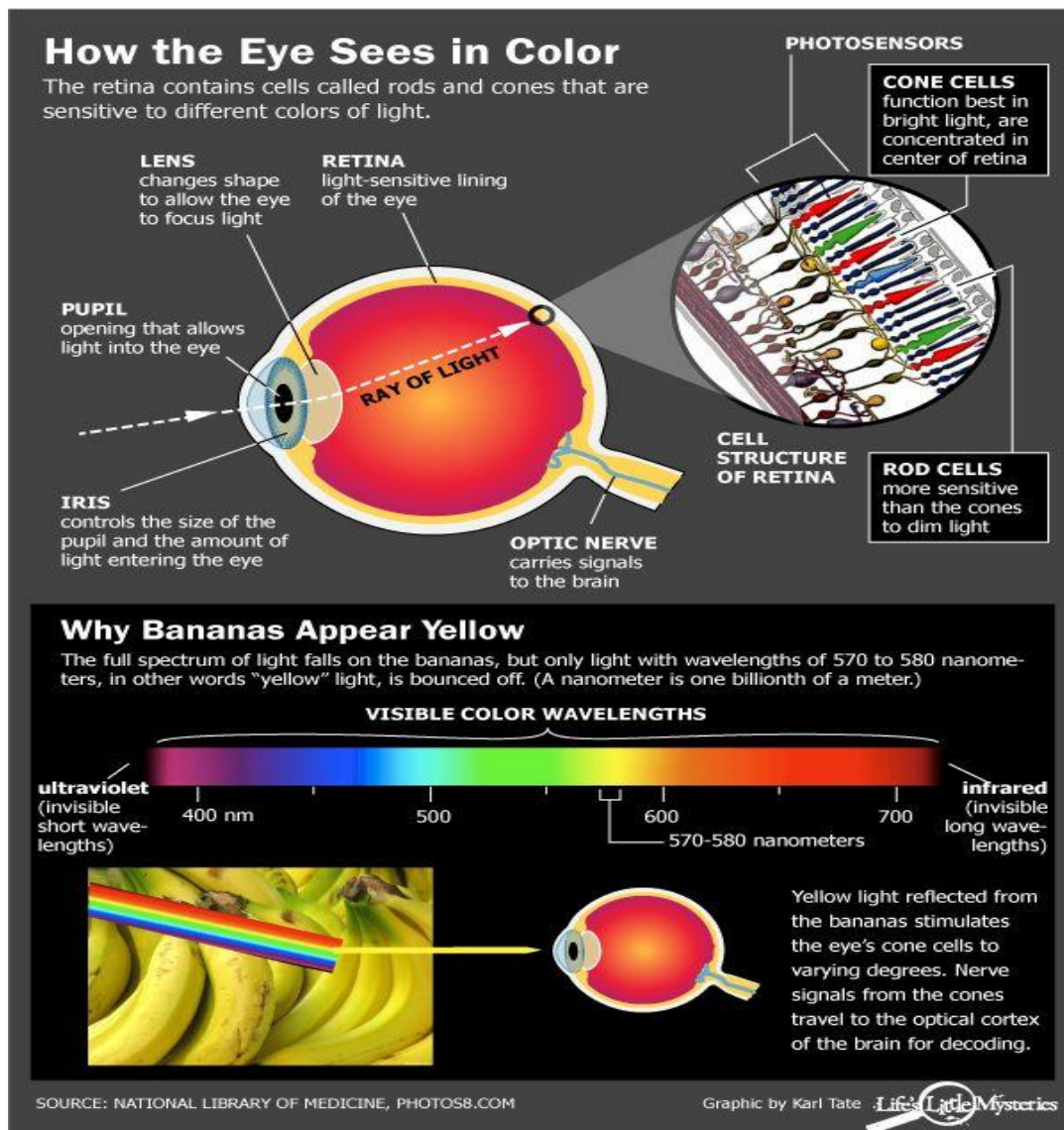


Figure 1.7: How human visualize color [7]

Not all of these cones are alike. About 64 percent of them respond most strongly to red light, while about a third are set off the most by green light. Another 2 percent respond strongest to blue light. When light from the banana hits the cones, it stimulates them to varying degrees. The resulting signal is zapped along the optic nerve to the visual cortex of the brain, which processes the information and returns with a color: yellow.

Humans, with our three cone types, are better at discerning color than most mammals, but plenty of animals beat us out in the color vision department. Many birds and fish have four types of cones, enabling them to see ultraviolet light, or light with wavelengths shorter than what the human eye can perceive.

Some insects can also see in ultraviolet, which may help them see patterns on flowers that are completely invisible to us. To a bumblebee, those roses may not be so red after all [7].

## **1.2.3 A quick look about colors**

### **1.2.3.1 What is a color?**

Color is the characteristic of human visual perception described through color categories, with names such as red, orange, yellow, green, blue, or purple. This perception of color derives from the stimulation of cone cells in the human eye by electromagnetic radiation in the visible spectrum. Color categories and physical specifications of color are associated with objects through the wavelength of the light that is reflected from them. This reflection is governed by the object's physical properties such as light absorption, emission spectra, etc [8].

### **1.2.3.2 Color vision**

Color vision is an ability of animals to perceive differences between light composed of different wavelengths (i.e., different spectral power distributions) independently of light intensity. Color perception is a part of the larger visual system and is mediated by a complex process between neurons that begins with differential stimulation of different types of photoreceptors by light entering the eye. Those photoreceptors then emit outputs that are then propagated through many layers of neurons and then ultimately to the brain. Color vision is found in many animals and is mediated by similar underlying mechanisms with common types of biological molecules and a complex history of evolution in different animal taxa. In primates, color vision may have evolved under selective pressure for a variety of visual tasks including the foraging for nutritious young leaves, ripe fruit, and flowers, as well as detecting predator camouflage and emotional states in other primates [9].

### **1.2.3.3 What is the visible light spectrum?**

The visible spectrum is the portion of the electromagnetic spectrum that is visible to the human eye. Electromagnetic radiation in this range of wavelengths is called visible light or simply light. A typical human eye will respond to wavelengths from about 380 to 740 nanometers. In terms of frequency, this corresponds to a band in the vicinity of 430–770 THz.

The spectrum does not contain all the colors that the human eyes and brain can distinguish. Unsaturated colors such as pink, purple variations like magenta, for example, are absent because they can only be made from a mix of multiple wavelengths. Colors containing only one wavelength are also called pure colors or spectral colors [10].

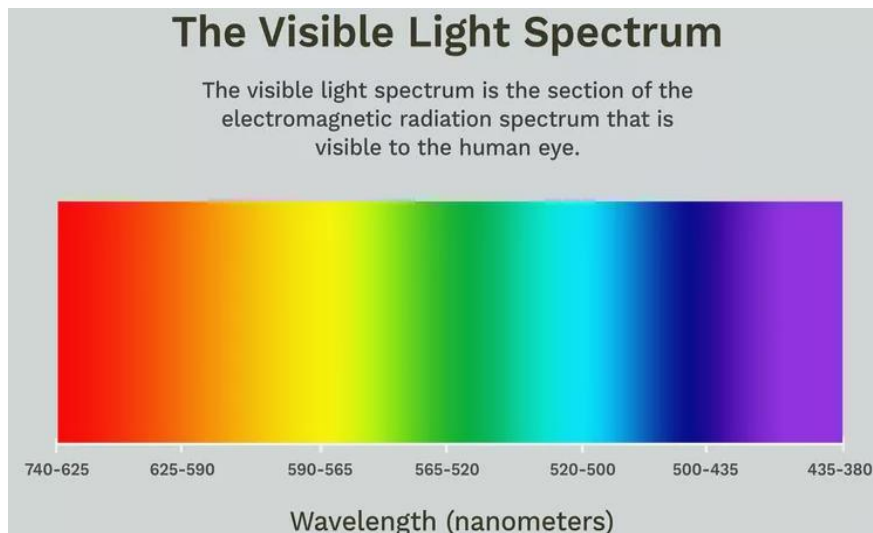


Figure 1.8: Visible light spectrum [10]

The wavelengths of visible light are:

- **Violet:** 380–450 nm
- **Blue:** 450–495 nm
- **Green:** 495–570 nm
- **Yellow:** 570–590 nm
- **Orange:** 590–620 nm
- **Red:** 620–750 nm

#### **A- Colors human see that aren't on the spectrum**

The visible spectrum does not encompass all the colors humans perceive because the brain also perceives unsaturated colors (e.g., pink is an unsaturated form of red) and colors that are a mixture of wavelengths (e.g., magenta). Mixing colors on a palette produces tints and hues not seen as spectral colors [11].

#### **B- Colors only animals can see**

Just because humans can't see beyond the visible spectrum doesn't mean animals are similarly restricted. Bees and other insects can see ultraviolet light, which is commonly reflected by flowers. Birds can see into the ultraviolet range (300–400 nm) and have plumage visible in UV.

Humans see further into the red range than most animals. Bees can see color up to about 590 nm, which is just before orange starts. Birds can see red, but not as far toward the infrared range as humans.

Some people believe the goldfish is the only animal that can see both infrared and ultraviolet light, but this notion is incorrect. Goldfish cannot see infrared light [11].

### 1.2.3.4 Additive, subtractive models

#### A- RGB color model

Media that transmit light (such as television) use additive color mixing with primary colors of red, green, and blue, each of which stimulates one of the three types of the eye's color receptors with as little stimulation as possible of the other two. This is called "RGB" color space. Mixtures of light of these primary colors cover a large part of the human color space and thus produce a large part of human color experiences. This is why color television sets or color computer monitors need only produce mixtures of red, green and blue light. See Additive color.

Other primary colors could in principle be used, but with red, green and blue the largest portion of the human color space can be captured. Unfortunately there is no exact consensus as to what loci in the chromaticity diagram the red, green, and blue colors should have, so the same RGB values can give rise to slightly different colors on different screens [12].

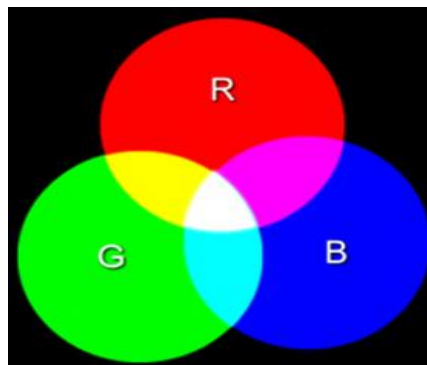


Figure 1.9: RGB color mode [13]

#### B- CMYK color model

It is possible to achieve a large range of colors seen by humans by combining cyan, magenta, and yellow transparent dyes/inks on a white substrate. These are the subtractive primary colors. Often a fourth ink, black, is added to improve reproduction of some dark colors. This is called the "CMY" or "CMYK" color space.

The cyan ink absorbs red light but transmits green and blue, the magenta ink absorbs green light but transmits red and blue, and the yellow ink absorbs blue light but transmits red and green. The white substrate reflects the transmitted light back to the viewer. Because in practice the CMY inks suitable for printing also reflect a little bit of color, making a deep and neutral black impossible, the K (black ink) component, usually printed last, is needed to compensate for their deficiencies. Use of a separate black ink is also economically driven when a lot of black content is expected, e.g. in text media, to reduce simultaneous use of the three colored inks. The dyes used in traditional color photographic prints and slides are much more perfectly transparent, so a K component is normally not needed or used in those media [12].

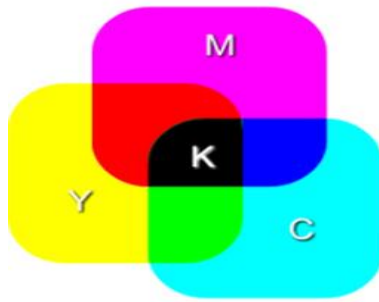


Figure 1.10: CMYK color mode [13]

## 1.3 Computer vision

### 1.3.1 Computer vision what does it means?

We are as human using our eyes and brains to understand the real-world or visualize things, scenes around us, we can understand images using our visual system and tell if the image is for a plane or boat (the classification task) or detect, localize objects in images, videos, etc.

On the other hand, computer vision search to give similar tasks to machines, maybe better in the near future. In other word, computer vision search to create techniques able to mimic the human vision using machine learning algorithms [14].

Computer vision system tries to perceive and understand the world through images and videos by constructing a physical model of the world so that the AI system can then take the appropriate actions [15].

### 1.3.2 Computer vision and human vision

#### A- Human vision

As a presented in Section 1.2.2, human vision system consist of:

- 1) A sensor or an eye to capture the image
- 2) A brain to process and interpret this image

The system then outputs a prediction of the image components based on what we understand and the information extracted from the image [15].

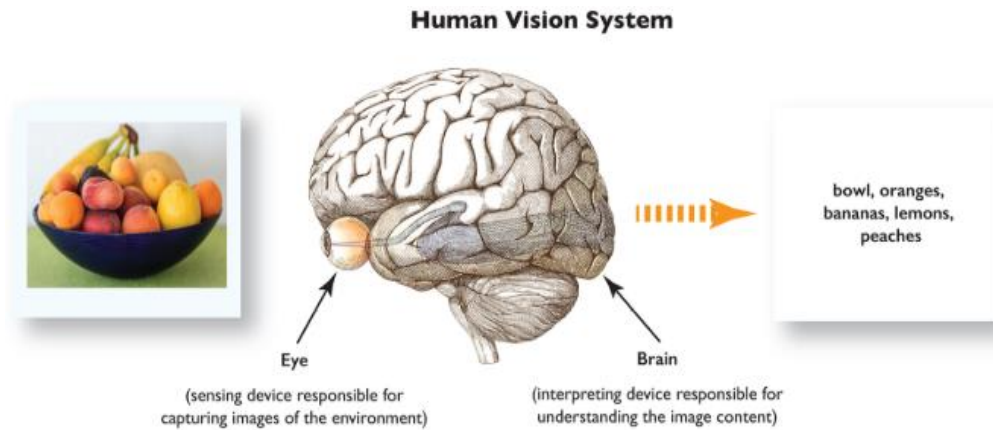


Figure 1.11: Human vision system [22]

### B- Computer vision system

In the last few years the researchers have reach an advance step in computer vision tasks by extending the human vision ability to machines. So, in order to mimic the human vision system, we need the same two main components:

- 1) A sensing device to mimic the function of the eye.
- 2) A powerful algorithm (such as machine learning, deep learning algorithms) to mimic the brain function in interpreting and understanding the image content [15].



Figure 1.12: Computer vision system [22]

#### 1.3.1.2 Applications

Thanks to the evolution in computational power and the amount of data available, Artificial Intelligence (AI) and deep learning have managed to achieve human performance on many complex visual perception tasks like image search and captioning, image and video classification, and object detection, etc. Moreover, deep neural networks are not restricted to computer vision tasks: they are also successful at Natural Language Processing (NLP) and Voice User Interface (VUI) tasks [15], etc.



## A- Image classification

Image classification involves assigning a label to an image. In recent years deep learning play remarkable role in the progress of computer vision applications, one of the successful algorithms is Convolution Neural Networks (CNNs) which is used in many different classification tasks [20]. Image classification examples: handwritten digit, breast cancer, facial expressions...

- Lung cancer diagnosis

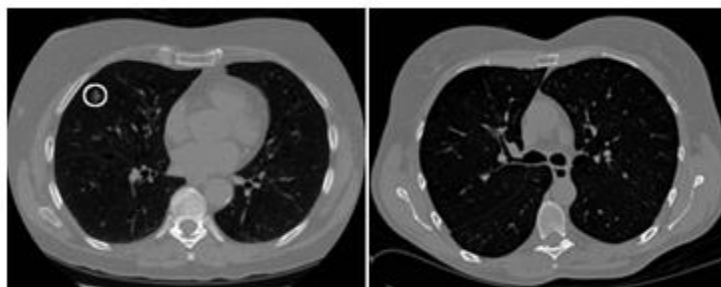


Figure 1.13: Cancer detected/ not detected [21]

-Traffic sign recognition



Figure 1.14: Traffic sign example [23]

## B- Object detection

If we are aiming to reach human levels of understanding, we have to add complexity to the system to be able to recognize multiple objects and their locations in an image. Algorithms such as YOLO (you only look once) [30], Faster R-CNN [31], Single Shoot Detector (SSD) [32], not only classify images, but can locate and detect each object in images that contain multiple objects. These deep learning systems can look at an image, break it up into smaller regions, and label each region with a class so that a variable number of objects in a given image can be

localized and labeled. You can imagine that such a task is a basic prerequisite for applications like autonomous systems [15].

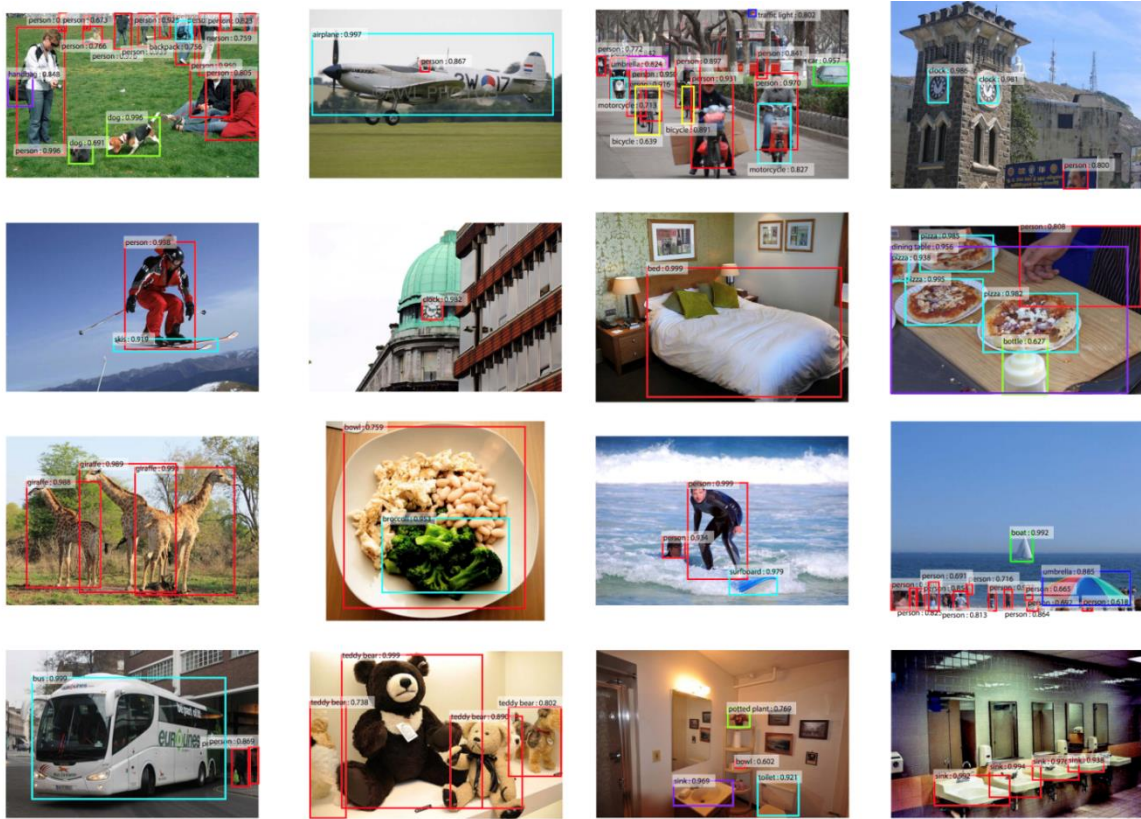


Figure 1.15: Object detection [16]

### C- Create images

Generative Adversarial Network is a new deep learning model invented by Ian Goodfellow in 2014, that can imagine new things. GANs, are sophisticated deep learning models that are able to generate stunningly accurate synthesized images of objects, people, and places among other things. If you give them a new set of images, they can make entirely new images that are realistic and have never been seen before. These photos are never seen before and totally imaginary [15]

This small blue bird has a short pointy beak and brown on its wings



This bird is completely red with black wings and pointy beak



Figure 1.16: Image creation [15]

### **1.3.2 Computer vision vs. image processing**

In image processing, an image is "processed", that is, transformations are applied to an input image and an output image is returned. The transformations can for example be "smoothing", "sharpening", "contrasting" and "stretching". The transformation used depends on the context and issue to be solved [17].

Unlike image processing, which is mainly focused on processing raw images without giving any knowledge feedback on them [14], computers are made to gain high-level understanding from the input digital images or videos with the purpose of automating tasks that the human visual system can do. It uses many techniques and Image Processing is just one of them [17], we can consider image processing as preprocessing step for computer vision algorithms.

The main difference between them is the goals. For example, if the goal is to enhance an image for later use, then this may be called image processing. If the goal is to emulate human vision, like image classification, object recognition, automatic driving, then it may be called computer vision [17].

## **1.4 Image classification**

### **1.4.1 Definition**

Classification consists assigning a class to an image by using a classification system. To classify the data into different classes or labels, the classification system needs a well understanding the relationship between the data and the class to be assigned. what we mean by well understanding is that before the classification task the real-world data may be complex ,incomplete or have information that's out of our interest so the system need to simplify or clarify the data before trying to classify it so the task be much easier to distinct between classes and give us a good results.

### **1.4.2 Features and classifiers concepts**

We can tell that the efficiency and robustness, accuracy of a computer vision application dependent on the quality of the data features (images in our case) and the classifiers [18].

An effective feature extractor would make the job of the classifier much easier to give better performance. On the contrary, simple features extractors need a "perfect" classifier to perform well [19]. For now this considered as impossible case, so we need an effective feature extractor to achieve reasonable performance [24].

#### **A- Features**

In computer vision, a feature is a measurable piece of data in your image that is unique to this specific object. It may distinct color in an image or a specific shape such as a line, edge, or an

image segment [25]. The combination of  $n$  features can be represented as a  $n$ -dimensional vector, called a feature vector. The quality of a feature vector is dependent on its ability to discriminate image samples from different classes [26].

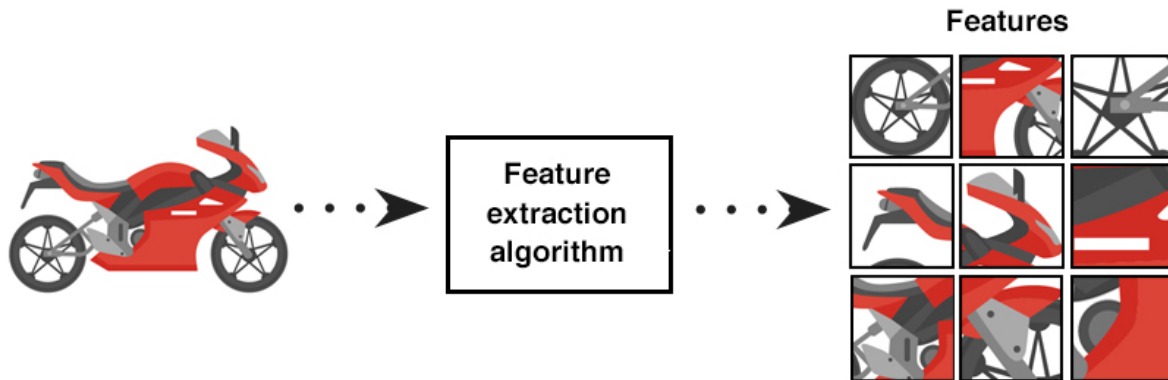


Figure 1.17: Features example [25]

## B- Classifiers

The task of the classifier is to use the feature vector to assign an image or region of interest to a class. The difficulty of the classification task depends on the variability in the feature values of images from the same category, relative to the difference between feature values of images from different categories [27]. In machine learning there are a classifier algorithms such as support vector machine (SVM) [33], random forest (RF) [34], Decision Tree [35], Neural Networks etc.

### 1.4.2.1 Features extraction (hand-engineered)

In traditional machine learning problems, we spend a good amount of time in manual features selection and engineering [15]. To quantify the contents of an image based on a particular component of the image we want to encode (i.e., shape, color, texture). Given these features, we then proceed to train our classifier and evaluate it [28]. Some of the hand-crafted feature sets are:

- Histogram of Oriented Gradients (HOG)
- Scale-Invariant Feature Transform (SIFT)
- Speeded Up Robust Feature (SURF)

### MACHINE LEARNING



Figure 1.18: Hand-engineered features extraction

### 1.4.2.1 Features extraction (automatic -engineered)

Deep learning skip features extraction step, the network extracts features automatically [15].

You just feed the raw image to the network. The network then learns filters inside its hidden layers that can be used to discriminate among object classes [28].

Neural networks play a role of feature extractors + classifiers that are end-to-end trainable as opposed to traditional ML models that use hand-crafted features [15].



Figure 1.19: Automatic-engineered features extraction

## 1.4.2 Classification process

### 1.4.3.1 Images collection

The first component of building a deep learning network is to gather our initial dataset. We need the images themselves as well as the labels associated with each image (supervised learning). These labels should come from a finite set of categories [28].

### 1.4.3.2 Images pre-processing

A lot of time waste in cleaning up and preparing the data before building learning model. The aim of digital image processing is to improve the image data (features) by suppressing unwanted distortions and/or enhancement of some important image features so that our AI-computer vision models can benefit from this improved data to work on [29]. Also make the data easier to analyze and process computationally.

### 1.4.3.3 Training

The network tries to learn how to distinct between each class by extraction features (as described in 1.4.2). When the model makes a mistake, it learns by updating parameters such weights and biases, and improves itself.

In fact the training phase needs to split into 3 sets (training, validation, and test). We will discuss this option in chapter 3.

### 1.4.3.4 Classifying

Now it is time to feed the extracted features vector to the classifier to output a class label for the images. As described in Section 1.4.2, the classification task is done by either one of these types:

- 1) Traditional machine learning algorithms like SVMs and Random Forest.
- 2) Fully connected layer (neural network) such as used in CNNs.

#### **1.4.3.5 Evaluation**

Finally we have the model ready to predict new images never seen before. The model predictions are compared to the ground-truth labels from our testing set. The ground-truth labels represent what the image category actually is. From there, we can compute the number of predictions our classifier got correct and compute aggregate reports such as precision, recall, and f-measure, which are used to quantify the performance of our network as a whole.

## **1.5 weather conditions image Classification**

### **1.5.1 Introduction**

Multi-class weather classification is valuable technique which has an effect for many computer vision applications, such as video surveillance, ADAS (Advanced driver-assistance systems), drone applications, and object recognition. However, we are in front of challenging task due to the multifariousness of weather conditions and inadequacy of discriminate features that help to differentiate among various weather conditions.

### **1.5.2 Description**

For any pattern recognition problem, it is important to select proper features. We always implement image classification by selecting interesting points as features or detecting the object emerging in the scene. However, weather classification from images is different from general image classification. It is impractical to use the traditional features for our problem because there can be the same object and interesting points under different weather conditions. Hence, it is not proper for applying the same kind of features as general image classification [41].

### **1.5.3 Features examples**

#### **A- Brightness**

Brightness is one of the most significant pixel characteristics. It also explains weather images nicely and can take different values for different type of weather conditions. Also, human eyes and brain use brightness metric in a scene to recognize weather type. Psychology unintentionally selects bright scenes as sunny or snowy mostly. All these reasons make it a good estimator to recognize weather images and distinguish weather types from each other [42].

## **B- Haze factor**

Cloudy or foggy weather may come with haze. We learned that the lowest and highest value in color channels tends to be the same value of atmospheric light. We need to find a haze function for detecting cloudy and foggy images.

For this purpose, a reference work introduces a model to calculate haze factor in images [43]. It simply uses calculated contrast value above and calculated darkness value in a similar way to our contrast metric then measure a haze factor between (0, 1) by using contrast and darkness metric's combination [42].

## **C- Sharpness**

Sharpness determines the amount of detail an imaging system can reproduce. "Classification of Weather Situations on Single Color Images" paper [44] says "Clearly distinguishable objects under friendly weather conditions are expected to have sharp edges with large contrast differences" and suggest sharpness usage to distinguish weather types. We also observed that sunny, cloudy and rainy weather images are sharper where snowy and foggy weather images are less sharper. Since light and weather kind of conditions affect how to look (more/less sharper) a taken picture, sharpness value in given image can be used as an image descriptor for sure [42].

## **1.5.4 Possible Applications**

### **A- ADAS(Advanced Driver Assistance System)**

Recently, the development of autonomous vehicles and intelligent driver assistance systems has drawn significant amount of attention from the general public. One of the most critical issues in the development of autonomous vehicles and driver assistance systems is their poor performance under adverse weather conditions.

Vehicle on-board cameras provide drivers with crucial visual signals and information for safe driving. These systems provide a good service under clear day condition. However, such systems should not degrade in severe weather such as (snow, rain and fog) and the camera system should continue to provide helpful information under these conditions, especially as the driver is under an increased workload. Bad weather reduces the scene contrast and visibility.

For example in snowy condition if there is moisture around a camera below freezing point, the frost can cover the camera's lens and deny the viewer seeing any activities besides the crystalline patterns of the snow. Optically, raindrops on the lens can change the focus of camera. As a result, part of the image affected by raindrops will be out of focus and blurred. These effects can lead to the failure of image processing such as pattern recognition [36].

With the advancement in technology and emergence of a new field, intelligent transportation, automated determination of weather condition has become more relevant. Present systems either deploy an array of sensors or use an in-vehicle camera to predict weather conditions. These

solutions have resulted in incremental cost and limited scope. To ensure smooth operation of all transportation services in all-weather conditions, a reliable detection system is necessary to classify weather in wild. The challenges involved in solving this problem is that weather conditions are diverse in nature and there is an absence of discriminate features among various weather conditions [37].

## **B- Drones**

If you want to make money flying drones commercially or even if you are having fun just flying them around in an open field; you had better understand how weather can affect the operation of your flying machine. It isn't a "little thing." Weather is the number one reason people have accidents with their drones [38].

Some pilots don't consider fog to be an obstacle for flying a drone. Yet still, it doesn't mean that fog is less important than other weather conditions. As a rule, aircraft don't handle moisture well. To predict fog, you need to look at the visibility index. Besides, if the weather is foggy, you won't see your drone in the sky [39].

If the drone is not water resistant, or it did not run prelaunch tests, it would be better to avoid launching it in heavy pouring rain. In order to be able to fly the drone and relatively control it, it would be safer to use it in case the rain is very mild, however it is still not encouraged to do so. The problem with colder rainy environments is that it negatively affects the lithium polymer battery in the drone. It will also drop it below the critical level because cold weather will require the drone to have more motor power to lift, so it may harm the electronic speed control and the motors. Other impacts include camera and damages that can occur from raindrops [40].

Weather is always changing, and thus it is on the UAV operator to ensure they know the current and forecasted weather conditions before they head out for their day of flying! [38].

## **1.6 Conclusion**

Image classification plays an important role in computer vision systems that need recognize and detect different objects quick and accurate. Self-driving cars, as example, use computer vision and image recognition to recognize different objects on the road such as signs, vehicles, and walking people.

The next chapter, we will see the main reason that helped computer vision applications to progress widely in the last few years which is deep learning.



# Chapter 2

## Deep Learning

### 2.1 Introduction

We can agree that the recent successes of computer vision applications specifically image classification; object detection and facial recognition are due to the fast progress of deep learning techniques under the umbrella of machine learning.

In this chapter, we will discuss deep learning approach and how impact in the progress of computer vision applications with one of most popular architecture used for image classification which is convolutional neural networks.

### 2.2 Introduction to neural networks

#### 2.2.1 Definition

A neural network is a simplification of our most powerful tool, the brain. It uses neurons that are all connected to each other through weights. The neurons are given some numerical input and are multiplied by the weights. The weights are the heart of the neural network, and by changing them to specific numerical values, we can process any input and get a desired output [45].

#### 2.2.2 Principal architectures

Neural net architecture describe how the way the neurons are connected, and the connection patterns between layers, learning methods, and how network transform the information from its input layer to the output layer.

There exist types of architectures such as:

- Single-Layer Feedforward Networks.
- Multilayer Feedforward Networks.
- Convolutional Networks.
- Recurrent Networks.

##### 2.2.2.1 Single-Layer Feedforward Networks

There are two layers in this structure:

- Input layer
- Output layer (computation nodes)

Input signals are passed on to the output layer via the weights and the neurons in the output layer compute the output signals [46]. (As shown in the figure 2.1).

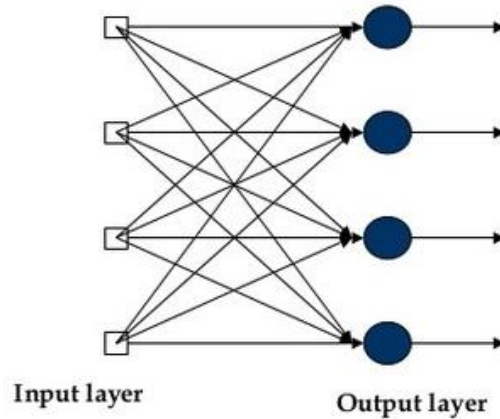


Figure 2.1: Single-Layer Feedforward Networks [47]

### 2.2.2.2 Multi-Layer Feedforward Networks

In this structure we can see an additional layer called hidden layer it can be one or more hidden layers. Hidden layers help the network to extract higher informations from its inputs. (As shown in the figure 2.2).

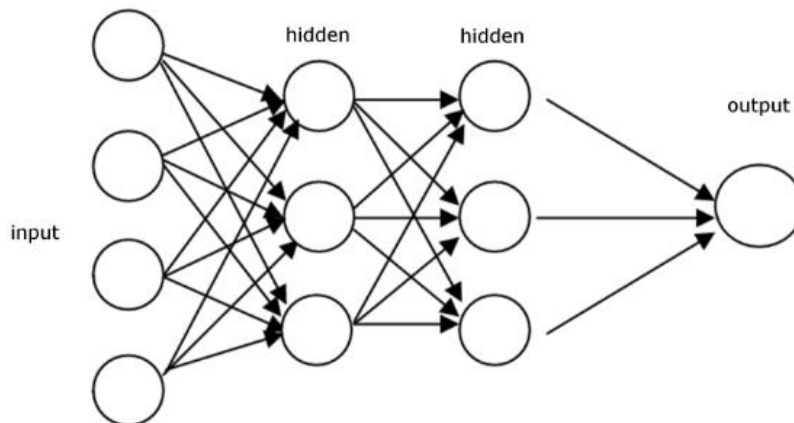


Figure 2.2: Multi-Layer Feedforward Networks [48]

In the next section we are going to discuss convolutional, and recurrent networks

### 2.2.3 Learning process (types)

As humans there are different ways to learn from our environments, the same with neural networks. We may categorize the learning processes to classes such as: supervised learning, unsupervised learning, reinforcement learning, semi-supervised learning.

#### 2.2.3.1 Supervised learning

Supervised learning describes a scenario of "teach by example". Gain the experience from training data (labeled data) to apply to testing data (unlabeled data). In other word, the goal of this type is to develop a procedure, a rule or a pattern that classifies unseen examples by analyzing the labeled examples. There are two main types of supervised learning problems:

- Classification: Supervised learning problem that involves predicting a class label.
- Regression: Supervised learning problem that involves predicting a numerical label.

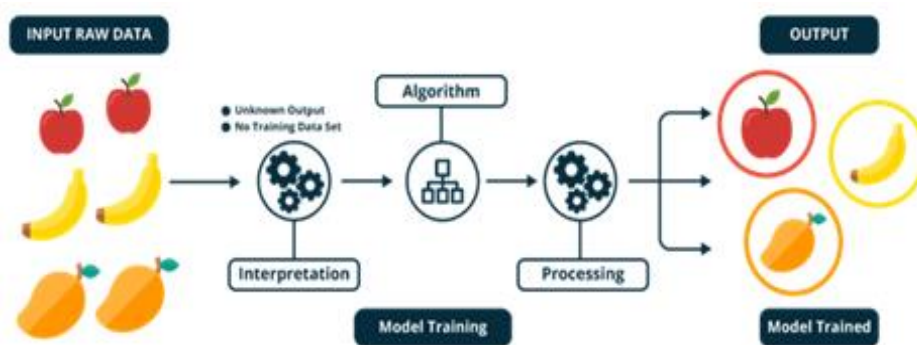


Figure 2.3: Supervised learning scheme [49]

#### 2.2.3.2 Unsupervised learning

This type of learning is the opposite of the first one, as a start we don't have a labeled data. The aim of the algorithm here is to group or label data by extract relationships, patterns in data, unsupervised learning operates upon only the input data without outputs or target variables. There are problems such as:

- Clustering: unsupervised learning problem that involves finding groups in data.

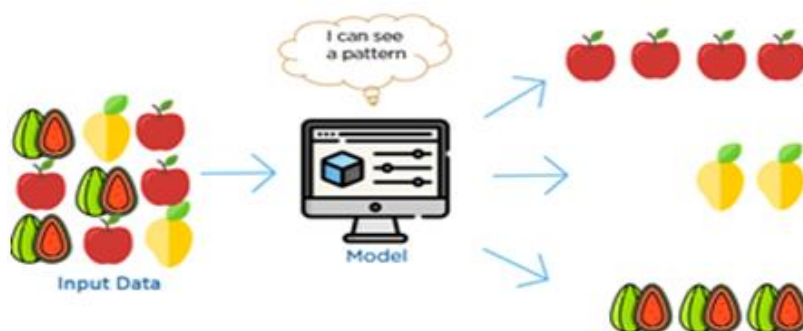


Figure 2.4: Unsupervised learning scheme [49]

### 2.2.3.3 Reinforcement learning

Reinforcement learning describes a class of problems where an agent operates in an environment and must learn to operate using feedback [50].

Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them [51].

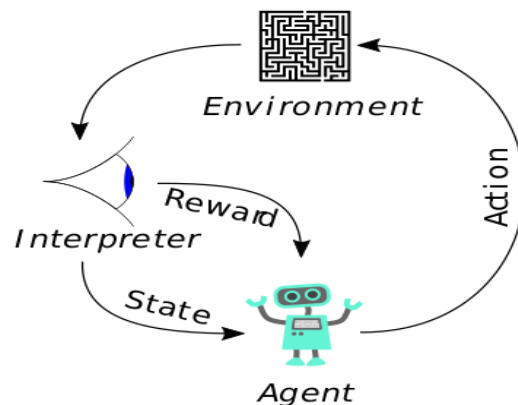


Figure 2.5: Reinforcement learning scheme [52]

In Reinforcement learning, the agent is one who takes decisions based on the rewards and punishments. Consider an example of a batsman in cricket. He tries to hit the ball if he misses he gets a negative point. If he hits the ball then he gets a reward. So, from these positive and negative experiences, he will understand how to play that particular ball. In this example, the batsman is an agent [53].

### 2.2.3.4 Semi-supervised learning

Semi-supervised learning is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data [54].

In semi-supervised learning we are given a few labeled examples and must make what we can of a large collection of unlabeled examples. Even the labels themselves may not be the oracular truths that we hope for [55].

The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. Semi-supervised learning is of great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data are scarce or expensive. Semi-supervised learning also shows potential as a quantitative tool to understand human category learning, where most of the input is self-evidently unlabeled [54].

## 2.3 Deep learning

### 2.3.1 Introduction

Everyone has an AI application in his hands, whether he understand it or not. We can tell that the latest successes of AI due to the progress of two very popular concepts machine learning and deep learning. But in the recent period, Deep learning take more intention than machine learning due to its success in giving a remarkable accuracy for training models with huge amount of data.

### 2.3.2 Definition

Deep learning is a subset of machine learning that imitates the workings of the human brain in processing data and creating patterns for use in decision making [56]. DL achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones [57].

We use deep learning when:

- Deep Learning really shines when it comes to complex problems such as image classification, natural language processing, and speech recognition.
- Deep Learning performs better than other techniques if the data size is large. But with small data size, traditional Machine Learning algorithms are preferable.
- Deep Learning techniques need to have high end infrastructure to train in reasonable time.
- When there is lack of domain understanding for feature introspection, Deep Learning techniques outshines others as you have to worry less about feature engineering.

### 2.3.3 The deep learning revolution

What happened in the past few years that make deep learning so shining? The two main reasons appear to be availability of massive labeled data sets and GPU computing. Here are many factors that impacted in the deep learning revolution:

- Appearance of large, high-quality labeled datasets - Data along with GPUs probably explains most of the improvements we've seen.
- Parallel computing with GPUs, The transition from CPU-based training to GPU-based has resulted in massive speed ups for these models.
- Improved architectures such as Resnet-50, VGG-16, and Alex-Net networks keep the gradients flowing smoothly, and let us increase the depth and flexibility of the network.
- For deep learning development, frameworks like tensorflow, keras, and pytorch provide high-level of abstraction. They let you focus on your model structure without having to

worry about dealing with the nitty-gritty details of implementing algorithms. they take care of the details behind the scenes.

- New regularization techniques like dropout, batch normalization, and data-augmentation allow us to train larger and larger networks without (or with less) overfitting [57].

## 2.3.4 Deep learning models

The last years gave us deep learning architectures, which greatly expanded the number and type of problems neural networks can address. This section discuss seven models of deep learning architectures.

### 2.3.4.1 Autoencoders

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs [58]. Autoencoders compress the input into a lower-dimensional code and then reconstruct the output from this representation. An autoencoder consists of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code. As shown in figure 2.5

To build an autoencoder we need 3 things: an encoding method, decoding method, and a loss function to compare the output with the target [59].

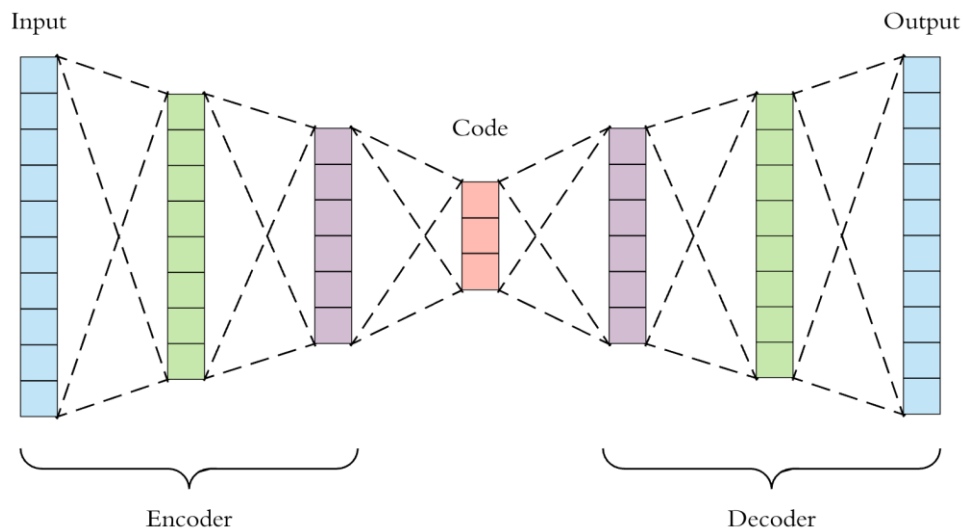


Figure 2.6: Autoencoders architecture [59]

### 2.3.4.2 Deep Belief Network (DBN)

A Deep Belief Network (DBN) is an unsupervised probabilistic deep-learning algorithm to produce outputs [60]. (DBNs) are composed of layers of Restricted Boltzmann Machines (RBMs) for the pre-train phase and then a feed-forward network for the fine-tune phase [61].

In the DBN, the input layer represents the raw sensory inputs, and each hidden layer learns abstract representations of this input. The output layer, which is treated somewhat differently than the other layers, implements the network classification. Training occurs in two steps: unsupervised pre-training and supervised fine-tuning [62].

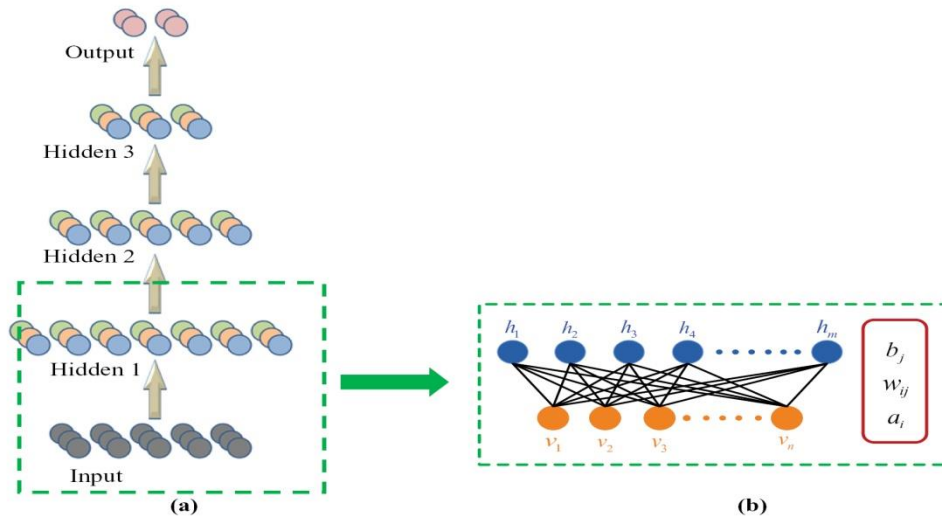


Figure 2.7: Deep Belief Network architecture [63]

In unsupervised pretraining, each RBM is trained to reconstruct its input (for example, the first RBM reconstructs the input layer to the first hidden layer). The next RBM is trained similarly, but the first hidden layer is treated as the input (or visible) layer, and the RBM is trained by using the outputs of the first hidden layer as the inputs. This process continues until each layer is pretrained. When the pretraining is complete, fine-tuning begins. In this phase, the output nodes are applied labels to give them meaning (what they represent in the context of the network). Full network training is then applied by using either gradient descent learning or back-propagation to complete the training process [62].

### 2.3.4.3 Generative Adversarial Networks (GANs)

The Generative Adversarial Network (GAN) is unsupervised learning algorithm belong to the set of generative models. It means that they are able to produce / to generate new content [64]. The idea introduced by Ian Goodfellow and colleagues at the University of Montreal in 2014.

Given a training set, the network automatically discovers and learns regularities and patterns in input data so it can self-learn to generate new data. It can essentially mimic any data set with small variations [60].

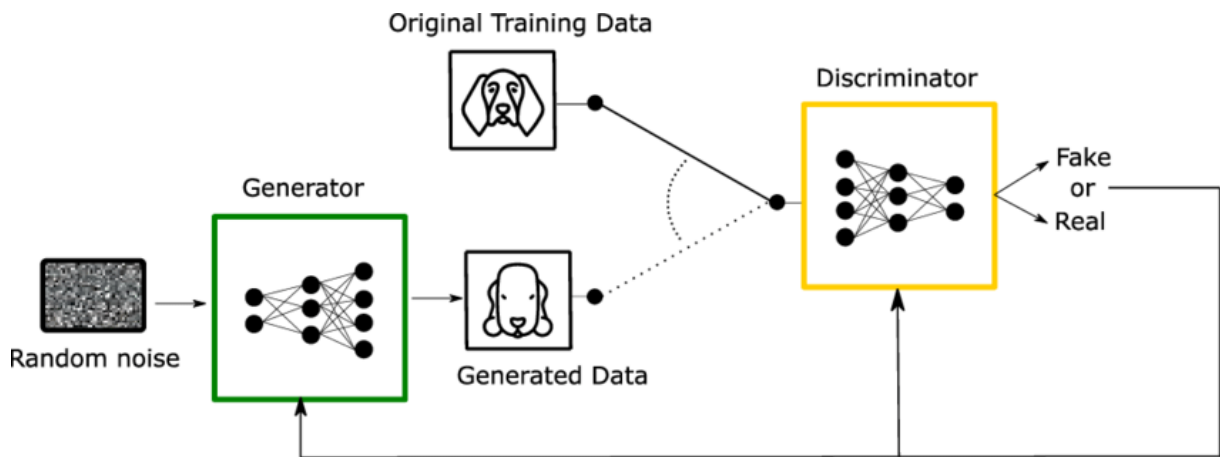


Figure 2.8: GAN architecture [65]

The GAN uses two submodels: generator and discriminator.

The generator creates new examples of data, while the discriminator distinguishes between real domain data and fake generated samples. They run repeatedly, making them more and more robust with each repetition [60].

The goal here is to make images realistic enough that the discriminator network is fooled to the point that it cannot distinguish the difference between the real and the synthetic input data [61].

#### 2.3.4.4 Recurrent Neural Network (RNNs)

A recurrent neural network (RNN) is a type of artificial neural network commonly used in speech recognition and natural language processing (NLP). RNNs are designed to recognize a data's sequential characteristics and use patterns to predict the next likely scenario [66]. RNNs are the engines behind speech recognition applications such as Apple's Siri and Google's Voice Search, as well as chatbots and translation tools [67].

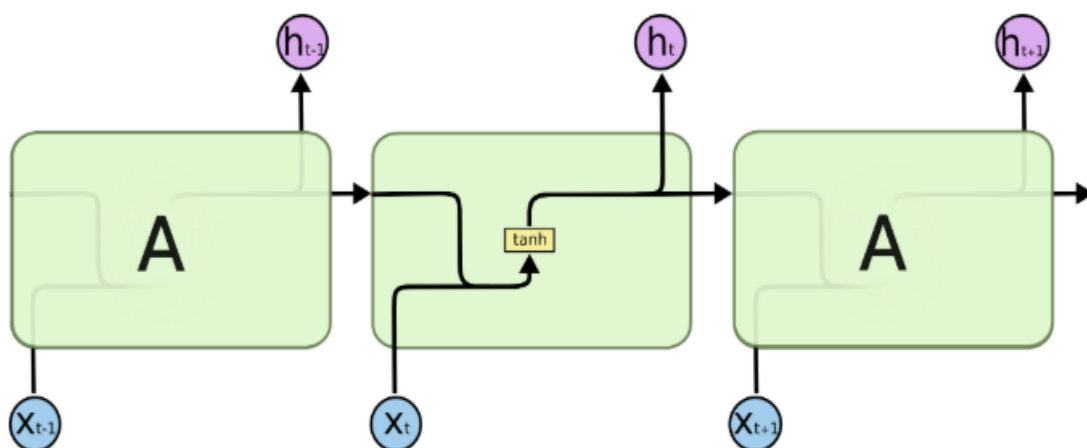


Figure 2.9: Simple RNN architecture [69]

Unlike other neural networks, an RNN has an internal memory that enables it to remember historical input; this allows it to make decisions by considering current input alongside



learning from previous input. In this way, an RNN can form a much deeper understanding of a sequence and its context than other types of deep learning algorithms, and therefore make more precise predictions [67].

There is a limitation with RNNs known as short-term memory which caused by the infamous vanishing gradient problem [68]. This lead us to the next deep learning model called LSTM.

**2.3.4.5 Long Short-Term Memory (LSTM)**

The long short-term memory (LSTM) algorithm is a type of RNN that allows deep recurrent networks to be trained without making the gradients that update weights become unstable. Patterns can be stored in memory for more extended periods, with the ability to add, remove, or modify data as needed [60].

The critical component of the LSTM is the memory cell and the gates (including the forget gate, but also the input gate). The contents of the memory cell are modulated by the input gates and forget gates. Assuming that both of these gates are closed, the contents of the memory cell will remain unmodified between one time-step and the next. The gating structure allows information to be retained across many time-steps, and consequently also allows gradients to flow across many time-steps. This allows the LSTM model to overcome the vanishing gradient problem that occurs with most Recurrent Neural Network models [61].

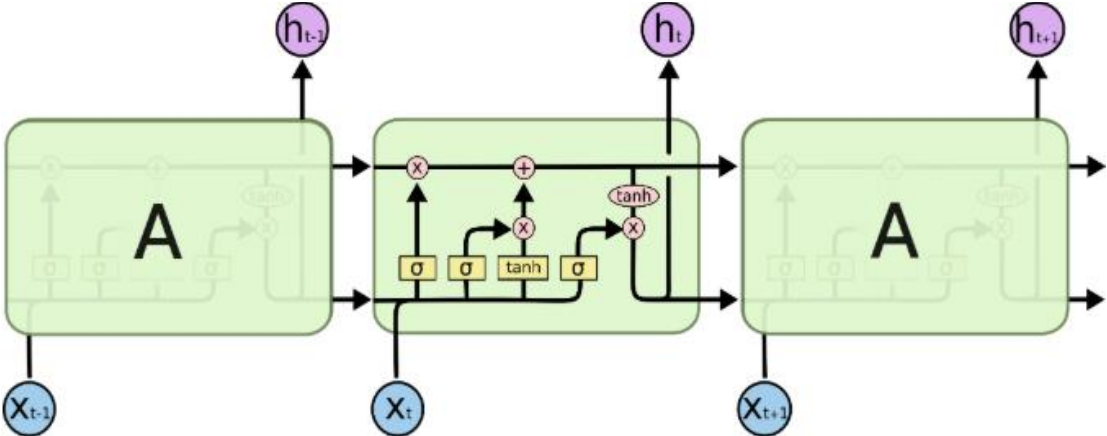


Figure 2.10: Simple LSTM architecture [69]

**2.3.4.6 Recursive Neural Networks (RNNs)**

Recursive neural networks comprise a class of architecture that can operate on structured input [70]. To produce a structured prediction over variable-size input structures, or a scalar prediction on it, by traversing a given structure in topological order [71]. Applications for recursive neural networks include image scene decomposition, NLP, Audio-to-text transcription, sentiment analysis.

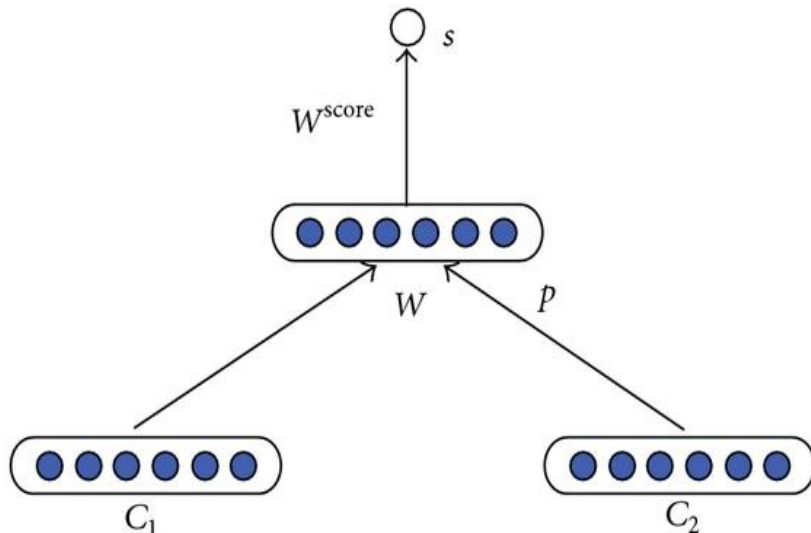


Figure 2.11: Recursive neural networks architecture [72]

### 2.3.4.7 Convolutional Neural Networks (CNNs)

#### A- Introduction

The advancements in Computer Vision with Deep Learning (Google photo search, facebook automatic tagging algorithms, amazon product recommendations) has been constructed and perfected with time, primarily over one particular algorithm — a Convolutional Neural Network [73].

#### B- Definition

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data [74]. CNNs operate in a way that is very similar to standard neural networks. A key difference, however, is that each unit in a CNN layer is a two- (or high-) dimensional filter which is convolved with the input of that layer. This is essential for cases where we want to learn patterns from high-dimensional input data. CNN filters incorporate spatial context by having a similar (but smaller) spatial shape as the input media, and use parameter sharing to significantly reduce the number of learn-able variables [61].

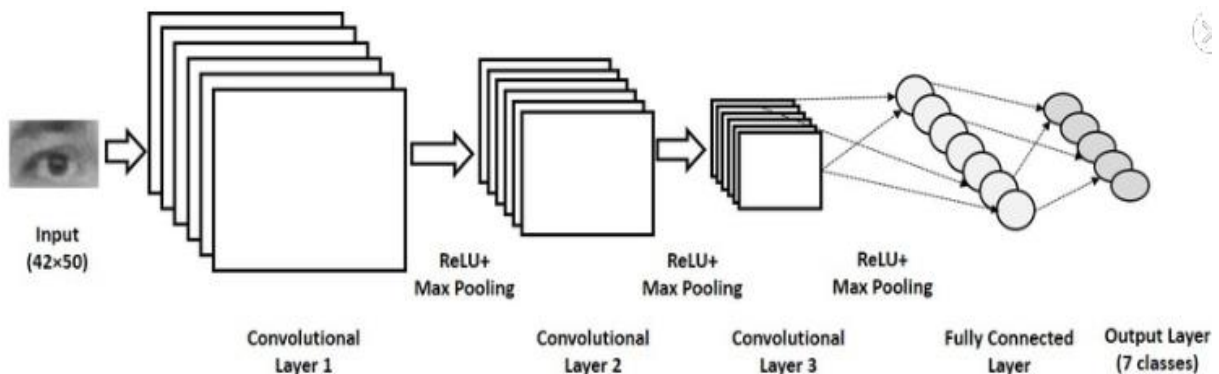


Figure 2.12: CNN architecture example [90]

## C- basic building blocks of CNN

Similar to traditional neural networks convolutional neural networks is also a sequence of layers. Each layer transforms an input image to an output image with some differentiable function that may or may not have parameters [76]. CNN architecture consists:

- Input layer
- Convolutional layer (Convolution + ReLU)
- Pooling layer
- Fully connected(FC) layer

### Input Layers

CNN take a 4D array as input. So input data has a shape of (batch\_size, height, width, depth), where the first dimension represents the **batch size** of the image and other three dimensions represent dimensions of the image which are height, width and depth. For example, RGB image would have a depth of 3 and the greyscale image would have a depth of 1 [77].

**batch size** = the number of training examples in one forward/backward pass. The higher the **batch size**, the more memory space you'll need [78].

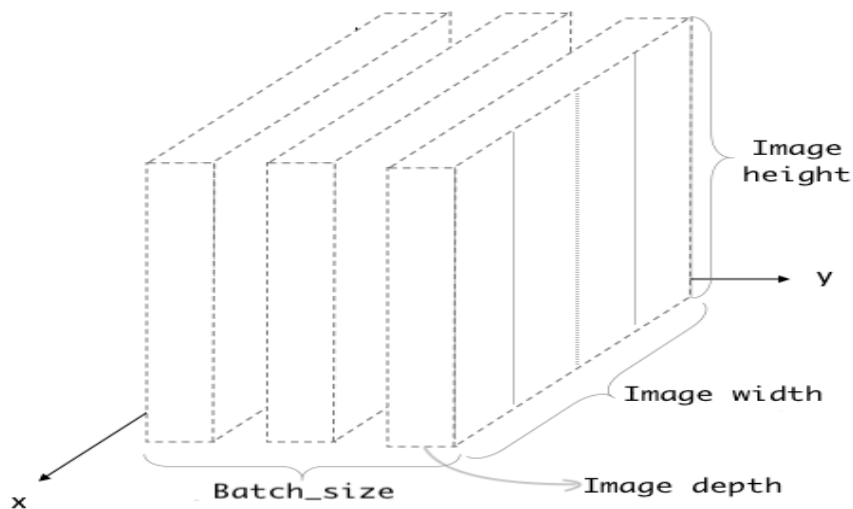


Figure 2.13: CNN Input shape [77]

### Convolutional Layers

A convolutional layer is the most important component of a CNN. It comprises a set of filters (also called convolutional kernels) which are convolved with a given input to generate an output feature map [14].

- **Filter (kernel/feature detector):**

Each filter in a convolutional layer is a grid of discrete numbers, the filter size is smaller than the input image size. The weights of each filter are learned during the training of CNN. This learning procedure involves a random initialization of the filter weights at the start of the training. Afterward, given input-output pairs, the filter weights are tuned in a number of different iterations during the learning procedure [14].

1	1	0
0	-1	1
-1	-1	1

Figure 2.14: Filter of size 3x3 [79]

• **Convolution operation:**

In mathematics, convolution is the operation of two functions to produce a third modified function. To explain this definition in the CNNs context, the first function is the input image and the second function is the CONV filter. By sliding the convolutional filter over the input image, the network is breaking the image into little chunks and it processes those chunks individually to assemble the modified image that is called feature map [15].

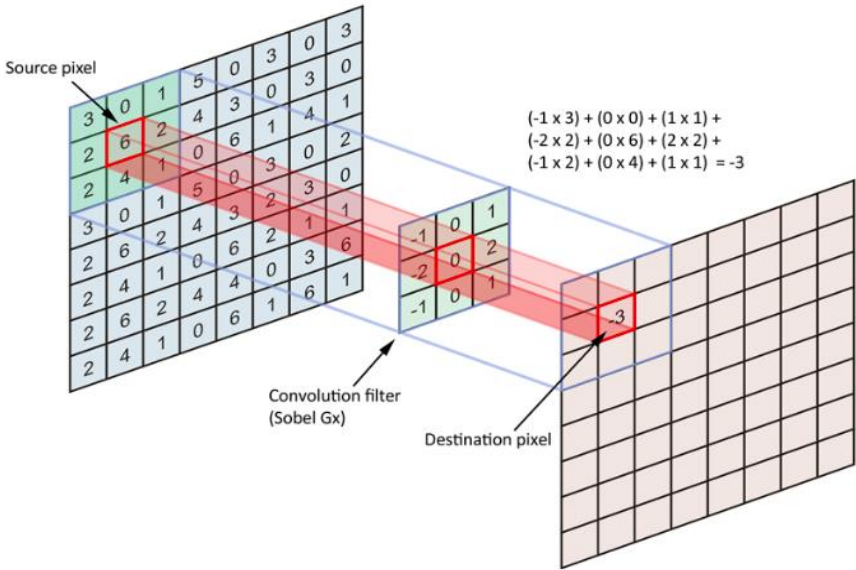


Figure 2.15: Convolution operation [80]

• **Feature map:**

The feature maps of a CNN capture the result of applying the filters to an input image, at each layer, the feature map is the output of that layer [81]. The size of the Feature Map (Convolved Feature) is controlled by three parameters which are depth, stride, and padding.

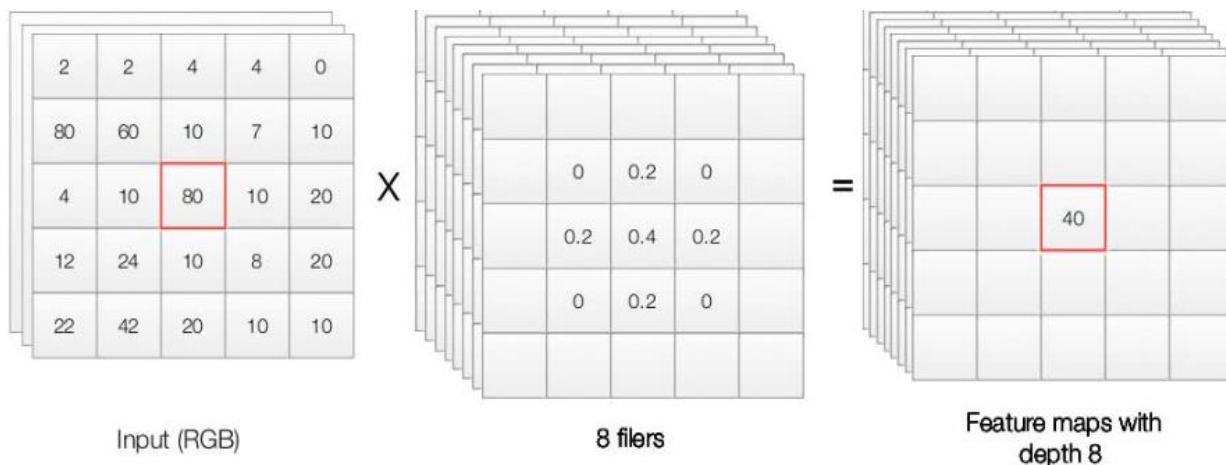


Figure 2.16: Feature maps [82]

### Introducing Non Linearity (activation layer)

After each convolution layer, it is convention to apply a nonlinear layer (or activation layer) immediately afterward. The purpose of this layer is to introduce nonlinearity to a system that basically has just been computing linear operations during the convolution layers (just element wise multiplications and summations) [84].

The most successful non-linearity for CNN's is the Rectified Non-Linear unit (ReLU), which combats the vanishing gradient problem occurring in sigmoids. ReLU is easier to compute and generates sparsity (not always beneficial) [85].

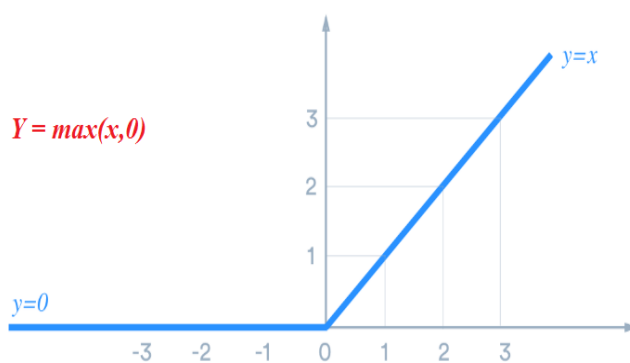


Figure 2.17: ReLU activation function [86]

The ReLU layer applies the function  $f(x) = \max(0, x)$  to all of the values in the input volume. In basic terms, this layer just changes all the negative activations to 0. This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolution layer [84].

### Filter 1 Feature Map

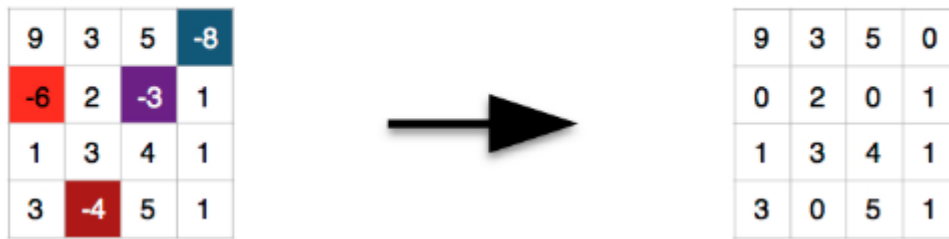


Figure 2.18: Applying ReLU after convolution operation [87]

### Hyperparameters used convolution layer:

- **Output depth** Depth corresponds to the number of filters we use for the convolution operation [76].
- **Filter size** Represents the height and width of the filters used [76].
- **Stride:** is the amount by which the filter slides over the image, large stride values decrease the size of the information that will be passed to the next layer [15].

If stride = 1, the filter will move one pixel / If stride = 2, the filter will move two pixel.

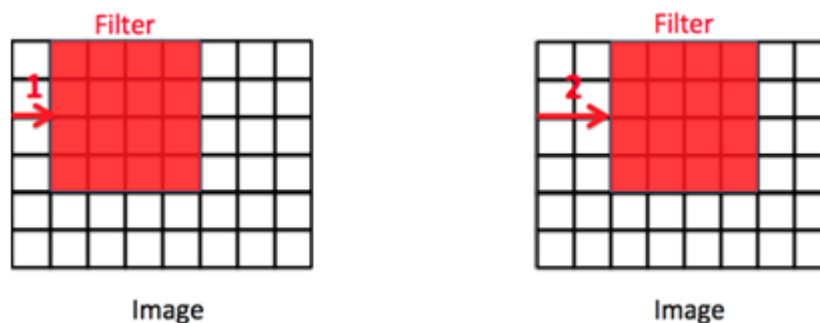


Figure 2.19: Convolution operation example with difference strides [83]

- **Padding:** often called zero-padding because we add zeros around the border of an image. Padding is most commonly used to allow us to preserve the spatial size of the input volume so the input and output width and height are the same. This way we can use CONV layers without necessarily shrinking the height and width of the volumes. This is important for building deeper networks, since otherwise the height/width would shrink as you go to deeper layers [15].

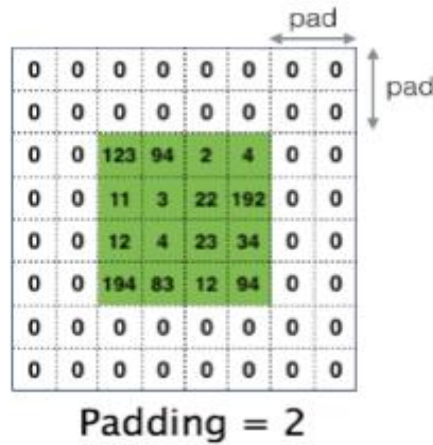


Figure 2.20: Convolution operation example with padding=2 [15]

### Pooling Layers

After adding several CONV layers this will produce a huge number of parameters (weights). This increase in the network dimensionality will increase the time and space complexity of the mathematical operations that take place in the learning process. This is when the pooling layers come in. Subsampling or pooling helps in reducing the size of the network by reducing the number of parameters passed to the next layer. The pooling operation resizes its input by applying a summary statistic function, such as a maximum or average, to reduce the overall number of parameters to be passed on to the next layer [15].

The goal of the pooling layer is to downsample the feature map into smaller number of parameters to extract representative features, reduce the computational complexity, helps reduce overfitting.

There are pooling types such as average pooling, sum pooling, max pooling, the last one is common method used in CNN [15].

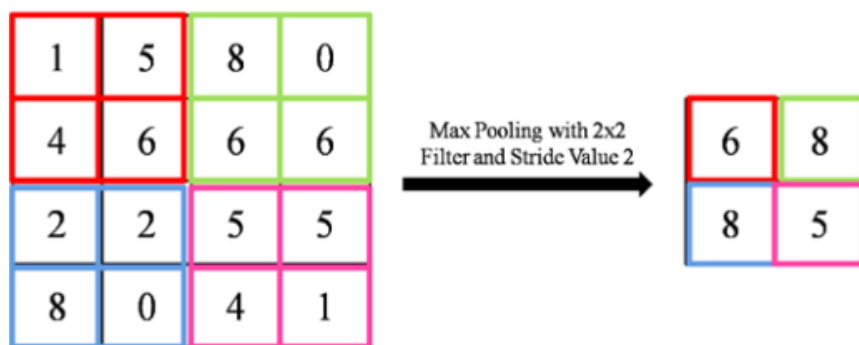


Figure 2.21: Illustration of max pooling with a pooling area of size 2x2 and stride of 2 [88]

### Fully connected Layer

Fully connected layer is a traditional multilayer perceptron structure. The objective of a fully connected layer is to take the results of the convolution/pooling process and use them to classify the image into a label (in a simple classification example).

The output of convolution/pooling is flattened into a single vector of values, each representing a probability that a certain feature belongs to a label. The label that receives the highest probability is the classification decision [89].

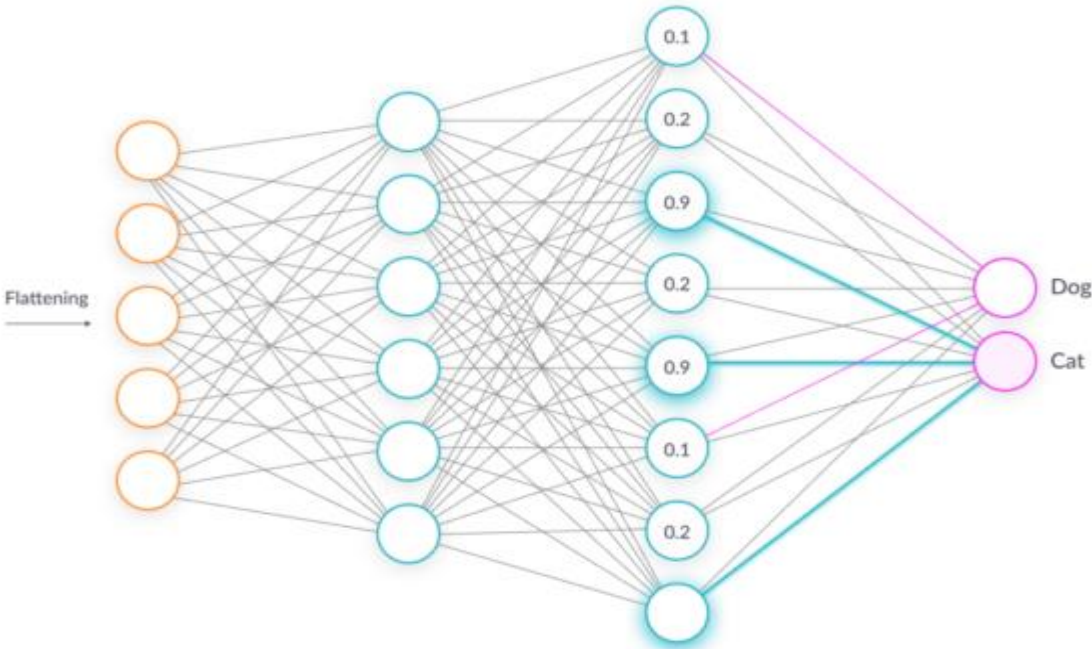


Figure 2.22: Fully connected layer [89]

The image below illustrates how the input values flow into the first layer of neurons. They are multiplied by weights and pass through an activation function (typically ReLU), just like in a classic artificial neural network. They then pass forward to the output layer, in which every neuron represents a classification label.

The fully connected part of the CNN network goes through its own backpropagation process to determine the most accurate weights. Each neuron receives weights that prioritize the most appropriate label. Finally, the neurons “vote” on each of the labels, and the winner of that vote is the classification decision [89].

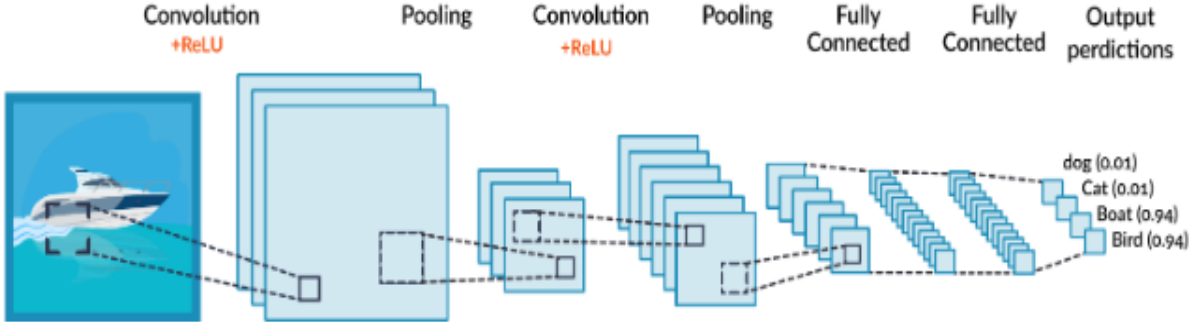


Figure 2.23: Simple CNN architecture with main layers [75]



## Common architectures in CNN

### LetNet

One of the earliest successful architectures of CNNs (developed in 1998 by Yann Lecun), originally used to read digits in images [61].

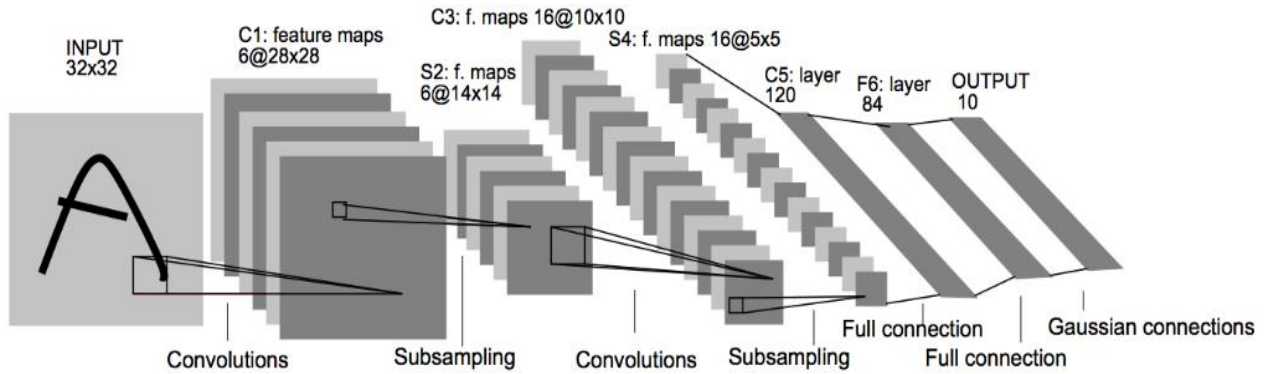


Figure 2.20: LetNet architecture [91].

### AlexNet

AlexNet was developed by Alex Krizhevsky et al. in 2012, the success of this model (Won the ILSVRC 2012) helped popularize CNNs in computer vision [61].

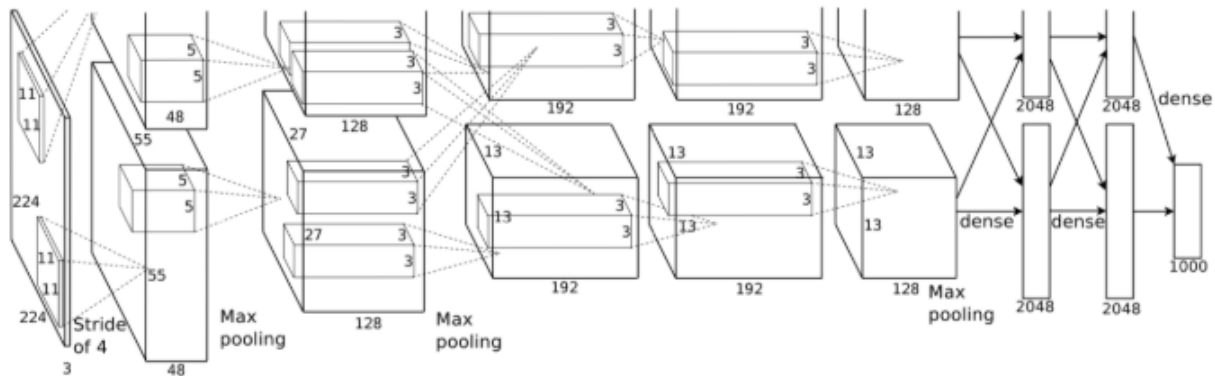


Figure 2.24: AlexNet architecture [91]

### GoogLeNet

Developed by Christian Szegedy and his team at Google, the winner of ILSVRC 2014 and the GoogLeNet architecture is also known as Inception Module [61].

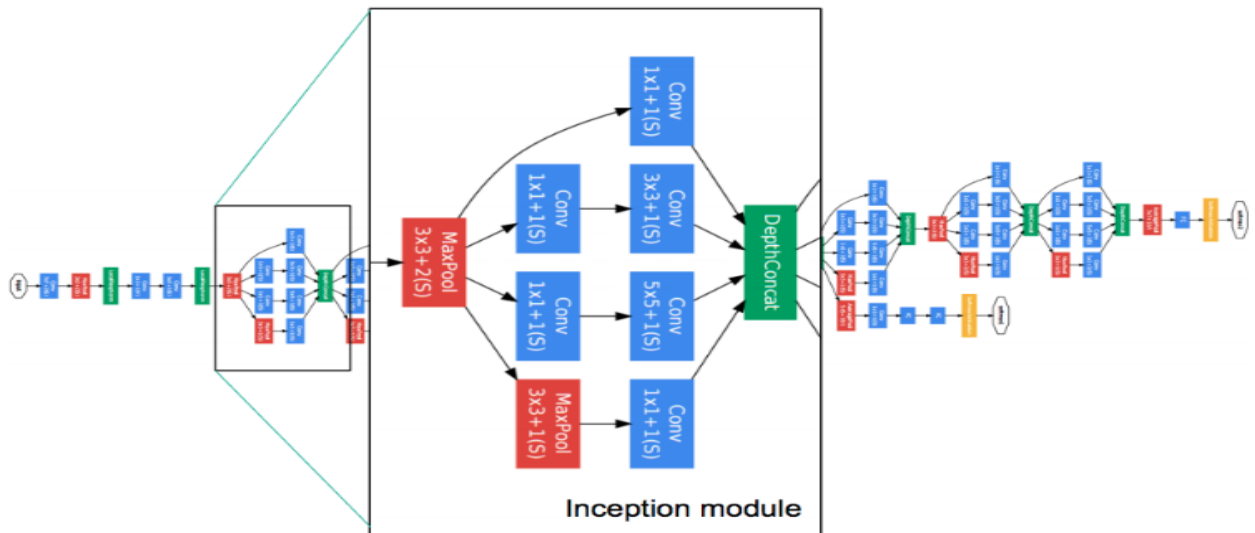


Figure 2.25: GoogLeNet architecture [92]

## VGG-16

Runner-Up in the ILSVRC 2014, Developed by Karen Simonyan and Andrew Zisserman, Showed that depth of network was a critical factor in good performance [61].

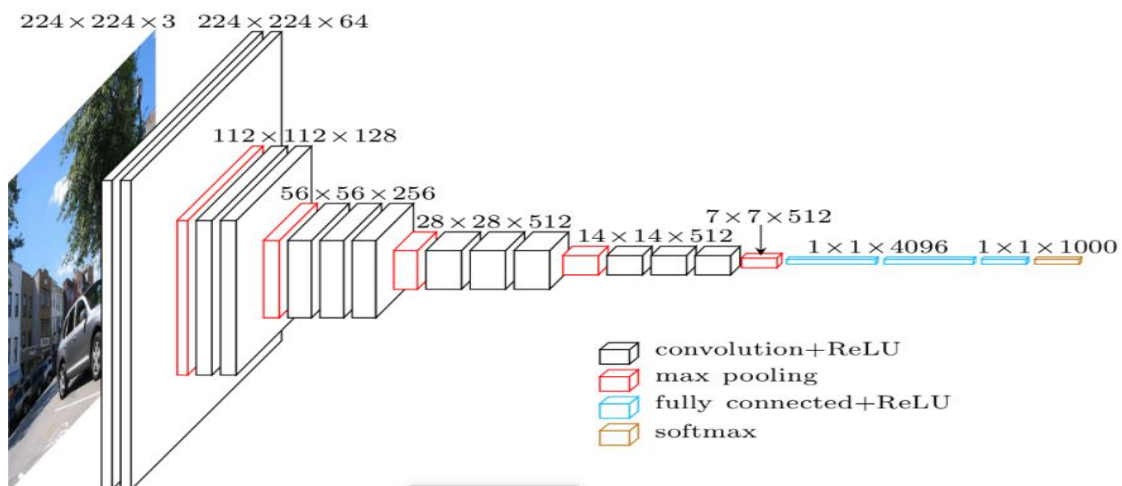


Figure 2.26: VGG-16 architecture [91]

## ResNet

It also called as Residual Neural Network (ResNet) by Kaiming. Trained on very deep networks (up to 1,200 layers), Won first in the ILSVRC 2015 classification task [61].

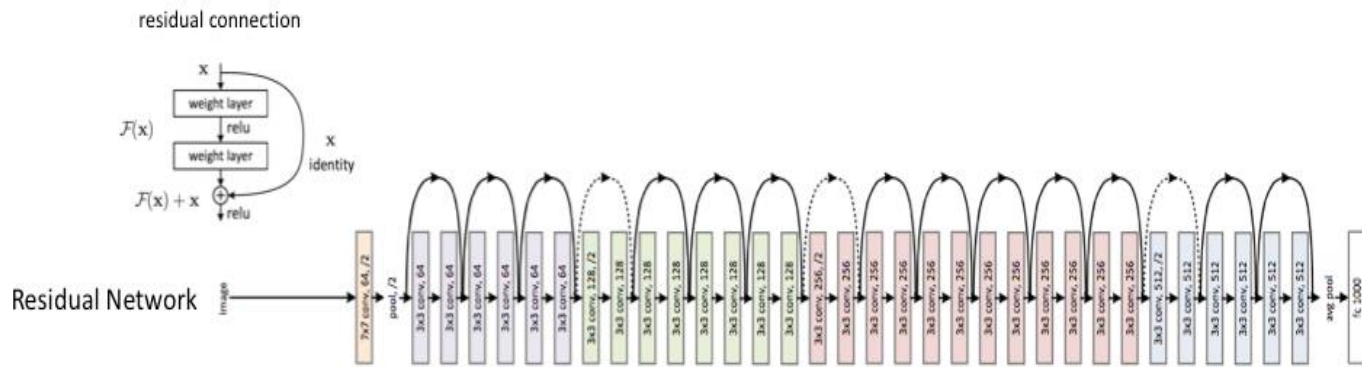


Figure 2.27: ResNet architecture [92]

## 2.3.5 Generalization in deep neural networks

### 2.3.5.1 Definition

Generalization is a term used to describe a model's ability to react to new data, it is probably the most important element of any deep learning model success. When training our deep learning model, there's training data which the model trains on, and there's testing data for checking the performance of the model. If the model performs well on the testing data which it never seen before, we can say that the model generalized well on the new data, and if the model fail to generalize on the new data then we are in front of phenomenon in neural networks called overfitting [94]. There are certain aspects of neural networks which we can control in order to prevent overfitting problem like cross-validation, dropout, weight regularization.

### 2.3.5.2 Cross-validation

Cross-validation is a powerful preventative measure against overfitting. The idea is using initial training data to generate multiple mini train-test splits. Use these splits to tune the model. In standard k-fold cross-validation, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining fold as the test set (called the "holdout fold") [95].

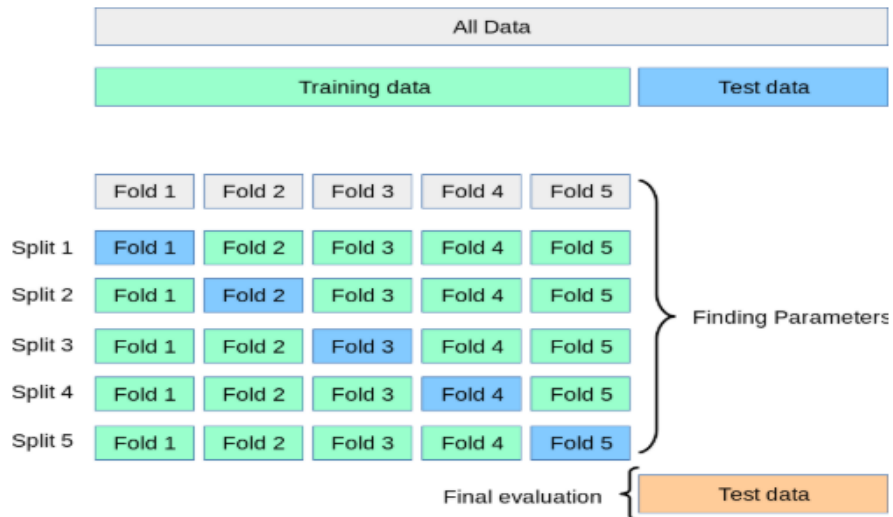


Figure 2.28: Cross-validation idea [96]

### 2.3.5.3 Dropout

In Neural Networks, adding dropout neurons is one of the most popular and effective ways to reduce overfitting in neural networks. What happens in dropout is that essentially each neuron in the network has a certain probability of completely dropping out from the network. This means that at a particular instant, there will be certain neurons which will not be connected to any other neuron in the network [94]. Here's an example:

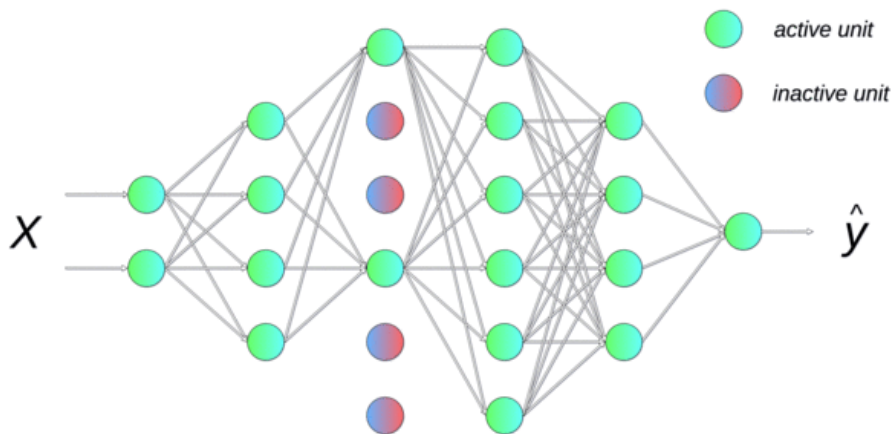


Figure 2.29: Dropout some neurons randomly with certain probability [97]

### 2.3.5.4 L1 and L2 Regularizations

One of the first methods we should try when we need to reduce overfitting is regularisation. It involves adding an extra element to the loss function, which punishes our model for being too complex or, in simple words, for using too high values in the weight matrix. This way we try to limit its flexibility, but also encourage it to build solutions based on multiple features [97].

There are two methods of doing this, they are called the L1 and L2 regularization. In L1 we take a small part of the sum of all the absolute values of the weights in the network. In L2, we take a small part of the sum of all the squared values of the weights in the network. We just add this expression to the overall loss function of the neural network [94].

$$L(x, y) \equiv \sum_{i=1}^n (y_i - h_{\theta}(x_i))^2 + \lambda \sum_{i=1}^n |\theta_i|$$

$$L(x, y) \equiv \sum_{i=1}^n (y_i - h_{\theta}(x_i))^2 + \lambda \sum_{i=1}^n \theta_i^2$$

Figure 2.30: L1 and L2 equations [98]

### 2.3.6 Deep learning applications

#### 2.3.6.1 Entertainment

Wimbledon 2018 used IBM Watson to analyse player emotions and expressions through hundreds of hours of footage to auto-generate highlights for telecast. This saved them a ton of effort and cost. Thanks to Deep Learning, they were able to factor in audience response and match or player popularity to come up with a more accurate model (otherwise it would just have highlights of the most expressive or aggressive players). Netflix and Amazon are enhancing their deep learning capabilities to provide a personalized experience to its viewers by creating their personas factoring in show preferences, time of access, history, etc. to recommend shows that are of liking to a particular viewer. Deep Learning AI is revolutionizing the filmmaking process as cameras learn to study human body language to imbibe in virtual characters.

#### 2.3.6.2 Fraud detection

Another domain benefitting from Deep Learning is the banking and financial sector that is plagued with the task of fraud detection with money transactions going digital. Autoencoders in Keras and Tensorflow are being developed to detect credit card frauds saving billions of dollars of cost in recovery and insurance for financial institutions. Fraud prevention and detection are done based on identifying patterns in customer transactions and credit scores, identifying anomalous behavior and outliers. Classification and regression machine learning techniques and neural networks are used for fraud detection. While machine learning is mostly used for highlighting cases of fraud requiring human deliberation, deep learning is trying to minimize these efforts by scaling efforts.

#### 2.3.6.3 Healthcare

Deep learning in healthcare has already left its mark. Google has spent a significant amount of time examining how deep learning models can be used to make predictions around hospitalized patients, supporting clinicians in managing patient data and outcomes [99]. Deep learning has been playing an important role in medical diagnosis and research. It helps with diagnosis of life-threatening diseases, pathology results and treatment cause standardization and understanding genetics to predict future risks of diseases[100].

Readmissions are a huge problem for the healthcare sector as it costs tens of millions of dollars in cost. The use of deep learning and neural networks, healthcare giants are mitigating health risks associated with readmissions while bringing down the costs. AI is also being exceedingly being used in clinical researches by regulatory agencies to find cures to untreatable diseases but physicians scepticism and lack of a humongous dataset are still posing challenges to the use of deep learning in medicine [98].

#### **2.3.6.4 Virtual assistants**

The most popular application of deep learning is virtual assistants ranging from Alexa to Siri to Google Assistant. Each interaction with these assistants provides them with an opportunity to learn more about your voice and accent, thereby providing you a secondary human interaction experience. Virtual assistants use deep learning to know more about their subjects ranging from your dine-out preferences to your most visited spots or your favorite songs. They learn to understand your commands by evaluating natural human language to execute them. Another capability virtual assistants are endowed with is to translate your speech to text, make notes for you, and book appointments. Virtual assistants are literally at your beck-and-call as they can do everything from running errands to auto-responding to your specific calls to coordinating tasks between you and your team members. With deep learning applications such as text generation and document summarizations, virtual assistants can assist you in creating or sending appropriate email copy as well [98].

### **2.3.7 Related works**

#### **2.3.7.1 Music genre classification**

Categorizing music files according to their genre is a challenging task in the area of music information retrieval (MIR). Hareesh Bahuleyan used a deep learning approach wherein a CNN model is trained end-to-end, to predict the genre label of an audio signal, solely using its spectrogram. Convolutional neural networks (CNNs) have been widely used for the task of image classification. In this study, the sound wave can be represented as a spectrogram, which in turn can be treated as an image. The task of the CNN is to use the spectrogram to predict the genre label (one of seven classes) [101].

#### **2.3.7.2 Lung cancer detection and classification**

Lung cancer is one of the most killer diseases in the developing countries and the detection of the cancer at the early stage is a challenge. Analysis and cure of lung malignancy have been one of the greatest difficulties faced by humans over the most recent couple of decades. Early identification of tumor would facilitate in sparing a huge number of lives over the globe

consistently. (S. Sasikala et.al) present an approach which utilizes a Convolutional Neural Network (CNN) to classify the tumors found in lung as malignant or benign. The accuracy obtained by means of CNN is 96%, which is more efficient when compared to accuracy obtained by the traditional neural network systems [102].

### **2.3.7.3 Photo-realistic facial Details synthesis from single image**

Anpei Chen et.al proposed a single-image 3D face synthesis technique that can handle challenging facial expressions while recovering fine geometric details. their technique employs expression analysis for proxy face geometry generation and combines supervised and unsupervised learning for facial detail synthesis. On proxy generation, they conduct emotion prediction to determine a new expression-informed proxy. On detail synthesis, they present a Deep Facial Detail Net (DFDN) based on Conditional Generative Adversarial Net (CGAN) that employs both geometry and appearance loss functions. For geometry, we capture 366 high-quality 3D scans from 122 different subjects under 3 facial expressions. For appearance, they used additional 163K in-the-wild face images and apply image-based rendering to accommodate lighting variations. Comprehensive experiments demonstrate that their framework can produce high-quality 3D faces with realistic details under challenging facial expressions [103].

### **2.3.7.4 Deepfakes creation and detection**

Deepfake (stemming from “deep learning” and “fake”) is a technique that can superimpose face images of a target person to a video of a source person to create a video of the target person doing or saying things the source person does. Deep learning models such as autoencoders and generative adversarial networks (GANs) have been applied widely in the computer vision domain to solve various problems these models have also been used by deepfake algorithms to examine facial expressions and movements of a person and synthesize facial images of another person making analogous expressions and movements. Deepfake algorithms normally require a large amount of image and video data to train models to create photo-realistic images and videos. As public figures such as celebrities and politicians may have a large number of videos and images available online, they are initial targets of deepfakes [104].

### **2.3.7.5 Human activity recognition**

A human activity recognition design has been proposed by Schalk Wilhelm Pienaa et.al using an LSTM-RNN deep neural architecture. The network is able to learn six classes (downstairs, jogging, sitting, standing, upstairs, walking) successfully and efficiently with only a few hundred epochs, reaching high accuracies above 94% and a loss of less than 30% has been reached in the first 500 epochs of training. The work can be further extended with different data sources as well as different classes. The model has been successfully exported and loaded to an Android app together with the Tensorflow library, giving the ability to perform real-time predictions. L2 regularization has already been implemented to prevent overfitting, but another solution would be to add dropout, which is one of the most effective and commonly used regularisation techniques for neural networks to prevent overfitting models [105].

### 2.3.7.6 Weather image recognition

- [Jehong An et. al 2018] describe a new weather image classification technique using Alexnet and Resnet (CNN architectures) combined with a multiclass Support Vector Machine (SVM). The experimental datasets used are the weather database [106], desnownet [107], and d-hazy dataset [108]. Experimental results show that this method works well with 4 classes used with classification accuracy is 97 % sunny, 100 % cloudy, 96 % hazy, and 95 % snowy [109].
- [Jose Carlos Villarreal Guerra et. al 2018] have created a new open source dataset consisting of images depicting three classes of weather i.e. rain, snow and fog called RFS Dataset. A novel algorithm has also been proposed which has used super pixel [110] delimiting masks as a form of data augmentation [111], they used 10 CNN architectures (such as CaffeNet, VGGNet16, VGGNet19, ResNet 50.....etc). they got reasonable results, about 80% accuracy in resnet50 architecture with 100 SP (superpixel) [112].
- [Mohamed R. Ibrahim et. al 2019] proposed a novel framework to automatically extract weather and visual conditions from street-level images of urban scenes. They downloaded 23,865 images from the web, for training and testing. A pipeline of four CNN models, so-called WeatherNet, is trained, relying on residual learning using ResNet50 architecture, to extract various weather and visual conditions they combine four models such as dawn/dusk or day or night(model 1), glare or no glare (model 2) , and clear or rainy or snowy (model 3), and or fog/no fog (model 4). Each model trained independently. WeatherNet shows promising results with 91.6%, 94.8%, 93.2%, 95.2% accuracies respectively for the 4 models [113].

## 2.4 conclusion

Deep learning methods showed high-level results in a lot of real world applications which impact directly or indirectly on computer vision field or other fields. Although of this improvement there are some limitations of course due to huge data and time complexity and other problems used for training deep models. In this chapter, we have discussed the buzz word 'deep learning' and how it used for image classification with detailed explanation about one of the successful models (CNN), also we touched some deep learning research areas.

Build on this discussion, we concluded that the weather image classification can benefit from deep learning techniques more. This motivated us to use deep learning model, which we will be discuss in the next chapter.



# Chapter 3

## System Architecture, Implementation and Results

### 3.1 Introduction

In our work, we focused on making a deep learning model that able to recognize different weather conditions from images. To achieve this, we proposed Convolutional neural network architecture, one of the successful deep learning algorithms in image recognition tasks.

In this chapter we are going to discover the process of how we prepare and pre-process our dataset for train and evaluate a CNN model for the weather conditions classification with the appropriate environments and tools and the achieved results.

### 3.2 System Architecture

#### 3.2.1 Introduction

In this work, we collected weather images from different weather datasets also we have created small dataset due to some issues in first datasets. We are using image pre-processing techniques and data augmentation. And as we mentioned before we are going to use Convolution neural network (CNN), which is a supervised algorithm, taken in consideration the big achievements of that algorithm in image classification.

CNN algorithm helps us to make Weather Conditions classification that we want to reach. This task is done by designing and training different CNN architectures with dataset until we get a model with high accuracy as possible as we can.

#### 3.2.2 System Architecture

Our system for classifying weather images goes through a number of stages, and this is the architecture that the figure 3.1 summarizes in general.

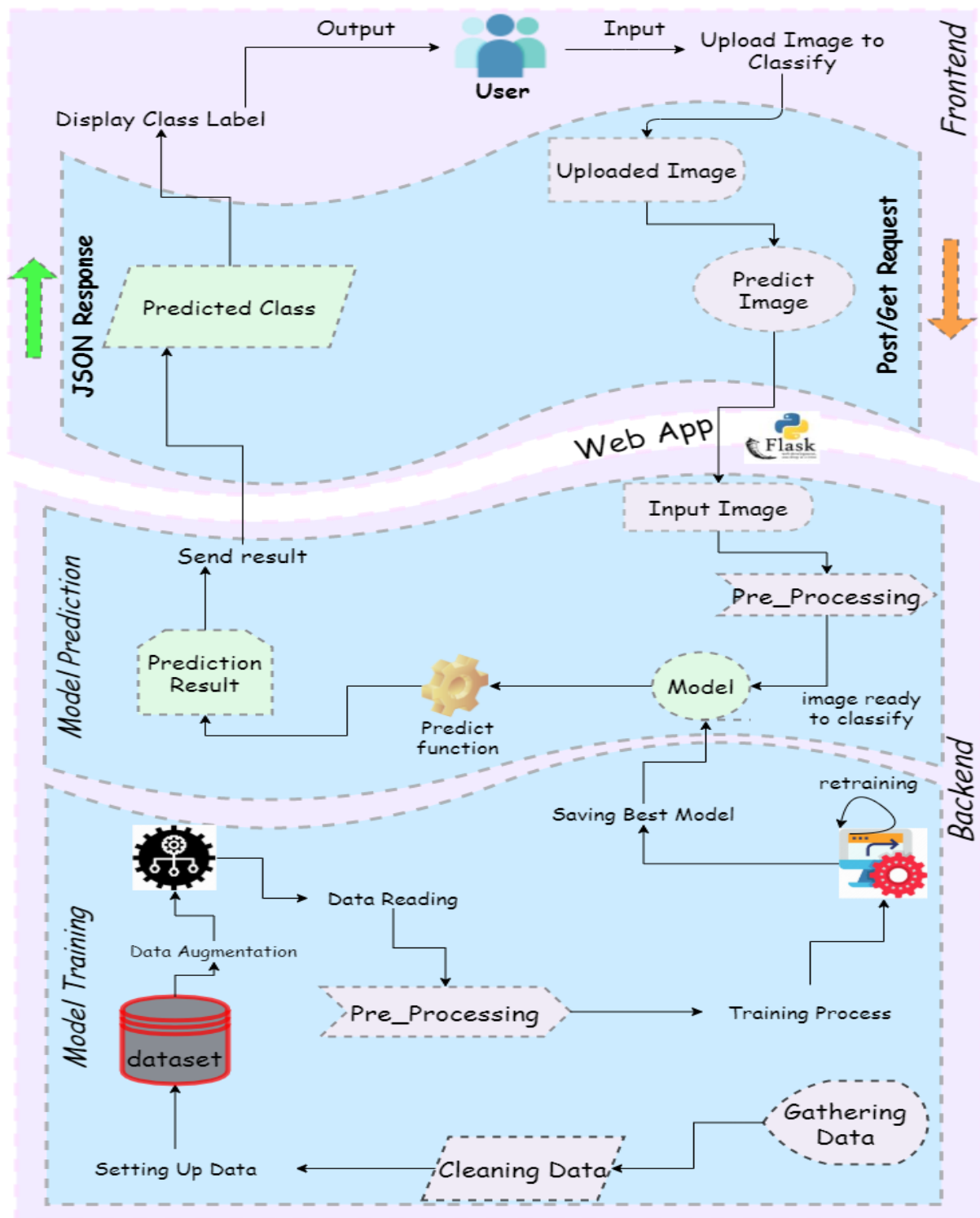


Figure 3.1: System General Architecture

We can see that our system has two main stages which are Front-end and Back-end:

*Front-end:* is synonymous with the user interface, is what happens in the browser everything the user sees and interact with. For our work the user upload an image and request for prediction then getting response of the class label from the back-end.

*Back-end:* is the base of our work, in this stage we are going to gathering images dataset and cleaning up the data also performing image pre-processing technique, data augmentation and designing the CNN architecture for the training with different models until we have an accurate model then saving it to make predictions later if we have a request from the user.

### 3.2.2.1 Back-end

The Back-end is the heart of the system and it going through a pipline as the following:

- A. Dataset gathering, preparation
- B. Dataset Pre-processing
- C. Model Training
- D. Model Evaluation
- E. Hyperparameter tuning
- F. Model Prediction

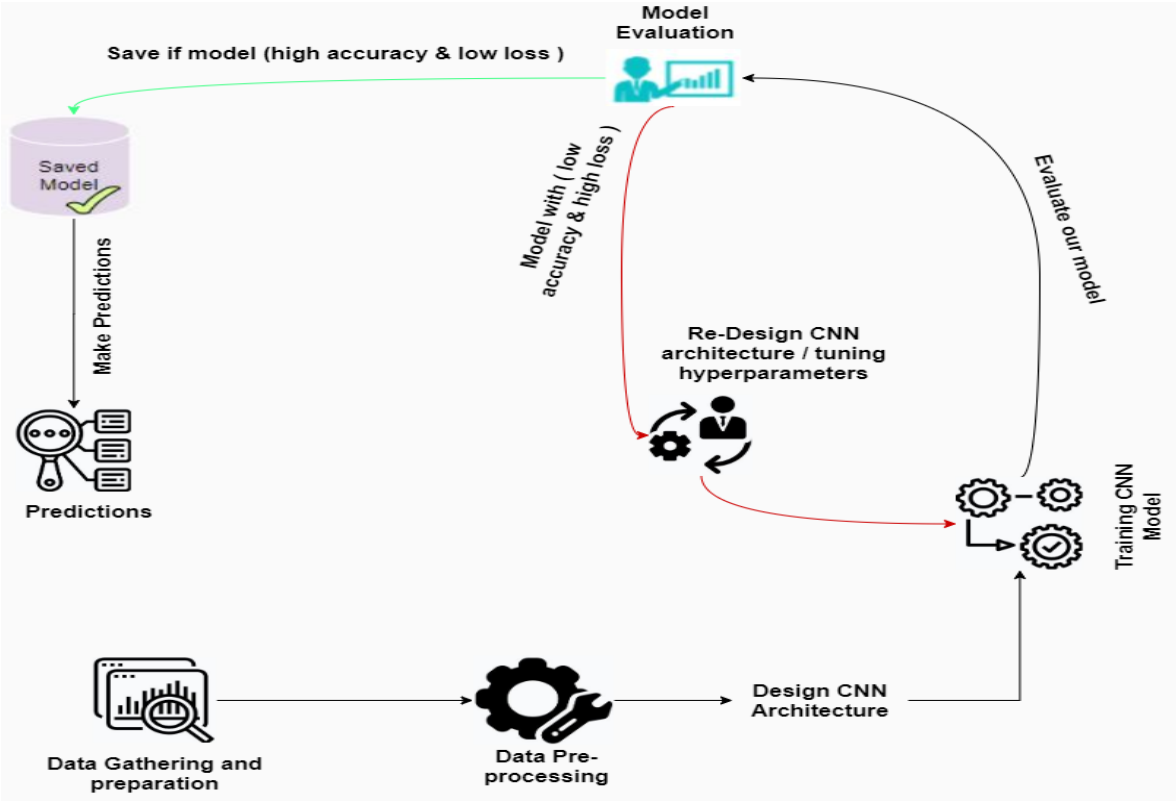


Figure 3.2: Back-end Architecture

## A- Dataset gathering, preparation

In our work we going to use five different weather conditions from images, our classes are the following:

- Cloudy weather condition
- Foggy weather condition
- Rainy weather condition
- Snowy weather condition
- Sunny weather condition

For achieving our purpose we collected weather images from different datasets used in related works and combining between them. Our first dataset source is:

- **Image2Weather Dataset**

Image2Weather dataset created by Wei-Ta Chu et. al in 2017 for their work "Camera as Weather Sensor: Estimating Weather Information from Single Images".

This dataset consists of 183,798 images in total. Overall, geographically the collected images span from 9.25 degrees west longitude to 30.4 degrees east longitude, and from 35 degrees north latitude to 62.21 degrees north latitude (covering most part of the Europe). The range of temperature is from -25 degree Celsius to 45 degree Celsius, and the range of humidity is from 4% to 100% [115].

Weather condition	Samples
Cloudy weather	45,662
Foggy weather	357
Rainy weather	1,369
Snowy weather	1,252
Sunny weather	70,501
Other images	64,657
<b>Total</b>	<b>183,798</b>

Table 3.1: Image2Weather Dataset [115]

We can notice from the table there are five conditions ( cloudy, foggy, rainy, snowy, sunny) also the table have a row called other images, according to the dataset creators they manually correct weather types and label images with unclear weather conditions into "others".

For our dataset we ignore the “others” images because it take a lot of time to label about 64,657 images manually. We also notice the big gap between classes samples which mean that this data is **Imbalanced** dataset.

- **Imbalanced** data typically refers to a problem with classification problems. Where the class distribution is not uniform among the classes. Typically, they are composed by two classes: The majority (negative) class and the minority (positive) class. In such cases, you get a pretty high accuracy just by predicting the majority class, but you fail to capture the minority class, which is most often the point of creating the model in the first place [116].

### - Re-sampling Dataset

To make our dataset balanced there are ways to do so:

#### Collecting more data

**Under-sampling:** Remove samples from the majority classes.

**Over-sampling:** Add more samples to the minority classes.

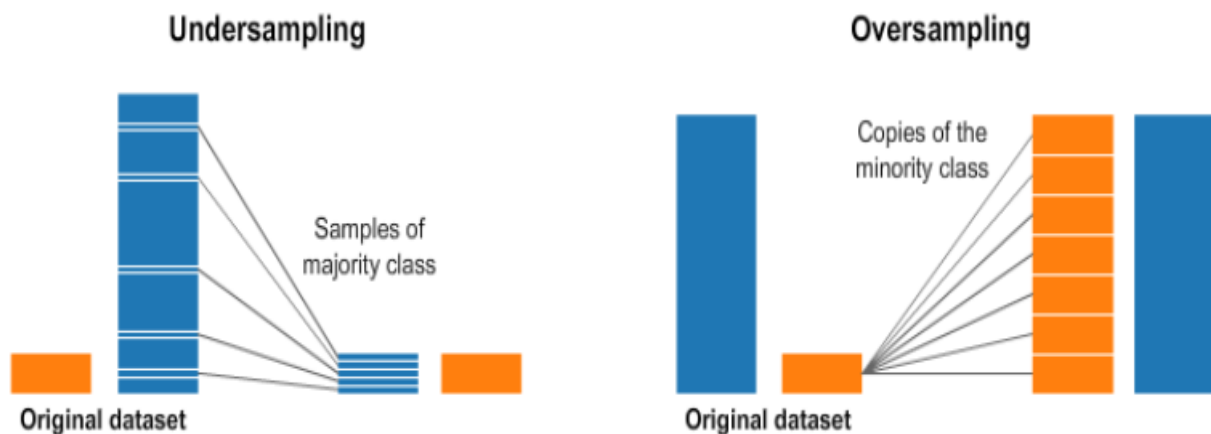


Figure 3.3: Re-sampling dataset [116]

To achieve our purpose for balancing the dataset we use all possible techniques mentioned above. In our way for collecting more data for the minority classes (foggy, rainy, and snowy).

We found three datasets for foggy class and one dataset for snowy for Over-sampling our data:

- **FRIDA (Foggy Road Image DAtabase) image database:**

FRIDA and FRIDA2 are databases of numerical images easily usable to evaluate in a systematic way the performance of visibility and contrast restoration algorithms. FRIDA comprises 90 synthetic images of 18 urban road scenes. FRIDA2 comprises 330 synthetic images of 66 diverse road scenes [117]. We use all this data for Over-sampling our data.



Figure 3.4: FRIDA dataset [117]

- **LIVE Image Defogging Database**

L. K. Choi et. al created this dataset for proposed model, called Fog Aware Density Evaluator (FADE), predicts the visibility of a foggy scene from a single image without reference to a corresponding fog-free image, without dependency on salient objects in a scene, without side geographical camera information, without estimating a depth dependent transmission map, and without training on human- rated judgments. FADE only makes use of measurable deviations from statistical regularities observed in natural foggy and fog-free images.

LIVE Image Defogging Database consists of 500 natural foggy images, 500 natural fog-free images, and 100 test foggy images [118]. We take for our foggy class about 500 images.



Figure 3.5: LIVE Image Defogging Database [118]

- **REALISTIC Single Image DEhazing (RESIDE)**

RESIDE highlights diverse data sources and image contents, and is divided into five subsets, each serving different training or evaluation purposes. We further provide a rich variety of criteria for dehazing algorithm evaluation, ranging from full-reference metrics, to no-reference metrics, to subjective evaluation and the novel task-driven evaluation. Experiments on RESIDE shed light on the comparisons and limitations of state-of-the-art dehazing algorithms, and suggest promising future directions [119].

Type	Samples
<b>Synthetic Indoor Hazy Images</b>	<b>110,500</b>
<b>Synthetic Outdoor Hazy Images</b>	<b>313,950</b>
<b>Realistic Hazy Images (Unannotated)</b>	<b>4,807</b>
<b>Realistic Annotated Hazy Images</b>	<b>4,322</b>

Table 3.2: RESIDE Dataset [119]

From this data we took Realistic Hazy Images (Unannotated) with 4,807 images.

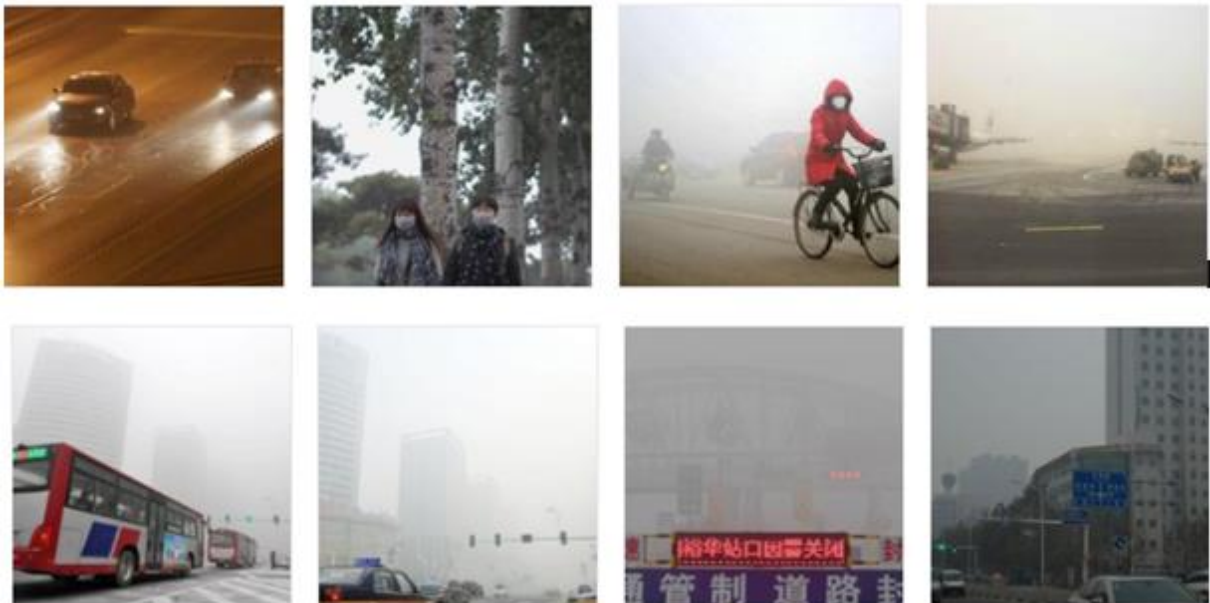


Figure 3.6: Samples of RESIDE dataset [119]

- **Snow100K dataset**

This dataset created by Liu et. al for *DesnowNet model* that automatically localizes the translucent and opaque snow particles from realistic winter photographs, and removes them to achieve better visual clarity on corresponding resultants [107].

Type	Samples
Synthesized snowy images	100k
Snow-free ground truth images	100k
Snow masks	100k
Realistic snowy images	1,329

Table 3.3: Snow100k Dataset [107]

We added the Realistic snows images for our dataset (1,329 samples)



Figure 3.7: Samples of snow100k dataset [107]



With all of this data our dataset still imbalanced, so we under-sampling the two classes with majority (Cloudy, Sunny), the last thing we did for balanced our data is creating a small dataset for the three minority classes (foggy, rainy, snowy).

class	Videos	Images
<b>Foggy Weather</b>	<b>10</b>	<b>1,916</b>
<b>Rainy Weather</b>	<b>16</b>	<b>2,631</b>
<b>Snowy Weather</b>	<b>12</b>	<b>1,419</b>

Table 3.4: New Dataset

We created this small dataset by downloading about 10 YouTube videos with a foggy condition, 16 videos with a rainy condition, and 12 videos with condition. The videos filmed with high quality cameras in different places from the world (New York, Boston, London, India, and China). The next step we converted those videos into frames, we toke a frame each 10 seconds for each video then deleted the frames with bad quality or noisy, and finally we added this new images to the old dataset.

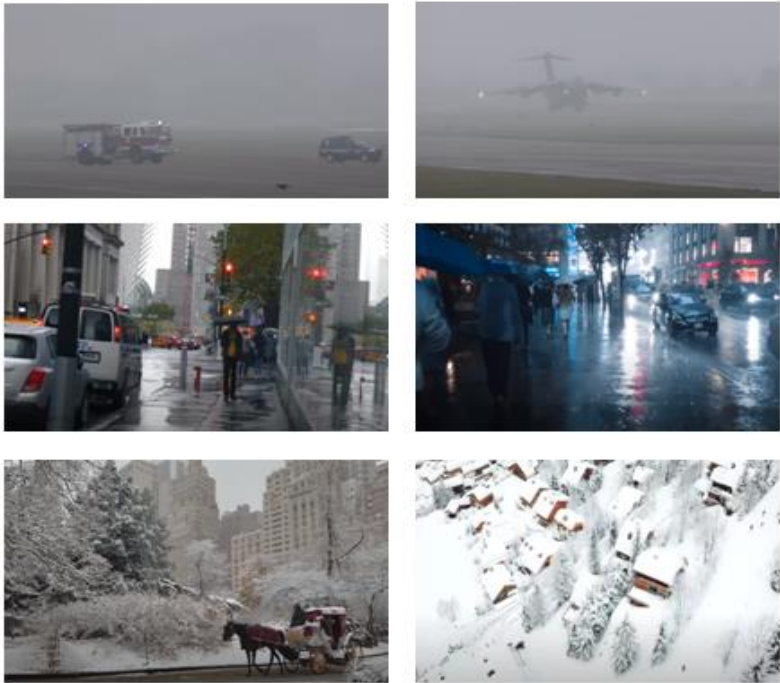


Figure 3.8: Samples of the new dataset

Finally, after we collected and prepared our dataset by removing bad quality, noisy images, and balanced the classes (under-sampling, over-sampling) we work with final dataset as showing in the table 3.5:

Weather condition	Samples
Cloudy weather	4,000
Foggy weather	4,000
Rainy weather	4,000
Snowy weather	4,000
Sunny weather	4,000

Table 3.5: Final Dataset

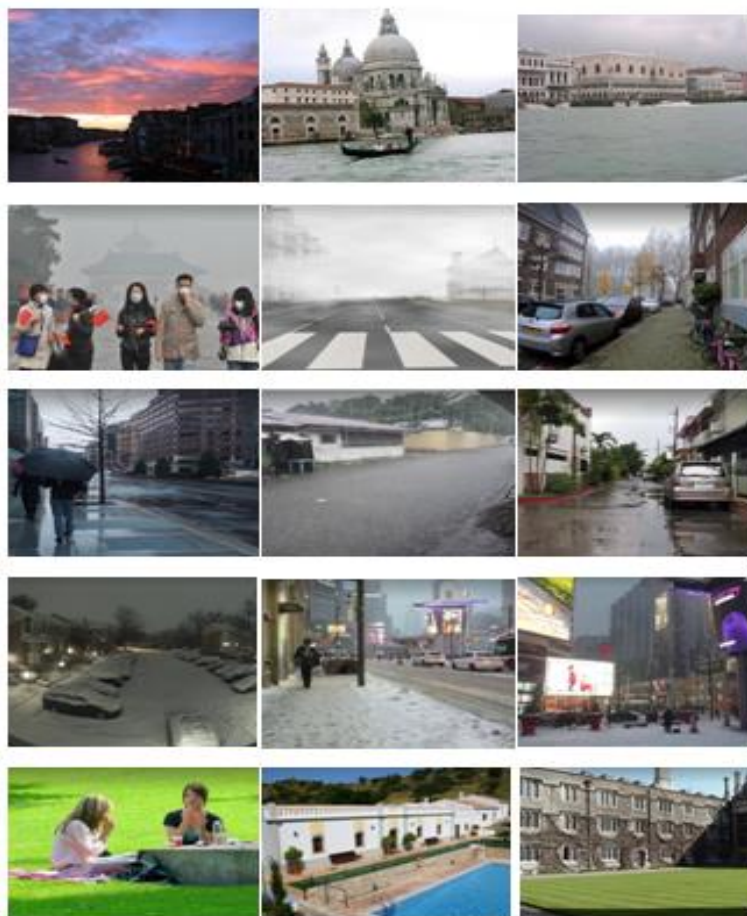


Figure 3.9: Samples of the final dataset

## B- Dataset Pre-processing

Data preprocessing is a technique that helps enhance the quality of data to promote the extraction of meaningful insights from the data. In simple way, data preprocessing in Machine Learning is a data mining technique that transforms raw data into an understandable and readable format.

Why we need data Pre-processing?

When it comes to creating a Machine Learning model, data preprocessing is the first step marking the initiation of the process. Typically, real-world data is incomplete, inconsistent, inaccurate (contains errors or outliers), and often lacks specific attribute values/trends. This is where data preprocessing enters the scenario – it helps to clean, format, and organize the raw data, thereby making it ready-to-go for Machine Learning models [120].

There is a pipeline to follow in data pre-processing as shown in figure 3.9:

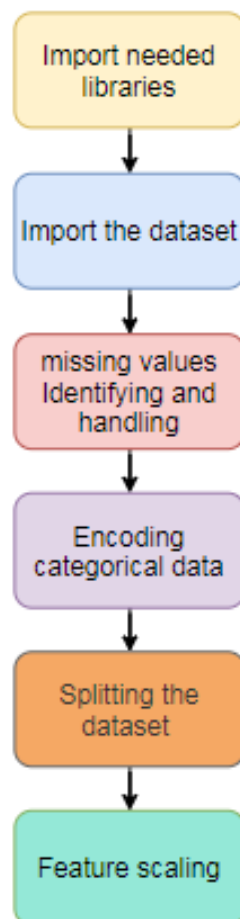


Figure 3.10: Pre-processing pipeline

### Stage 1: Import needed libraries

It's the first step to take in data pre-processing. Libraries are particularly useful for storing frequently used routines because we don't need to explicitly link them to every program that uses them which help us to simply process our dataset. This is what python providing us with of useful image processing libraries that help also for simply loading our dataset, we going to discover them in the implementation section.

### Stage 2: Import the dataset

We can import the dataset for many ways it depend on the format files of the dataset such as (data.npy, data.csv ...) each dataset format can be loaded with specific library. And it is useful if we put our dataset in same working directory to make the access more rapidly.

### Stage 3: missing values Identifying and handling

In data preprocessing, it is pivotal to identify and correctly handle the missing values, if we miss this step we can ended up with inaccurate model, there are two ways to handle missing data the first one is Deleting a particular row its commonly used to handle the null values where more than 75% of the values are missing and it's not always recommended. The second one is calculating the mean it is useful for features having numeric data like age, salary, year, etc. Here, you can calculate the mean, median, or mode of a particular feature or column or row that contains a missing value and replace the result for the missing value.

### Stage 4: Encoding categorical data

Categorical data refers to the information that has specific categories within the dataset. Since, machine learning models are based on Mathematical equations and you can intuitively understand that it would cause some problem if we can keep the Categorical data in the equations because we would only want numbers in the equations [121]. There a methods to do this work like dummy variable, label encoding, frequency encoding, and one hot encoding [122]. A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1. For our dataset (images) are already numerical but we need this step for our labels (cloudy, foggy, rainy, snowy, and sunny). As shown in the table 3.6:

Label	Integer	One Hot Encoding
Cloudy	0	[1, 0, 0, 0, 0]
Foggy	1	[0, 1, 0, 0, 0]
Rainy	2	[0, 0, 1, 0, 0]
Snowy	3	[0, 0, 0, 1, 0]
Sunny	4	[0, 0, 0, 0, 1]

Table 3.6: One Hot Encoding

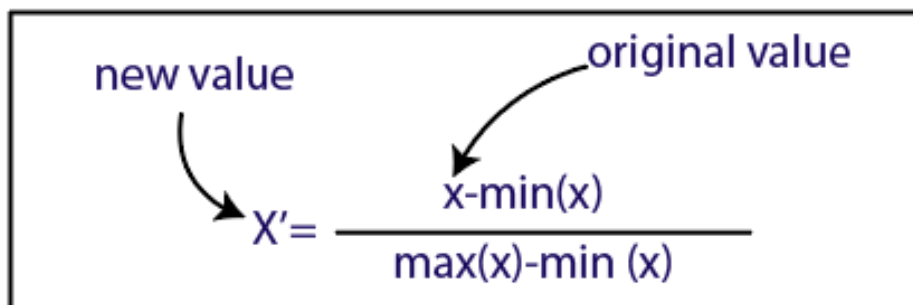
### Stage 5: Splitting the dataset

Every dataset for Machine Learning model must be split into training/test set. Training set is used for training our model. Here, the model knows the output. A test set, on the other hand, is used for testing the model for checking how accurate the model is. The ML model uses the test set to predict outcomes.

Usually, we take 70% or 80% of the data for training the model while leaving out the rest 30% or 20%. For our work we use 80:20 ratios.

### Stage 6: Feature scaling

Feature scaling marks the end of the data preprocessing in Machine Learning. It is a method to standardize the independent variables of a dataset within a specific range. In other words, feature scaling limits the range of variables so that you can compare them on common grounds [120]. There are methods to do the feature scaling step depending on the data such as Standardization, Normalization, and Mean normalization. We work with images so we used Normalization method which also called Re-scaling.



The diagram shows the normalization formula:  $X' = \frac{x - \min(x)}{\max(x) - \min(x)}$ . An arrow labeled "original value" points from the variable  $x$  in the numerator to the text "original value". Another arrow labeled "new value" points from the variable  $X'$  to the text "new value".

Figure 3.11: Normalization formula [120]

Generally, normalization consists in re-scaling the range of features to scale the range in  $[0, 1]$  or  $[-1, 1]$ .

For our case images, pixels range is  $[0-255]$ , when we apply the formula the new value  $X'$  always will be the original value divided by 255.

### C- Training

The heart of our process is the training of the model that requires patience and experimentation. Most of the “learning” is done at this stage. First thing to do for our training process is setting up convolutional neural networks model architecture (choosing the model) then we will use our pre-processed data to incrementally improve our model’s ability to predict whether a given image is one of the five weather condition hoping to get an accurate model after we evaluate the model.

### D- Model evaluation

In this step we need to evaluate our trained model with the real world data (data unseen before by the model). However, through its training, the model should be capable enough to extrapolate the information and deem whether the weather is cloudy or foggy or rainy or snowy or sunny. Else if our model is not good enough with real data we need to tuning hyperparameter or re-design our CNN architecture then retrain our model with the new settings.

- Accuracy is a common evaluation metric for classification problems. It's the number of correct predictions made as a ratio of all predictions made.
- A confusion matrix provides a more detailed breakdown of correct and incorrect classifications for each class.

The short explanation of how to interpret a confusion matrix is as follows: The diagonal elements represent the number of points for which the predicted label is equal to the true label, while anything off the diagonal was mislabeled by the classifier. Therefore, the higher the diagonal values of the confusion matrix the better, indicating many correct predictions [123].

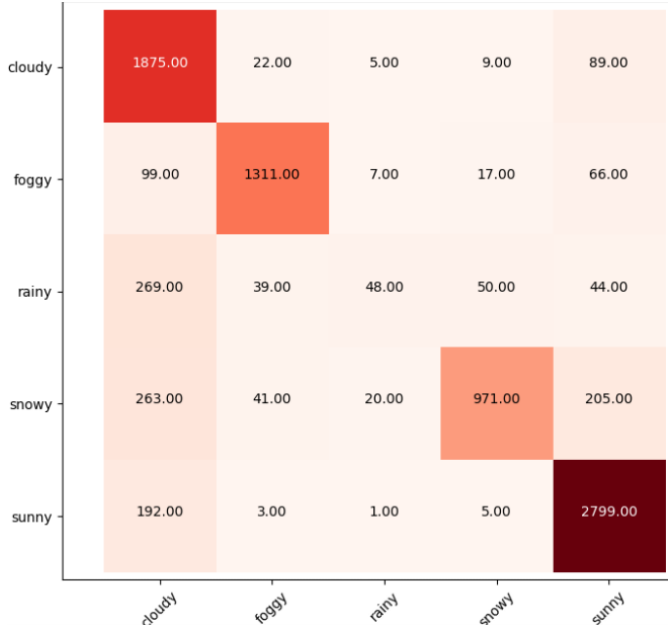


Figure 3.12: Confusion matrix example

**E- Hyperparameter tuning**

hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. The goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to improve model performance. Some techniques used to tune hyperparameters are Grid Search, Random Search, Bayesian Optimization or manually [124]. For our work we fine tuning manually based on the first choice of hyperparameters and their score, we change a part of them (or all of them) then train the model again, and check the difference in the score. Some CNN Hyperparameters: learning rate, batch size, epoch number, activation function, etc.

## F- Model Prediction

The final step of our work is prediction. Now we consider our model to be ready for practical applications. This is the point of all this work, where the value of our CNN model is realized.

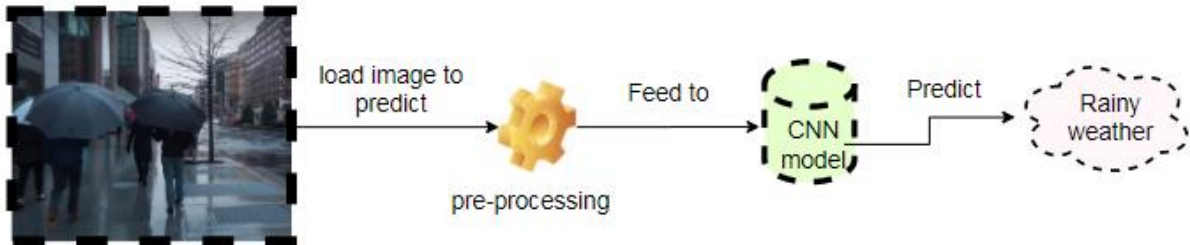


Figure 3.13: Prediction example

Note! For the pre-processing step is required as the user provided image should be resized and scaled in the same way as the pre-processing in training stage otherwise the model will misclassified the labels .

### 3.2.2.2 Front-end

The purpose of this part is making the interactions (request/ response) for converting data from back-end to the graphical interface so that users can view and interact with that data by simplify the access to each part of our application. As shown in the figure 3.13:

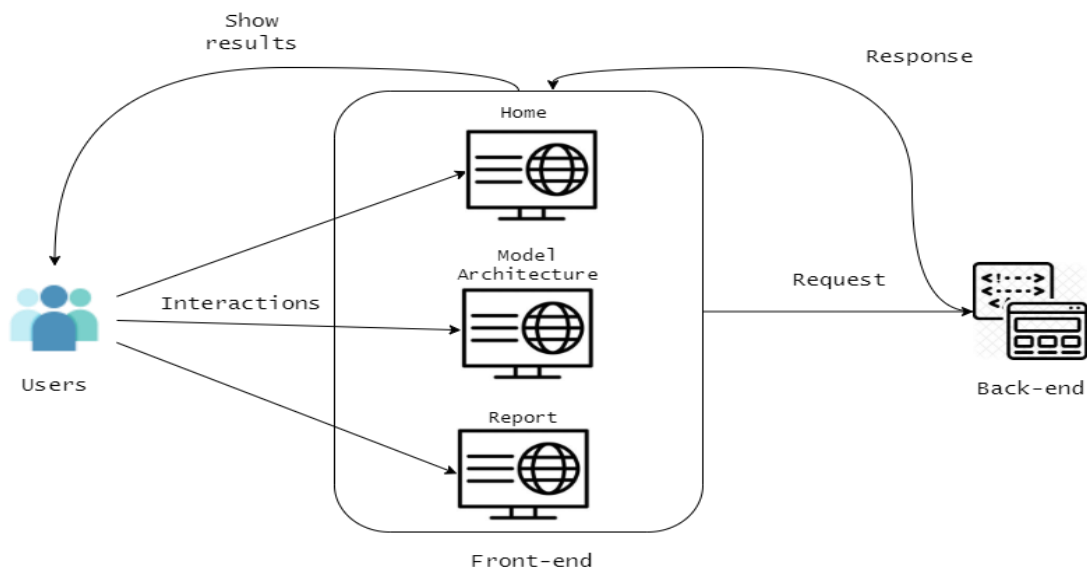


Figure 3.14: Front-end system

## 3.3 Environments and Implementation

### 3.3.1 Environments and tools

To achieve our purpose of our work we need a different environments and tools for the backend from the programming language, APIs, libraries, IDE. For the frontend we are going to use programming language, style sheet, markup language, text editor, IDE...etc

### 3.3.1.1 Back-end Environments and tools

#### Python



Figure 3.15: Python logo

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. We use Python 3.6 for our project.

#### PyCharm



Figure 3.16: Pycharm logo

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.

#### TensorFlow





Figure 3.17: TensorFlow 2.0 logo

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

### **Keras**



Figure 3.18: Keras logo

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the Xception deep neural network model.

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0.

### **OpenCV**



Figure 3.19: OpenCV logo

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

### **Matplotlib**



Figure 3.20: Matplotlib logo

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

### **NumPy**



Figure 3.21: Numpy logo

NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

### **Scikit-learn**



Figure 3.22: Scikit-learn logo

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

### Flask



Figure 3.23: Flask logo

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

## Google Colab



Figure 3.24: Google Colab logo

Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

### 3.3.1.2 Front-end Environments and tools

#### JavaScript



Figure 3.25: JavaScript logo

JavaScript often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

#### HTML



Figure 3.26: HTML5 logo

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

## CSS



Figure 3.27: CSS3 logo

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

## Sublime text



Figure 3.28: Sublime text logo

Sublime Text is a shareware cross-platform source code editor with a Python application programming interface. It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses.

### 3.3.2 Back-end development

This section is about backend development (training CNN), we going through the training phase including pre-processing steps using the programming language Python with the help of deep learning libraries and API's.

#### 3.3.2.1 Import libraries

First step is import what we need from python libraries to start our deep learning training, this are libraries used in our training as shown in list 3.1:

```

1. import os
2. import numpy as np
3. import tensorflow as tf
4. from keras import regularizers
5. from keras.preprocessing.image import ImageDataGenerator
6. from keras.models import Sequential
7. from keras.layers import Dropout, Flatten, Dense, BatchNormalization
8. from keras.layers.convolutional import Conv2D, MaxPooling2D
9. from keras import callbacks
10. from keras.optimizers import Adam
11. import time
12. import pickle

```

Listing 3.1: Import Libraries.

### 3.3.2.2 Dataset loading

The second step is loading our dataset that going to be used in the training, but since we don't have an npy or csv files for our dataset ( we have images in folders) we need to create our data.npy to simplify the pre-processing steps with a simple python code as shown in list 3.2:

```

1. training_path = 'D:/weather_model/fdata/training'
2. def training_data():
3.     train = []
4.     train_label = []
5.     for subdir, dirs, files in os.walk(training_path):
6.         for file in files:
7.             name_ext = os.path.basename(file)
8.             img_name, image_extension = os.path.splitext(name_ext)
9.             image = cv2.imread(os.path.join(subdir, file))
10.            try:
11.                image = cv2.resize(image, (224, 224))
12.            except Exception as e:
13.                print(str(e))
14.            if "cloudy" in img_name:
15.                train.append(image)
16.                train_label.append("cloudy")
17.            elif "foggy" in img_name:
18.                train.append(image)
19.                train_label.append("foggy")
20.            elif "rainy" in img_name:
21.                train.append(image)
22.                train_label.append("rainy")
23.            elif "snowy" in img_name:
24.                train.append(image)
25.                train_label.append("snowy")
26.            elif "sunny" in img_name:
27.                train.append(image)
28.                train_label.append("sunny")
29.        train_data = np.array(train)
30.        train_labels = np.array(train_label)
31.        np.save('train', train_data)
32.        np.save('train_labels', train_labels)
33.
34.    training_data()

```

Listing 3.2: From images to npy file.

After we prepare our training data and validation data as well in npy files, the dataset already to load as described in the following code:

```
1. train_data = np.load('drive/My Drive/weather/train.npy')
2. train_labels = np.load('drive/My Drive/weather/train_labels.npy')
3.
4. valid_data = np.load('drive/My Drive/weather/validation.npy')
5. valid_labels = np.load('drive/MyDrive/weather/validation_labels.npy')
```

Listing 3.3: Loading dataset.

For loading the dataset we read them by calling a method that read npy files with a giving path of the dataset (npy files saved before).

### 3.3.2.3 Dataset pre-processing

The third step to train our model we need first to pre-process our dataset by following the next steps:

1. Data normalization: we need to rescale our images from the range [0-255] to the range [0-1] using the min-max normalization method by dividing each pixel value of the image by 255.
2. Encoding Categorical Labels: in our labels npy files we have categorical values and that's problem in machine learning models since they're a mathematical models so we need to encode our Labels. We use OneHotEncoder from Scikit-learn library, the output would a binary vector for each class.
3. Reshaping Input Images: our images with input (224x224x3), before feed them to CNN for training we need to reshape them to 4D tensors. So input data has a shape of (number of samples, 224, 224, 3), where the first dimension represents the batch size of the image.

```
1. # data normalization
2. train_data = train_data/255
3. validation_data = validation_data/255
4. #Encoding Categorical train/validation Labels
5. train_ = array(train_labels)
6. val_ = array(val_labels)
7. # integer encode
8. encoder=LabelEncoder()
9. encoded1=encoder.fit_transform(train_)
10. encoded2=encoder.fit_transform(val_)
11. # binary vector encode
12. onehot_encoder = OneHotEncoder(categories='auto')
13. encoded1=encoded1.reshape(len(encoded1), 1)
14. encoded2=encoded2.reshape(len(encoded2), 1)
15. train_labels = onehot_encoder.fit_transform(encoded1)
16. val_labels = onehot_encoder.fit_transform(encoded2)
17. #Reshaping data
18. train_data = train_data.reshape(train_data.shape[0],
19. img_width, img_height, 3)
20. val_data = validation_data.reshape(validation_data.shape[0],
21. img_width, img_height, 3)
```

Listing 3.4: Dataset pre-processing.

### 3.3.2.4 Model training

Since our dataset is ready now we going to design convolutional neural network model architecture using sequential model from Keras with setting configurations as shown in the next code:

```
1. model = Sequential()
2. model.add(Conv2D(32, kernel_size=(3,3), input_shape=(img_width,
3. img_height,3), activation='relu'))
4. model.add(MaxPooling2D(pool_size=(2, 2)))
5. model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
6. model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
7. model.add(MaxPooling2D(pool_size=(2, 2)))
8. model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
9. model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
10. model.add(MaxPooling2D(pool_size=(2, 2)))
11. model.add(Flatten())
12. model.add(Dense(512, activation='relu'))
13. model.add(Dropout(0.7))
14. model.add(Dense(classes_num, activation='softmax'))
```

Listing 3.5: Model architecture.

The next thing thing to do is begin our training session as the following:

```
1. model.compile(loss='categorical_crossentropy',
2.               optimizer=Adam(lr=0.0001),
3.               metrics=['accuracy'])
4. checkpoint_filepath = 'drive/My Drive/weather_model/_ck0.ckpt'
5. my_callbacks = [
6.     EarlyStopping(monitor='val_loss',
7.                  mode='min',
8.                  verbose=1,
9.                  patience=20),
10.    ModelCheckpoint(filepath=checkpoint_filepath,
11.                   verbose=1,
12.                   monitor='val_accuracy',
13.                   mode='max',
14.                   save_best_only=True)
15. ]
16. model.fit(
17.     train_data,
18.     train_labels,
19.     validation_data=(validation_data, validation_labels),
20.     batch_size=64,
21.     epochs=100,
22.     verbose=2
23. )
24. model.save('drive/My Drive/weather/colab_model0.h5')
```

Listing 3.6: Model training.



Here we are using Adam optimizer, we launch our training with 100 epochs and 64 batch size. Also we use other parameters (Early stopping that stop the training after certain number of epochs if the loss of our model didn't improve to prevent overfitting),(checkpoint save the model with the max accuracy during the training). Finally we save the trained model.

### 3.3.2.5 Model evaluation

Evaluation is the stage when we check whether the model is best fit for the given new data unseen before. Keras model provides a function, evaluate to check the model accuracy and loss.

```
1. score = model.evaluate(test_data, test_labels, verbose = 0)
2. print('Test loss:', score[0])
3. print('Test accuracy:', score[1])
```

Listing 3.7: Model evaluation.

After checking the accuracy and loss ( we also use confusion matrix to evaluate our model) of the model we decide if our model is ready for predictions or we need to go back to the training stage for fine tuning hyperparameters or reDesign CNN architecture.

### 3.3.3 Front-end development

The reason for this part is to make our work easy to work with from any user want to test and use the application. We use a simple structure for the pages; all the pages are accessible from one to another.

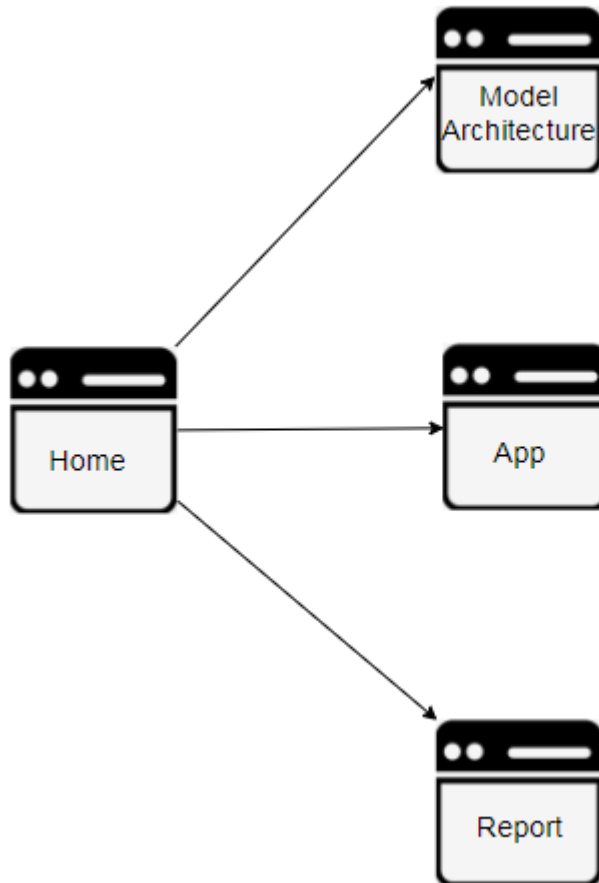


Figure 3.29: Front-end architecture

We use the HTML for creating and structure the content of the pages also taking the advantages of styling the pages from CSS to make the presentation of Web pages, including colors, layout, and fonts. For making a good look to the user eye. We use JavaScript to create responsive, interactive, dynamic elements for our pages. and make it easier and more attractive for enhancing the user experience. As shown in the next figures for the application:

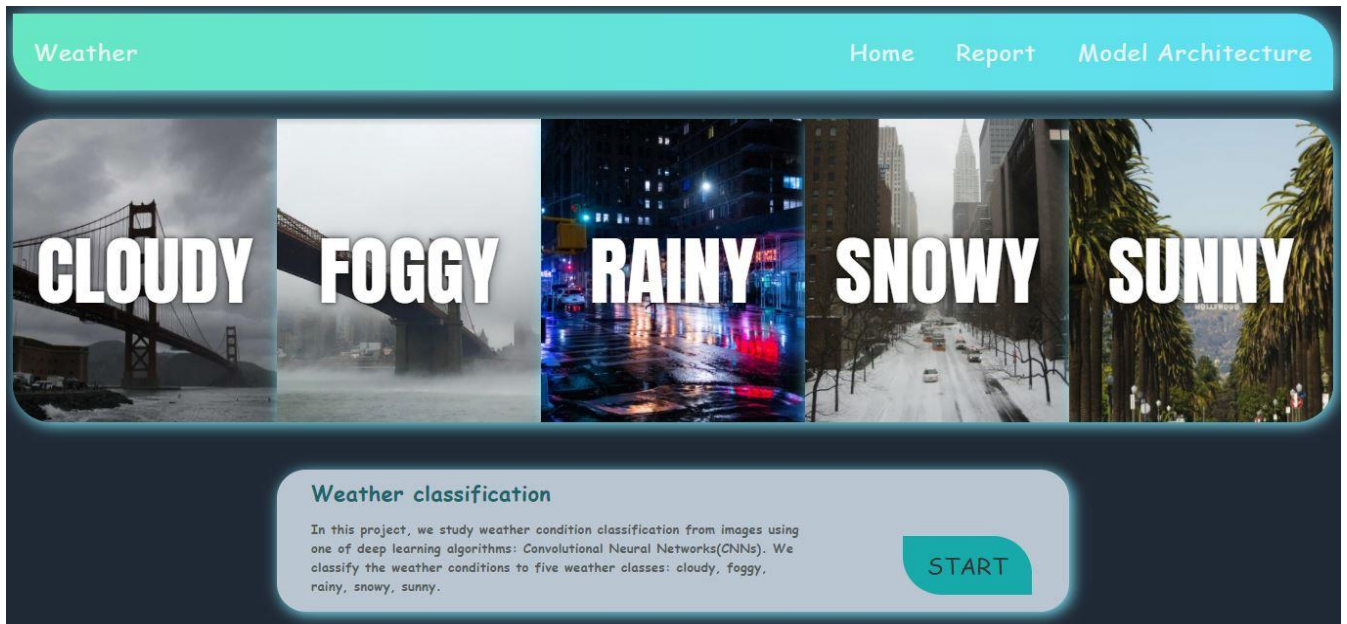


Figure 3.30: App Page 1

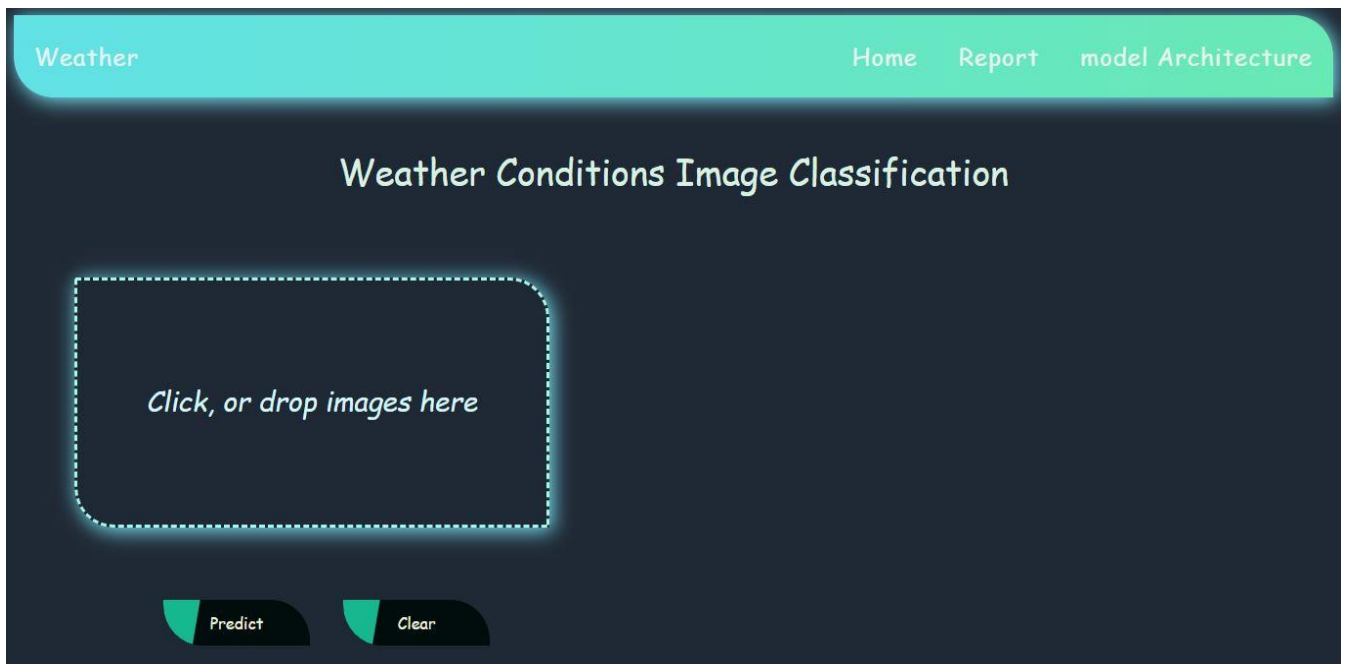


Figure 3.31: App Page 2A.

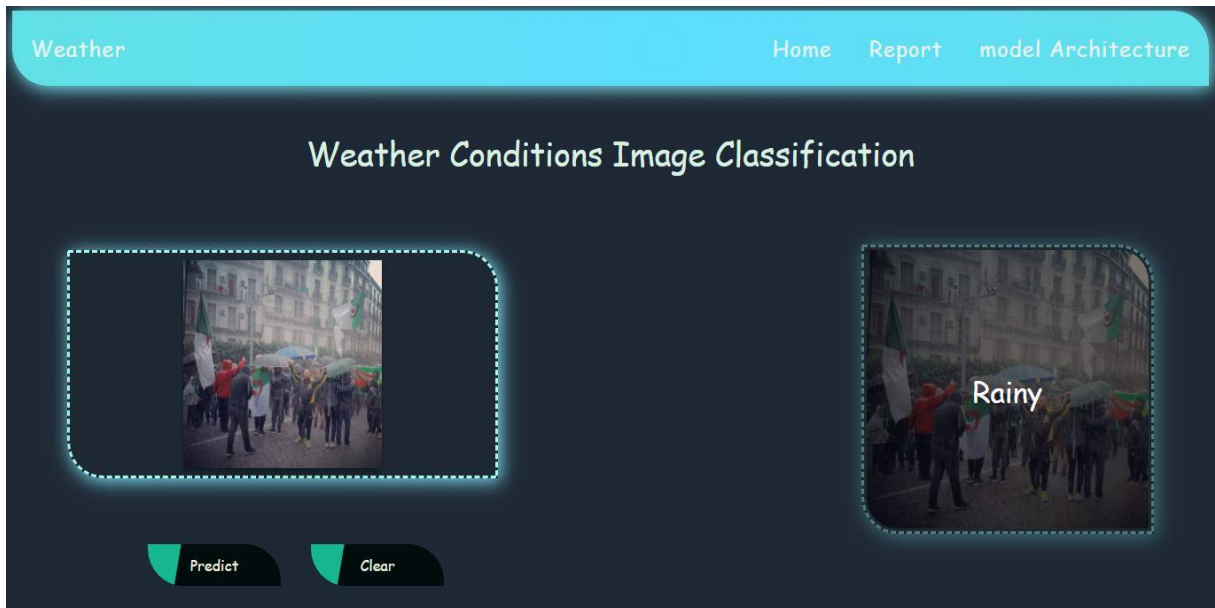


Figure 3.32: App Page 2B

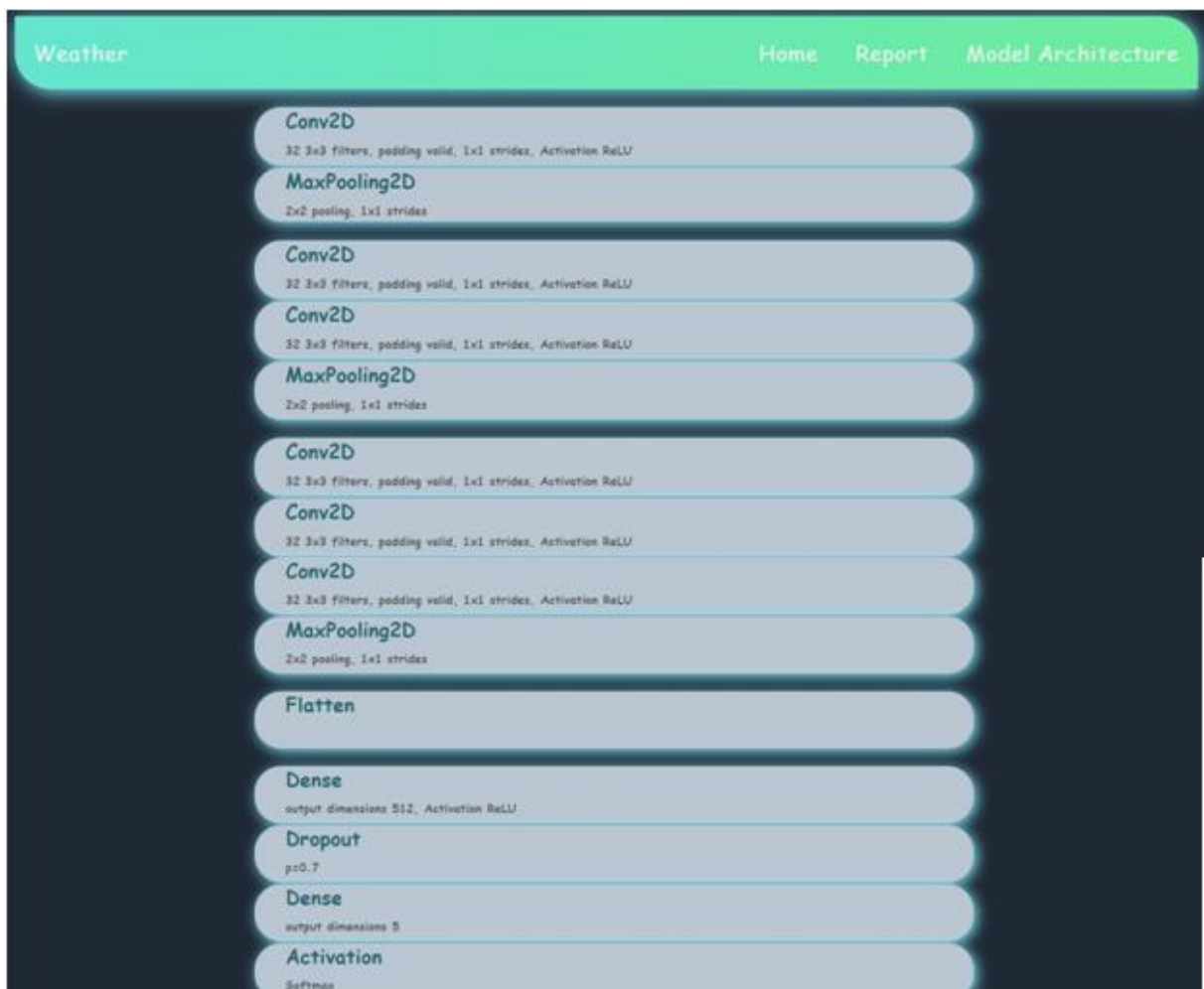


Figure 3.33: App Page 3

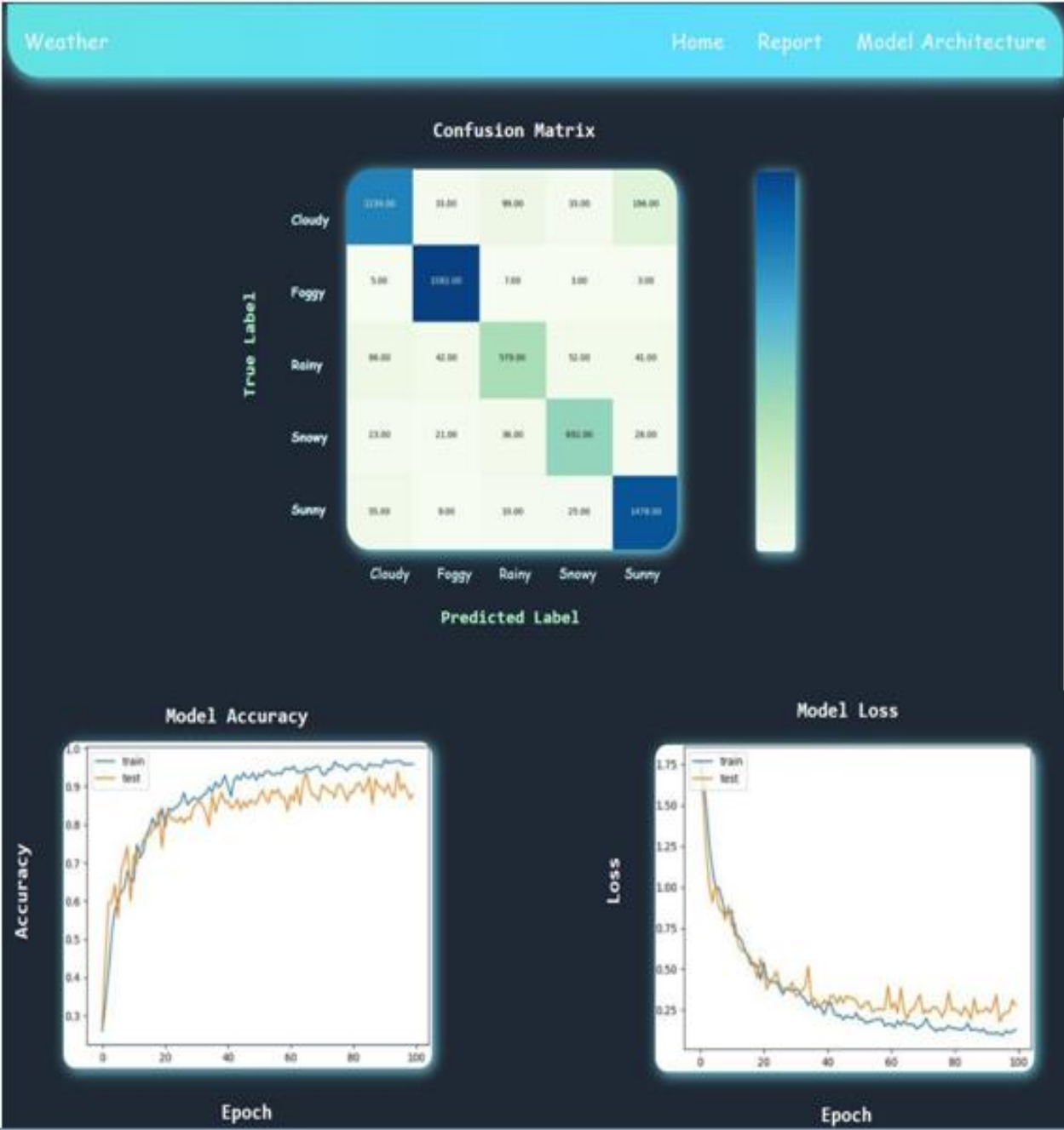


Figure 3.34: App Page 4

### 3.4 Results and discussion

In order to train our dataset we need a paralleling computing with minimum resources to gain time and train different CNN architectures and that what Google providing us with free cloud service (Google Colab) that supports free GPU, Google Colab providing us a single 12GB NVIDIA Tesla K80 GPU that can be used up to 12 hours continuously.

#### 3.4.1 Results

Architecture	Optimizer	Training_time (colab GPU)	Loss %	Accuracy %
1/Conv2D-MaxPool-Conv2D-MaxPool-FC-FC	SGD	10 min	70%	63%
2/Conv2D-Conv2D-MaxPool-Conv2D-MaxPool-FC-FC	SGD	15 min	60%	64%
3/Conv2D-MaxPool-Conv2D-MaxPool-FC-FC	RMSPROP	11 min	56%	69%
4/Conv2D-MaxPool-Conv2D-MaxPool-Conv2d-MaxPool-FC-FC-FC	RMSPROP	20 min	61%	72%
5/Conv2D-Conv2D-MaxPool-Conv2D-FC-FC	RMSPROP	17 min	50%	70%
6/Conv2D-MaxPool-Conv2D-FC-FC-FC	Adam	13 min	45%	73%
7/Conv2D-Conv2D-Conv2D-MaxPool-FC-FC-FC	Adam	28 min	62%	79%
8/Conv2D-Conv2D-Conv2D-MaxPool-Conv2D-MaxPool-FC-FC-FC	Adam	35 min	52%	81%
9/Conv2D-MaxPool-Conv2D-Conv2D-Maxpool-Conv2D-Conv2D-MaxPool-FC-FC	Adam	50 min	30%	86%

Table 3.7: Weather conditions classification tests

#### Test 1:

We used a simple CNN architecture ( 2 convolution, 2 maxpooling, 2 fullyconnected), with ‘SGD’ optimize,20 epoch, 64 batch size we get in 10 minutes about 63% accuracy and 70% loss which is a high loss value lead to a lot of misclassifications in predictions.

#### Test 2:

We added a convolution layer in this test and kept the same hyperparameters from the first architecture the loss value decreased which is a good sign but the accuracy just increased about 1% and that’s not the improvement we want.

**Test 3:**

We used the same simple architecture and hyperparameters in the first test with changing the optimizer to 'RMSprop' our accuracy is up to 69% and also the loss down to 56% which is a good sign for now.

**Test 4:**

We kept the same optimizer 'RMSprop' but we change the architecture here by adding ( 1 convolution layer, 1 maxpooling, 1 fullyconnected), the training time increased to 20 minutes we get good accuracy about 72% but the loss up to 61% which is a bad results for us.

**Test 5:**

We simplified the architecture in order to get better results but the only good sign for this changing is that the loss is down for the first time to 50% with just 70% accuracy.

**Test 6:**

We used a simple architecture as the first test but with another optimizer which is 'Adam', the loss down to 45% which is a good thing for the model but the accuracy doesn't get better to much about just 73%.

**Test 7:**

We used 3 convolution layers, 1 maxpooling, and 3 fullyconnected layers with the same optimizer the accuracy is increased to 79% but the loss increased to much about 62% and that is another problem in the model.

**Test 8:**

We used in this test (3 Conv,1 Mxpool, 1 Conv, 1 Mxpool, 3 FC) the loss value down to 59% good sign for our model also the accuracy for the first time pass above 80%

**Test 9:**

We increased the complexity of the architecture and we change the batch size from 64 to 128, we get remarkable improvement, the loss down to 30% and the accuracy up to 86%.

Based on the test 9 which is quite good model, we want to decreased the loss value. So for our last test we kept the same architecture but we increased the dropout probability from 0.5 to 0.8, also we changed the learning rate in 'adam' optimizer from 0.001 to 0.0001, we get about 90% accuracy with a 22% loss in 100 epoch.

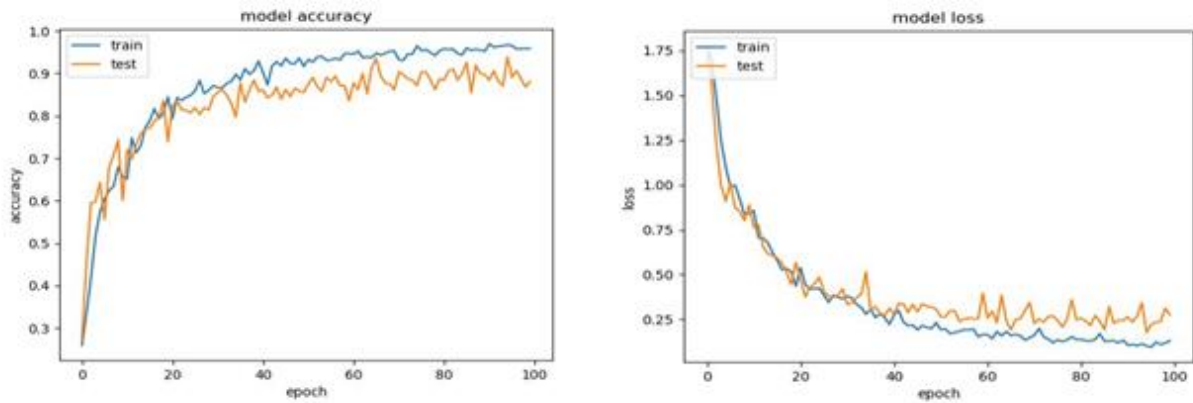


Figure 3.35: Accuracy and Loss graphs

We tested our model on 500 data images unseen before, 100 images for each weather conditions:

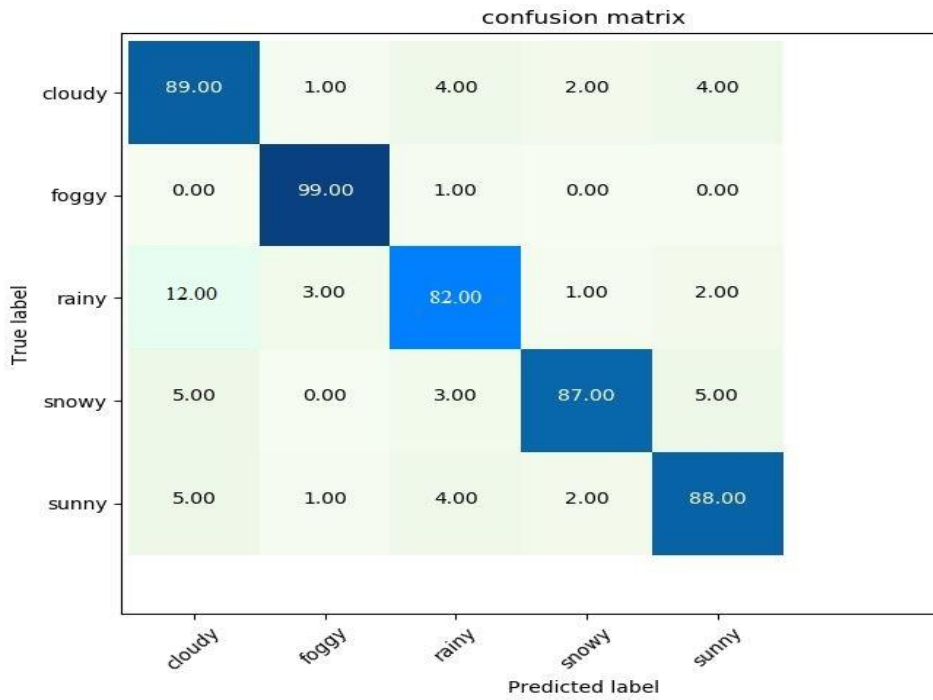


Figure 3.36: Confusion Matrix

The results of the confusion matrix for testing new images show good results as following:

- **Cloudy:** 89 from 100 images predicted true ( 89% accuracy, 11% error rate)
- **Foggy:** 99 from 100 images predicted true ( 99% accuracy, 1% error rate)
- **Rainy:** 82 from 100 images predicted true ( 82% accuracy, 18% error rate)
- **Snowy:** 87 from 100 images predicted true ( 87% accuracy, 13% error rate)
- **Sunny:** 88 from 100 images predicted true (88% accuracy, 12% error rate)

We conclude that the model had an accuracy 89% and 11% error rate for new data and this is a good result.



### 3.4.2 Results discussion and comparison:

Model	Year	Weather classes	Accuracy
1/ Weather Classification: A new multi-class dataset, data augmentation approach and comprehensive evaluations of CNN [112].	2018	cloudy-foggy-rainy-snowy-sunny 5 Classes	<b>80%</b>
2/ WeatherNet: Recognising Weather and Visual Conditions from Street-Level Images Using Deep Residual Learning [113].	2019	Clear-rain-snow 3 Classes	<b>93%</b>
3/ A Framework for the Classification Task Recognizing Weather Condition in an Image using Supervised Learning Methods [125].	2020	cloudy-foggy-rainy-snowy-sunny 5 Classes	<b>86%</b>
4/ ResNet15: Weather Recognition on Traffic Road with Deep Convolutional Neural Network [126].	2020	foggy-rainy-snowy-sunny 4 classes	<b>96%</b>
5/ <b>Our CNN Model</b>	2020	cloudy-foggy-rainy-snowy- sunny 5 Classes	<b>90%</b>

Table 3.8: Comparison weather recognition works

We have built a supervised deep learning model CNN can recognize weather conditions for five classes: cloudy, foggy, rainy, snowy, sunny. Our model trained on different weather images locations (nature scenes, surveillance cameras, street-level, urban traffic, synthesis images).

- ✓ Our CNN model can reach 90% accuracy testing on 4000 images to five classes which is a convenient results comparing to the model one (2018, 5 classes) and the model three (2020, 5 classes) which there accuracies about 80% and 86%.
- ✓ Comparing our model to the model two (2019, 3 classes) from the perspective of accuracy is better than ours but we can notice that this model detect only 3 classes unlike ours which detect 5 classes, the same thing for the model four (2020, 4 classes) better accuracy with less weather conditions to detect.

Also the models two and four are trained on level-street/urban traffic images and that's mean there is a possibility to misclassify weather images from different types such as nature scenes or surveillance cameras. In the other hand our model trained images from different locations and we can notice the absence of cloudy class because cloudy class is tricky and hard to distinguish from other weather conditions (due to that all other weather conditions also involve clouds).

### **3.4.3 Limitation**

With the good results we had our system for weather condition classification still missing some points as the following:

- Our system can't classify weather conditions into multi-label classification such as: cloudy-rainy, cloudy-snowy, cloudy-sunny, sunny-snowy, foggy-rainy.....etc
- We have a quite high loss with 22% this lead to misclassify the weather.
- Such system needs more informations such as time (day/night) and visual conditions.
- This model won't recognize weather with bad quality images.

### **3.4.4 Future works**

In the future, from our methodology there's a chance to develop a powerful system for weather recognition based on our work by improving some points as following:

- Make our system able to classify weather conditions to multi-label (more than one weather condition in a single image) classification.
- Add the time conditions (day/night/sunshine/sunrise for example) to the system.
- Experimenting with transfer learning technique using developed CNN architecture such as ResNet, VGG16, GoogleNet...etc. may enhance the accuracy of the model.

### **3.5 Conclusion**

In this chapter we presented the architecture of our system also the way how we prepare dataset to train a CNN model for weather condition classification.

Among the related works we discuss with our achievement we can tell that our model is a promising model for weather recognition with some modification In the future.

# General Conclusion

Weather conditions often play a valuable role in the proper functioning of transportation systems, agriculture, and other systems. Present solutions either deploy sensors or use an in-vehicle camera to predict weather conditions. However, the installation, configuration of sensors will cost a lot of manpower and material resources and those solutions are used in a limited scope.

To ensure the knowledge about all-weather conditions, an automation detection system is necessary to classify weather conditions. Due to this we have used a supervised deep learning method by proposing accurate convolutional neural network architecture can detect weather conditions from images.

The aim of our system is to make it as new or alternative solution for the current means in weather recognition tasks in anywhere and at anytime taken benefits from deep learning CNN model that helped to extract complex informations from single-images for five weather conditions cloudy, foggy, rainy, snowy, and sunny.

Such systems as an example can avoid a lot of transportation accidents in highways, and it can help to organize urban traffics and improve driving efficiency. Some driver assistant can improve driving safety through weather detection, such slow down speed in bad weather conditions, warning drivers to keep distance with other cars, and automatically turning on wipers in rainy weather.

## Bibliography

1. <https://people.cs.clemson.edu/~dhouse/courses/405/notes/pixmaps-rgb.pdf>
2. <https://en.wikipedia.org/wiki/Pixel>
3. <https://towardsdatascience.com/introduction-to-images-c9c7abe6bfd2>
4. <https://oneroomwithaview.com/2017/05/16/12-angry-men-still-relevant/>
5. Colour Design, Theories and Applications by Best, Janet
6. [cgi.csc.liv.ac.uk/~frans/CurrentResearch/Thessi/hanafiJHijazi2012-7-22.pdf](http://cgi.csc.liv.ac.uk/~frans/CurrentResearch/Thessi/hanafiJHijazi2012-7-22.pdf)
7. <https://www.livescience.com/32559-why-do-we-see-in-color.html>
8. <https://en.wikipedia.org/wiki/Color>
9. [https://en.wikipedia.org/wiki/Color\\_vision](https://en.wikipedia.org/wiki/Color_vision)
10. [https://en.wikipedia.org/wiki/Visible\\_spectrum](https://en.wikipedia.org/wiki/Visible_spectrum)
11. <https://www.thoughtco.com/understand-the-visible-spectrum-608329>
12. [https://en.wikipedia.org/wiki/Color\\_models](https://en.wikipedia.org/wiki/Color_models)
13. <https://upload.wikimedia.org/wikipedia/commons/5/5b/Cmyk-rgb.jpg>
14. <https://ieeexplore.ieee.org/document/8295029>
15. <https://www.bookdepository.com/Deep-Learning-for-Vision-Systems-Mohamed-Elgendy/9781617296192>
16. <https://machinelearningmastery.com/applications-of-deep-learning-for-computer-vision/>
17. <https://cs.stackexchange.com/questions/7050/what-are-the-differences-between-computer-vision-and-image-processing>
18. [https://www.researchgate.net/publication/4260042\\_Feature\\_Mining\\_for\\_Image\\_Classification](https://www.researchgate.net/publication/4260042_Feature_Mining_for_Image_Classification)
19. [https://www.byclb.com/TR/Tutorials/neural\\_networks/ch1\\_1.htm](https://www.byclb.com/TR/Tutorials/neural_networks/ch1_1.htm)
20. <https://www.fritz.ai/image-recognition/>
21. <https://arxiv.org/pdf/1705.09435.pdf>
22. <https://medium.com/@ManningBooks/how-does-computer-vision-work-bc35b0fb5df5>
23. <https://chsasank.github.io/keras-tutorial.html>

24. <https://www.coursehero.com/file/p1q0d81/C-H-A-P-T-E-R-2-Features-and-Classi%EF%AC%81ers-Feature-extraction-and-classi%EF%AC%81cation/>
25. <https://freecontent.manning.com/the-computer-vision-pipeline-part-4-feature-extraction/>
26. [https://en.wikipedia.org/wiki/Feature\\_\(machine\\_learning\)](https://en.wikipedia.org/wiki/Feature_(machine_learning))
27. <https://www.coursehero.com/file/pcjjcta/Figure-21-a-The-aim-is-to-design-an-algorithm-which-classifies-input-images/>
28. <https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/>
29. <https://towardsdatascience.com/image-pre-processing-c1aec0be3edf>
30. <https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/>
31. <https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4>
32. <https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11>
33. <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>
34. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
35. <https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac>
36. [https://www.researchgate.net/profile/Ming\\_Ding15/publication/331723697\\_The\\_Impact\\_of\\_Adverse\\_Weather\\_Conditions\\_on\\_Autonomous\\_Vehicles\\_Examining\\_how\\_rain\\_snow\\_fog\\_and\\_hail\\_affect\\_the\\_performance\\_of\\_a\\_selfdriving\\_car/links/5e644b8d92851c7ce04f094c/The-Impact-of-Adverse-Weather-Conditions-on-Autonomous-Vehicles-Examining-how-rain-snow-fog-and-hail-affect-the-performance-of-a-self-driving-car.pdf?origin=publication\\_detail](https://www.researchgate.net/profile/Ming_Ding15/publication/331723697_The_Impact_of_Adverse_Weather_Conditions_on_Autonomous_Vehicles_Examining_how_rain_snow_fog_and_hail_affect_the_performance_of_a_selfdriving_car/links/5e644b8d92851c7ce04f094c/The-Impact-of-Adverse-Weather-Conditions-on-Autonomous-Vehicles-Examining-how-rain-snow-fog-and-hail-affect-the-performance-of-a-self-driving-car.pdf?origin=publication_detail)
37. [https://www.researchgate.net/publication/326799649\\_Weather\\_Classification\\_A\\_new\\_multiclass\\_dataset\\_data\\_augmentation\\_approach\\_and\\_comprehensive\\_evaluations\\_of\\_Revolutional\\_Neural\\_Networks](https://www.researchgate.net/publication/326799649_Weather_Classification_A_new_multiclass_dataset_data_augmentation_approach_and_comprehensive_evaluations_of_Revolutional_Neural_Networks)
38. <https://www.m3aerial.com/blog/tag/understand+how+weather+can+affect+operation+of+drones>
39. <https://www.aeromotus.com/information-for-drone-users/how-weather-affects-your-drone-s-performance/>
40. <https://www.thatsclever.co.uk/drones-vs-bad-weather/>
41. [https://www.researchgate.net/publication/303097134\\_Scene-free\\_Multi-class\\_Weather\\_Classification\\_on\\_Single\\_Images](https://www.researchgate.net/publication/303097134_Scene-free_Multi-class_Weather_Classification_on_Single_Images)
42. <https://drive.google.com/file/d/1HFyAUvnkS61Xat9cUBAhG-4hvwR1T8lb/view>

43. "Rainy Weather Recognition from In-Vehicle Camera Images for Driver Assistance", Hiroyuki Kurihata;Tomokazu Takahashi;Ichiro Ide;Yoshito Mekada;Hiroshi Murase;Yukimasa Tamatsu;Takayuki Miyahara
44. "Brightness Calculation in Digital Image Processing" , Bezryadin, Sergey; Bourov, Pavel; Ilinih, Dmitry , 1st International Symposium on Technologies for Digital Photo Fulfillment, pp. 10-15(6)
45. <https://deepai.org/machine-learning-glossary-and-terms/neural-network>
46. <https://slideplayer.com/slide/12875315/>
47. <https://www.quora.com/Can-we-have-a-single-hidden-layer-in-a-multilayer-perceptron>
48. <https://www.oreilly.com/library/view/neural-networks-with/9781788397872/eb49931a-7122-4b97-a843-bb6e1817cc12.xhtml>
49. <https://medium.com/datadriveninvestor/exploring-machine-learning-f1dc6f3ec902>
50. <https://machinelearningmastery.com/types-of-learning-in-machine-learning/>
51. <http://incompleteideas.net/book/bookdraft2017nov5.pdf>
52. [https://en.wikipedia.org/wiki/Reinforcement\\_learning](https://en.wikipedia.org/wiki/Reinforcement_learning)
53. <https://intellipaat.com/community/44608/what-is-agent-in-reinforcement-learning>
54. <https://ieeexplore.ieee.org/document/6813505>
55. <https://ieeexplore.ieee.org/document/8451098/>
56. <https://www.investopedia.com/terms/d/deep-learning.asp>
57. [http://beamandrew.github.io/deeplearning/2017/02/23/deep\\_learning\\_101\\_part1.html](http://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html)
58. <http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/>
59. <https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798>
60. <https://www.simplilearn.com/deep-learning-algorithms-article>
61. <https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html>
62. <https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-machine-learning-deep-learning-architectures/>
63. <https://iopscience.iop.org/article/10.1088/0957-0233/26/11/115002>
64. <https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29>

65. [https://www.researchgate.net/figure/Generative-Adversarial-Network-Architecture\\_fig3\\_334100947](https://www.researchgate.net/figure/Generative-Adversarial-Network-Architecture_fig3_334100947)
66. <https://searchenterpriseai.techtarget.com/definition/recurrent-neural-networks>
67. <https://www.arm.com/glossary/recurrent-neural-network>
68. <https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9>
69. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
70. <https://www.cs.cornell.edu/~oirsoy/files/nips14drsv.pdf>
71. [https://en.wikipedia.org/wiki/Recursive\\_neural\\_network](https://en.wikipedia.org/wiki/Recursive_neural_network)
72. <https://www.hindawi.com/journals/aaa/2013/617618/fig3/>
73. <https://medium.com/nybles/a-brief-guide-to-convolutional-neural-network-cnn-642f47e88ed4>
74. <https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network>
75. <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/>
76. <https://medium.com/@sdoshi579/convolutional-neural-network-learn-and-apply-3dac9acfe2b6>
77. <https://towardsdatascience.com/understanding-input-and-output-shapes-in-convolution-network-keras-f143923d56ca>
78. <https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>
79. <https://cs231n.github.io/convolutional-networks/>
80. <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>
81. <https://towardsdatascience.com/visualising-filters-and-feature-maps-for-deep-learning-d814e13bd671>
82. <https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/>
83. <https://medium.com/machine-learning-algorithms/what-is-stride-in-convolutional-neural-network-e3b4ae9baedb>
84. <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/>

85. <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>
86. <https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7>
87. <https://confusedcoders.com/data-science/deep-learning/cnn-with-tensorflow-for-deep-learning-beginners>
88. <https://www.semanticscholar.org/paper/Implications-of-Pooling-Strategies-in-Convolutional-Sharma-Mehra/cc07d2fe2884fdd2a50482c6a5c0111c4195fddc>
89. <https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/>
90. <https://medium.com/school-of-ai-trivandrum/understanding-cnn-completely-bd5bb861c11d>
91. <https://www.jeremyjordan.me/convnet-architectures/>
92. <https://mc.ai/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet/>
93. <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/>
94. <https://medium.com/deep-learning-demystified/generalization-in-neural-networks-7765ee42ac23>
95. <https://elitedatascience.com/overfitting-in-machine-learning#how-to-prevent>
96. [https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)
97. <https://towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a>
98. <https://www.ijert.org/deep-learning-technology-a-vital-tool-for-national-development>
99. <https://www.aidoc.com/blog/deep-learning-in-healthcare/>
100. <https://robots.net/tech-reviews/deep-learning-applications/>
101. <https://arxiv.org/pdf/1804.01149.pdf>
102. <https://www.ijitee.org/wp-content/uploads/papers/v8i2s/BS2713128218.pdf>
103. <https://arxiv.org/pdf/1903.10873v5.pdf>
104. <https://arxiv.org/pdf/1909.11573.pdf>
105. <https://arxiv.org/ftp/arxiv/papers/1905/1905.00599.pdf>
106. <http://www.cse.cuhk.edu.hk/leojia/projects/weatherclassify/index.htm>
107. <https://sites.google.com/view/yunfuliu/desnownet>



108. [http://ancuti.meo.etc.upt.ro/D\\_Hazzy\\_ICIP2016/](http://ancuti.meo.etc.upt.ro/D_Hazzy_ICIP2016/)
109. [https://www.researchgate.net/publication/331353178\\_Weather\\_Classification\\_using\\_Convolutional\\_Neural\\_Networks](https://www.researchgate.net/publication/331353178_Weather_Classification_using_Convolutional_Neural_Networks)
110. <https://medium.com/@darshita1405/superpixels-and-slic-6b2d8a6e4f08>
111. [https://bair.berkeley.edu/blog/2019/06/07/data\\_aug/](https://bair.berkeley.edu/blog/2019/06/07/data_aug/)
112. [https://www.researchgate.net/publication/326799649\\_Weather\\_Classification\\_A\\_new\\_multiclass\\_dataset\\_data\\_augmentation\\_approach\\_and\\_comprehensive\\_evaluations\\_of\\_Convolutional\\_Neural\\_Networks](https://www.researchgate.net/publication/326799649_Weather_Classification_A_new_multiclass_dataset_data_augmentation_approach_and_comprehensive_evaluations_of_Convolutional_Neural_Networks)
113. <https://arxiv.org/ftp/arxiv/papers/1910/1910.09910.pdf>
114. <https://ieeexplore.ieee.org/document/7059167>
115. <https://www.cs.ccu.edu.tw/~wtchu/projects/Weather/index.html>
116. <https://www.kdnuggets.com/2020/01/5-most-useful-techniques-handle-imbalanced-datasets.html>
117. <http://perso.lcpc.fr/tarel.jean-philippe/bdd/frida.html>
118. [https://live.ece.utexas.edu/research/fog/fade\\_defade.html](https://live.ece.utexas.edu/research/fog/fade_defade.html)
119. <https://sites.google.com/view/reside-dehaze-datasets>
120. <https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/>
121. <https://hackernoon.com/what-steps-should-one-take-while-doing-data-preprocessing-502c993e1caa>
122. <https://medium.com/analytics-vidhya/categorical-variable-encoding-techniques-17e607fe42f9>
123. <https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f>
124. <https://towardsdatascience.com/why-you-should-do-feature-engineering-first-hyperparameter-tuning-second-as-a-data-scientist-334be5eb276c>
125. <https://www.ijrte.org/wp-content/uploads/papers/v8i5/E6360018520.pdf>
126. <http://downloads.hindawi.com/journals/amete/2020/6972826.pdf>

## Listings

3.1 Import Libraries.....	77
3.2 From images to npy file.....	77
3.3 Loading dataset. ....	78
3.4 Dataset pre-processing.....	78
3.5 Model architecture.....	79
3.6 Model training.....	79
3.7 Model evaluation.....	80