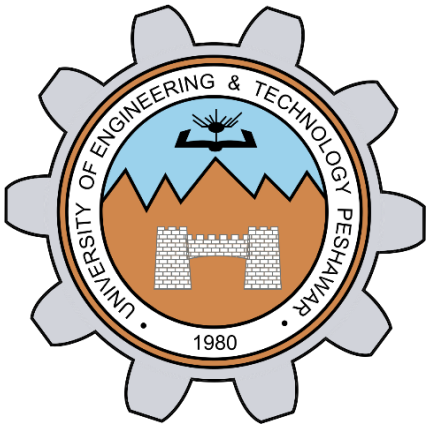
CLASSIFYING WEATHER CONDITIONS USING DEEP LEARNING: A STUDY OF RESNET-50



THESIS
Data Science Mini Project
B.Sc. in Computer Science
Department of Computer Science and Information Technology

Submitted By

Zeeshan Haider (19PWBCS)

Supervisor
DR. Iftikhar Ahmed

University of Engineering and Technology, Peshawar, Pakistan

Table of contents

Description Page No.

1. Abstract:

Weather phenomenon recognition notably affects many aspects of our daily lives, for example, weather forecast, road condition monitoring, transportation, agriculture and the detection of the natural environment. The recognition of weather condition from still images is quite challenging due to weather diversity and lack of distinct characteristics that exists in many weather conditions. Some researchers have used the K-nearest neighbor method to recognize a specific extract of a weather condition, to test the efficiency of the recognition task. In order to enhance the accuracy of recognizing weather conditions, here I am using the approach of convolutional layers of Resnet-50 model to extract the essential features of an image. Thereafter, uses the fully connected layers and the softmax classifier to recognize and classify the images.
Here In this work, we propose a Resnet-50 based approach to classify different types of weather from images. We utilize a dataset of 3604 images, representing different types of weathers. Our Resnet-50 model achieved an accuracy of 79% in classifying weather. This work serves as a proof of concept for using Resnet-50 in weather classification and has potential for further development.

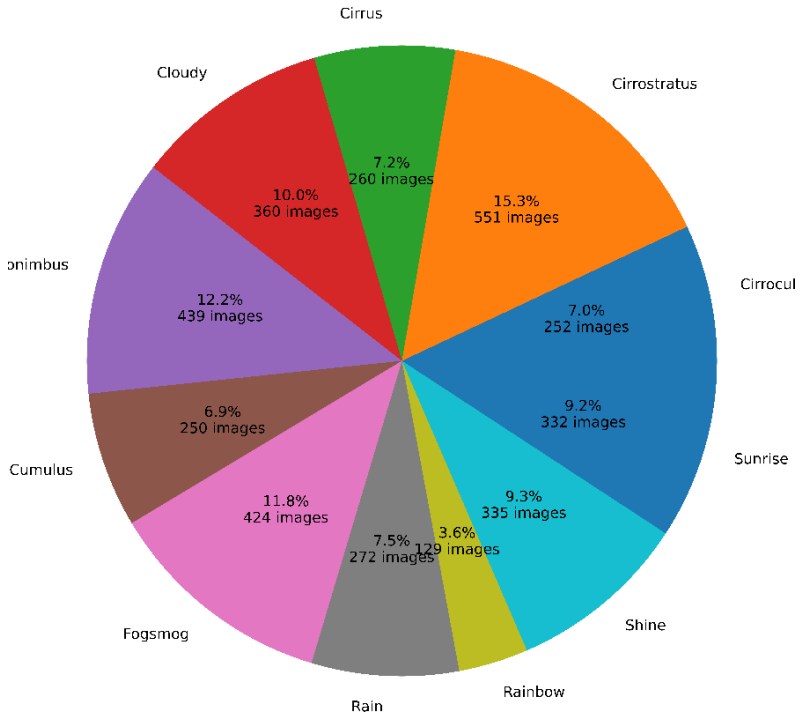
2. Introduction

The weather condition has a great impact on our daily lives, from indoor and outdoor activities such as our dressing, travelling, sporting, to solar technologies and productions. To this point, it has become necessary to keep track on the daily weather condition for a range of our daily activities. Any region's weather is closely tied to the presence of clouds. Clouds play a major role in all types of precipitation. Although not all clouds may result in precipitation, they are crucial for controlling the weather in some regions. Different types and heights of clouds exist in various geographical locations, such as the Earth's tropics or poles [1]. The sky always has clouds, and they are ever-changing. Clouds serve as indicators of atmospheric conditions and are crucial for weather forecasting and warnings, as well as for controlling the Earth's energy balance, temperature, and weather [2]. Before creating a weather prediction, meteorologists investigate the specifics of the cloud type since they are constantly changing.
Therefore, it necessitates for a different method of weather recognition from still images by using computer vision techniques [3]. It is also important to many other applications of computer vision [4] such as image retrieval [5], image restoration also the outdoor surveillance system is reliant on its improvement [6], and vehicle assistant driving systems [7], could also take advantage of the end result of weather recognition.
A support vector machine (SVM), and a decision tree based method used to recognize and categorize weather conditions from an image by pulling features such as contrast, slope, saturation and noises, [8], [9]. This technique used a very small of dataset, the recognition error rate of rainy and foggy and days were 25%, and 15% respectively, but the actual desires could not be met [10]. Though the size of image dataset was not sufficient which resulted to lack of generalization, and robustness.
Deep learning neural networks are gaining more popularity in the areas of image classification like computer vision. Deep Neural Networks (DNNs) are multi-layered feed forward neural networks, especially, deep convolutional neural networks (CNNs) which learns features automatically from raw images. These are made up of simple processing components known as artificial neurons, they are organized in a layered manner. Every layer in the neuron is interconnected through partial connection, to a neuron in the next layer, and the flow of several layers are used for automatic feature extraction, which becomes difficult when there is increase in the number of layers.
As CNN can extract rich, abstract, and deep semantic information from weather images, they are

superior to traditional methods of weather recognition to a large extent. A pretrained AlexNet network was fine-tuned through a two-class weather dataset, and the recognition accuracy reached 82.2%, achieving good results [11, 12]. Then, the features extracted by hand and by CNN were combined to improve the classification performance, and the recognition accuracy can reach 91.4% [12]. A pretrained GoogLeNet network was fine-tuned through a large-scale extreme weather dataset collected by themselves and obtained a more accurate model of weather recognition, with the recognition accuracy up to 94.5% [13]. Then, a new double-fine-tuning strategy was proposed to train the GoogLeNet model and optimize the original GoogLeNet network [14]. The size of the model was only one-third of that of the original model, and the recognition accuracy was improved to 95.46%.

3. Dataset

This dataset consist of 3,604 images belonging to 11 classes, as a pi graph is shown below:



I am collected these images from the following sources.

- GitHub dataset
- Keggle dataset

Here the class's names are listed;

classes = ["Cirroculumulus", "Cirrostratus", "Cirrus", "Cloudy", "Cumulonimbus","Cumulus","Fogsmog","Rain","Rainbow","Shine","Sunrise"]

- **Cirroculumulus**

Cirrocumulus clouds are a type of high-level cloud that typically form at altitudes of 20,000 to 30,000 feet. They are composed of ice crystals and are often seen as small, white, rounded masses arranged in rows or waves. They are usually uniform in shape and size and are often described as "sheep back" clouds due to their appearance. These clouds are usually formed by strong, steady winds that are blowing in the same direction at similar speeds. They are typically associated with fair weather and light winds and are often seen in the sky on clear, sunny days. However, they can also be an indicator of an approaching storm or a change in weather conditions, so it's important to pay attention to other weather indicators as well.

- **Cirrostratus**

Cirrostratus clouds are a type of high-level cloud that typically form at altitudes of around 20,000 feet. They are composed of ice crystals and are often seen as thin, white, sheet-like clouds that cover much of the sky. They can be difficult to distinguish from other types of clouds and are often confused with cirrus clouds. They typically form when the atmosphere is stable, and the temperature and humidity are uniform. They can also be formed by the spreading of a warm front. These clouds are usually associated with fair weather and light winds, but they can also be an indicator of an approaching storm or a change in weather conditions. They can also cause halo effect around the sun or the moon.

- **Cumulonimbus**

Cumulonimbus clouds are a type of thunderstorm cloud that can reach great heights, sometimes up to 60,000 feet or more. They are large, dense, and vertically developed clouds that are composed of water droplets and ice crystals. They are often described as "towering" clouds due to their height and shape, and they can appear in a variety of forms, including anvil-shaped, cauliflower-shaped, and even wall-like. These clouds are usually associated with severe weather, including thunderstorms, heavy rain, strong winds, hail, and even tornadoes. They are formed by strong rising currents of warm, moist air, which can cause the cloud to continue growing and developing until it reaches its maximum height. Due to this, they are often called as "Thunderheads". These clouds should be taken very seriously and people should be aware of the possible hazards associated with them.

4. Requirements:

- Tensorflow
- Keras
- Numpy
- Pandas
- Opencv-python
- Scikit learn
- Matplotlib

Technical terms used

1. Machine Learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data,

Known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuseson making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis throughunsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that mapsinputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it onits own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the programis provided feedback that's analogous to rewards, which it tries to maximize. Other approaches have been developed which don't fit neatly into this three-fold categorization, and sometimes more than one is used by the same machine learning system.

2. Computer vision

Computer vision is an interdisciplinary scientific field that deals with how computerscan gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the humanvisual system can do, Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high dimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions, Understanding in this context means thetransformation of visual images (the input of the retina)into descriptions of the worldthat make sense to thought

processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from imagedata using models constructed with the aid of geometry, physics, statistics, and learning theory.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multi- dimensional data from a 3D scanner or medical scanning device. The technologicaldiscipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. Itinvolves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images.

The image data can take many forms, such as video sequences, views from multiplecameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

3. Deep learning

Deep learning methods aim at learning feature hierarchies with features from higherlevels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, withoutdepending completely on human-crafted features. Deep learning algorithms seek toexploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these conceptsare built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domainswhere the inputs (and even output) are analog. Meaning, they are not a few quantitiesin a tabular format but instead are images of pixel data, documents of text data or files of audio data. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels ofabstraction.

4. Tensor Flow

Tensor Flow 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 usedfor both research and production at Google, Tensor Flow is Google Brain's second- generation system. Version 1.0.0 was released on February 11, while the referenceimplementation runs on single devices, Tensor Flow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units)

Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computingplatforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), andfrom desktops to clusters of servers to mobile and edge devices.

The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories onGitHub mentioned Tensor Flow, of which only 5 were from Google. Unlike other numerical libraries intended for use in Deep Learning like Theano, Tensor Flow wasdesigned for use both in research and development and in production systems, not least Rank Brain in Google search and the fun Deep Dream project. It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

5. Keras

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it providesclear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that canrun on top of Theano or Tensor Flow. It was developed to make implementing deeplearning models as fast and easy as possible for research and development.It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

Keras was developed and maintained by François Chollet, a Google engineer usingfour guiding principles:

- Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined inarbitrary ways.

- Minimalism: The library provides just enough to achieve an outcome, no frills andmaximizing readability.

- Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

- Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or Tensor Flow backend.

6. Convolutional Neural Network:

Over decades, computer vision techniques have been developed and achieved significant performance due to deep learning approaches [12], e.g., convolutional neural networks (CNNs). According to the ImageNet Challenge [13], InceptionV3 [14] is the first runner up for image classification in ImageNet 2015 and was published in CVPR 2016. It is the upgrade version of GoogleNet[15], which reduces computational complexity. The main concept was to factorize convolutions into smaller convolutions. Fig. 1 shows an overview of InceptionV3 architecture. MobileNetV2[16] aims to be a lightweight model. It is based on an inverted residual structure and depth wise separable convolutions to reduce the complexity and model size. NASNet[17] is a current state-of-the-art on ImageNet classification with 82.70% top-1 accuracy. The architecture is based on neural architecture search (NAS) framework. The main concept is to search the best convolutional layer (cell) on the smaller dataset (i.e., CIFAR- 10) and apply this layer to larger data such as ImageNet by stacking copies of this layer.

However CNN,are known to have high computation cost for training an entire model from scratch because they have several convolutional layers and connectwith fully connected layers. It also requires a lot of data to get better accuracy and reduce overfitting. In order to reduce these limitations, there is a method called transfer learning. Transfer learning [17−19] refers to the transfer of existing weights from pre-trained networks on a large dataset such as ImageNet[20] and COCO[21]. The main purpose is to reuse the parameters in the feature extraction layers to produce feature vectors instead of training them. The model has then replaced its full connected layers (classification layers) with the fully connected layersof the new dataset. Therefore, we can reduce computation costs by training the model on new classification layers. In this project, we apply dog breeds as the classification output and dog face images as the input images. We compare three pre-trained networks, including InceptionV3, MobileV2, and NASNet, to see the most suitable net for this particular task of our research question. Thiscould be a good guideline for any other similar tasks.

7. ResNet50

ResNet50 is a specific version of the ResNet (Residual Network) architecture, which is a type of

convolutional neural network (CNN). ResNet50 is a deep learning model trained on more than a million images from the ImageNet dataset. It's a popular choice for image classification tasks and is known for its ability to achieve high accuracy while also being able to handle very deep network architectures. ResNet50 is trained to identify 1000 classes of objects, the architecture of the model is based on "residual connections" which allows the model to learn even deeper representation of the image.

Proposed System

The proposed system focuses on how to identify the weather from different classes by using RESNET50 with the help of computer vision and deep learning algorithm by using the Tensor flow, Keras, numpy, matplotlib.pyplot and pandas library.

1. Python Code

```
# Load the Drive helper and mount
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')

# After executing the cell above, Drive files will be present in "/content/drive/My
Drive".
!ls "/content/drive/My Drive"

# Important imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
import random
from os import listdir
from sklearn.preprocessing import  LabelBinarizer
from keras.preprocessing import image
#from keras.preprocessing.image import img_to_array, array_to_img
from tensorflow.keras.utils import img_to_array, array_to_img
from keras.optimizers import Adam
from PIL import Image
from keras.models import Sequential
#from keras.layers.normalization import BatchNormalization
from tensorflow.keras.layers import BatchNormalization
from keras.layers import Conv2D, MaxPooling2D, Activation, Flatten, Dropout, Dense,
LeakyReLU,AveragePooling2D
from sklearn.model_selection import train_test_split
```

```
# Listing directory
!ls "/content/drive/My Drive/WeatherImages"

# Plotting 25 images to check dataset
plt.figure(figsize=(11,11))
path = "/content/drive/My Drive/WeatherImages/Rainbow"
for i in range(1,26):
    plt.subplot(5,5,i)
    plt.tight_layout()
    rand_img = imread(path +'/'+ random.choice(sorted(listdir(path))))
    plt.imshow(rand_img)
    plt.title('Rainbow')
    plt.xlabel(rand_img.shape[1], fontsize = 10)
    plt.ylabel(rand_img.shape[0], fontsize = 10)



# Check number of images for each type of weather condition
num_cloudy = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Cloudy/'))
num_rain = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Rain/'))
num_shine = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Shine/'))
num_sunrise = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Sunrise/'))
num_fogsmog = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Fogsmog/'))
num_rainbow = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Rainbow/'))
num_Cirroculumulus =
len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Cirroculumulus/'))
num_Cirrostratus =
len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Cirrostratus/'))
num_Cirrus = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Cirrus/'))
num_Cumulonimbus =
len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Cumulonimbus/'))
num_Cumulus = len(os.listdir(r'C:\Users\HP\Desktop\WeatherImages/Cumulus/'))



#classes = ["Cirroculumulus", "Cirrostratus", "Cirrus", "Cloudy",
"Cumulonimbus","Cumulus","Fogsmog","Rain","Rainbow","Shine","Sunrise"]

# Plot distribution of classes
def label_pie(pct, allvals):
    absolute = int(round(pct/100.*np.sum(allvals)))
    return "{:.1f}%\n{:d} images".format(pct, absolute)

fig = plt.figure(figsize=(6,6))
```

```
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
weather_conditions = ["Cirroculumulus", "Cirrostratus", "Cirrus", "Cloudy",
"Cumulonimbus","Cumulus","Fogsmog","Rain","Rainbow","Shine","Sunrise"]
num_images =
[num_cloudy,num_rain,num_shine,num_sunrise,num_fogsmog,num_rainbow,num_Cirroculumul
us,num_Cirrostratus,num_Cirrus,num_Cumulonimbus,num_Cumulus]
ax.pie(num_images, labels = weather_conditions, autopct=lambda pct: label_pie(pct,
num_images), textprops={'fontsize': 8})
plt.title('Composition of original dataset ({} total
images)'.format(sum(num_images)), fontsize=15)
plt.show()
```

```
# Setting root directory path and creating empty list
dir = "/content/drive/My Drive/WeatherImages"
root_dir = listdir(dir)
image_list, label_list = [], []

# Reading and converting image to numpy array
for directory in root_dir:
  for files in listdir(f"{dir}/{directory}"):
    image_path = f"{dir}/{directory}/{files}"
    image = Image.open(image_path)
    image = image.resize((150,150)) # All images does not have same dimension
    image = img_to_array(image)
    image_list.append(image)
    label_list.append(directory)

# Visualize the number of classes count
label_counts = pd.DataFrame(label_list).value_counts()
label_counts

# Checking count of classes
num_classes = len(label_counts)
num_classes

# Checking x data shape
np.array(image_list).shape

# Checking y data shape
label_list = np.array(label_list)
label_list.shape

# Splitting dataset into test and train
```

```
x_train, x_test, y_train, y_test = train_test_split(image_list, label_list,
test_size=0.2, random_state = 10)


# Normalize and reshape data
x_train = np.array(x_train, dtype=np.float16) / 225.0
x_test = np.array(x_test, dtype=np.float16) / 225.0
x_train = x_train.reshape( -1, 150,150,3)
x_test = x_test.reshape( -1, 150,150,3)

# Binarizing labels
lb = LabelBinarizer()
y_train = lb.fit_transform(y_train)
y_test = lb.fit_transform(y_test)
print(lb.classes_)

# Splitting the training data set into training and validation data sets
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size =
0.2)


from tensorflow.keras.applications.resnet50 import ResNet50


# Get the ResNet50 base model
basemodel = ResNet50(weights = 'imagenet', include_top = False, input_tensor =
Input(shape=(150, 150, 3)))


basemodel.summary()

# freeze the model weights

for layer in basemodel.layers:
  layers.trainable = False

headmodel = basemodel.output
headmodel = AveragePooling2D(pool_size = (4,4))(headmodel)
headmodel = Flatten(name= 'flatten')(headmodel)
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)#
headmodel = Dense(256, activation = "relu")(headmodel)
headmodel = Dropout(0.3)(headmodel)
#headmodel = Dense(256, activation = "relu")(headmodel)
#headmodel = Dropout(0.3)(headmodel)
headmodel = Dense(11, activation = 'softmax')(headmodel)
```

```
model = Model(inputs = basemodel.input, outputs = headmodel)

model.summary()

# Compiling model
model.compile(loss = 'categorical_crossentropy', optimizer =
Adam(0.0005),metrics=['accuracy'])

# Training the model
epochs = 25
batch_size = 64
history = model.fit(x_train, y_train, batch_size = batch_size, epochs = epochs,
validation_data = (x_val, y_val))

#Plot the training history
plt.figure(figsize=(12, 5))
plt.plot(history.history['accuracy'], color='r')
plt.plot(history.history['val_accuracy'], color='b')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'val'])
plt.show()

#Plot the loss history
plt.figure(figsize=(12, 5))
plt.plot(history.history['loss'], color='r')
plt.plot(history.history['val_loss'], color='b')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'val'])
plt.show()

# Calculating test accuracy
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")

# Storing model predictions
y_pred = model.predict(x_test)

# Plotting image to compare
img = array_to_img(x_test[6])
img
```

```
# Finding max value from predition list and comaparing original value vs predicted
labels = lb.classes_
print(labels)
print("Originally : ",labels[np.argmax(y_test[6])])
print("Predicted : ",labels[np.argmax(y_pred[6])])

# Saving model
model.save("/content/drive/My Drive/intel_image.h5")
```

Results

The results of our project are listed below;

- **The desired output on 25 epochs**

```
# Plotting image to compare
img = array_to_img(x_test[6])
img
```

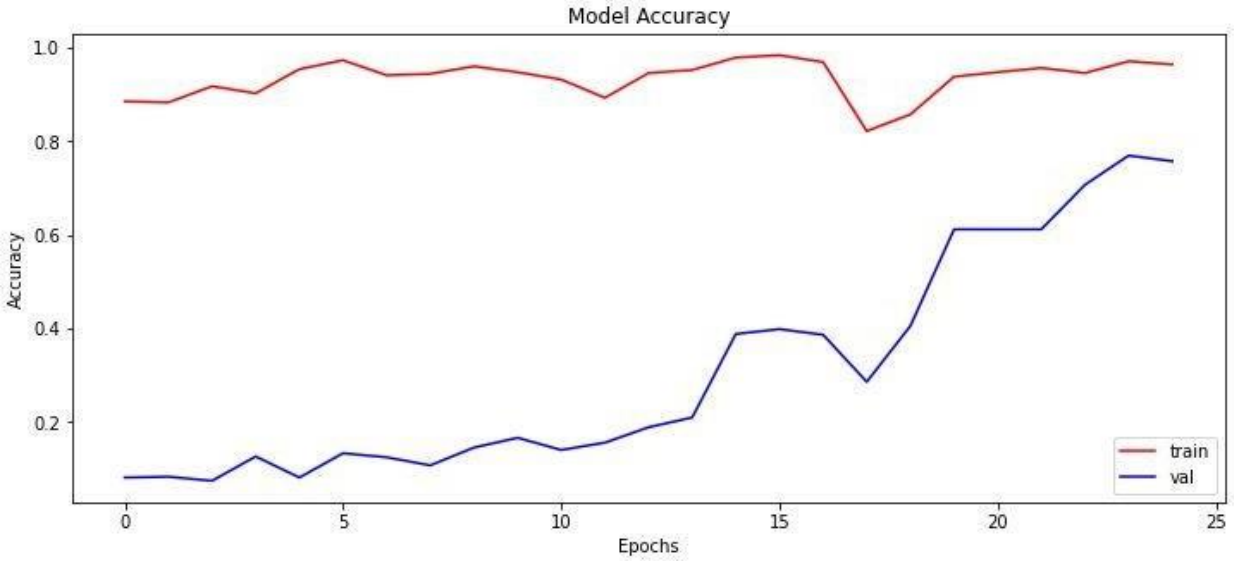
```
# Finding max value from predition list and comaparing original value vs predicted
labels = lb.classes_
print(labels)
print("Originally : ",labels[np.argmax(y_test[6])])
print("Predicted : ",labels[np.argmax(y_pred[6])])
```

```
['Cirroculumulus' 'Cirrostratus' 'Cirrus' 'Cloudy' 'Cumulonimbus'
 'Cumulus' 'Fogsmog' 'Rain' 'Rainbow' 'Shine' 'Sunrise']
Originally :  Rain
Predicted :  Rain
```

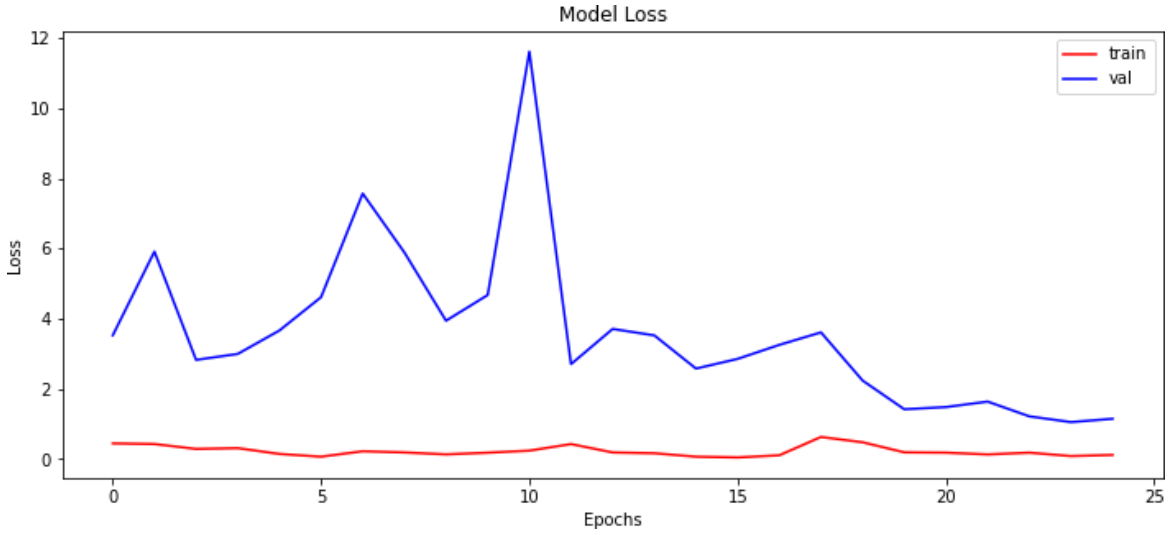
- **Accuracy on 25 epochs is 79.3%**

```
[42] # Calculating test accuracy
     scores = model.evaluate(x_test, y_test)
     print(f"Test Accuracy: {scores[1]*100}")

     23/23 [==============================] - 1s 57ms/step
     Test Accuracy: 79.33425903320312
```

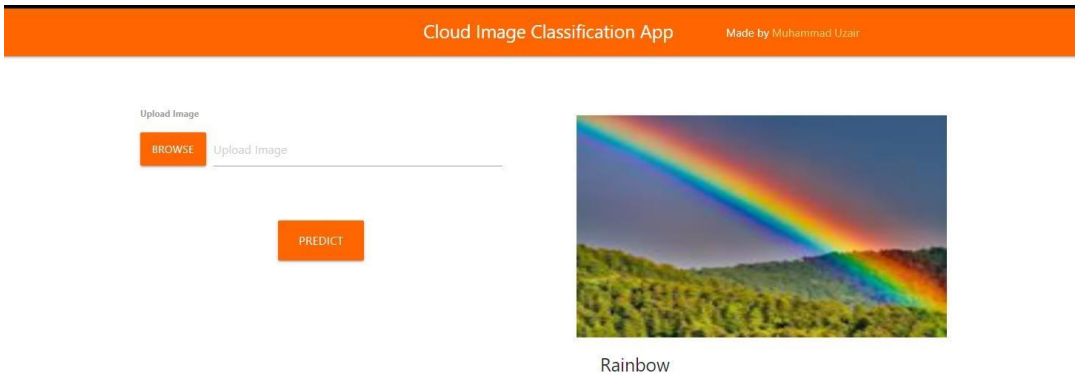


- **Model Loss**



Flask App

- Now I save this model .h5 file and run it through that I have made for it, as result as shown



Conclusion

At the end we concluded that we can achieve more accuracy by increasing no of epochs .Because here I am only taking 50 epochs due to computational power of my PC and getting accuracy 79%. We can also increase the number of classes as well. From the above results you can see that by using this model we can easily predict the class of weather in a second which is a time consuming task in case of human.

References

[1] Satapathy, J.; Thapliyal, P. Retrieval of cloud-cleared radiances using numerical weather prediction model-analysis and forecast fields for INSAT-3D sounder longwave window channel observations. J. Atmos. Sol. Terr. Phys. **2021**, 217, 105602. [CrossRef]

[2] Lin, F.; Zhang, Y.; Wang, J. Recent advances in intra-hour solar forecasting: A review of ground-based sky image methods. Int. J.Forecast. **2022**, in press. [CrossRef]

[3] Elhoseiny M, Huang S, and Elgammal A. 2015. Weather classification with deep convolutional neural . networks. In Image Processing (ICIP) IEEE International Conference pp. 3349–3353.

[4] Chen H, Ding G, Zhao S, Han. J. 2018. Temporal-difference learning with sampling baseline for image captioning, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence.

[5] Qayyum A, Anwar S.M, Awais M. Majid. 2017. Medical image retrieval using deep convolutional neural network, Neurocomputing 266 8–20.

[6] Elhoseiny M, Huang S, Elgammal. A. 2015. Weather classification with deep convolutional neural networks, in: Image Processing (ICIP), IEEE International Conference on, pp. 3349–3353.

[7] Kurihata H, Takahashi T, Mekada Y, Muras .H, Tamatsu T 2005. Miyahara, Rainy weather recognition from in-vehicle camera images for driver assistance, in: IEEE Proceedings. Intelligent Vehicles Symposium, pp. 205–210.

[8] Li Q, Kong Y, Xia S.M .2014. A method of weather recognition based on outdoor images. In Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP), IEEE, Lisbon, Portugal, pp. 510–516.

[9] Jindal A, Dua A, Kaur K, Singh M, Kumar N, Mishra. S. 2016. Decision tree and SVM-based data analytics for the detection in smart grid, IEEE Transactions on Industrial Informatics, vol. 12, no. 3, pp. 1005–1016.

[10] Narasimhan S.G, Wang C, Nayar S.K. 2002. All the images of an outdoor scene, In Computer Vision-ECCV, pp. 148–162, Springer, Berlin, Germany.

[11] M. Elhoseiny, S. Huang, and A. Elgammal, "Weather classification with deep convolutional neural networks," in *Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP)*, pp. 3349–3353, IEEE, Quebec City, QC, Canada, September 2015.

View at: Publisher Site | Google Scholar

[12] C. Lu, D. Lin, J. Jia, and C.-K. Tang, "Two-class weather classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2510–2524, 2017.

View at: Publisher Site | Google Scholar

[13] Z. Zhu, L. Zhuo, P. Qu, K. Zhou, and J. Zhang, "Extreme weather recognition using convolutional neural networks," in *Proceedings of the 2016 IEEE International Symposium on Multimedia (ISM)*, pp. 621–625, IEEE, San Jose, CA, USA, December 2016.

View at: Publisher Site | Google Scholar

[14] Z. Zhu, J. Li, L. Zhuo, and J. Zhang, "Extreme weather recognition using a novel fine-tuning strategy and optimized GoogLeNet," in *Proceedings of the 2017 International Conference on Digital*

Image Computing: Techniques and Applications (DICTA), pp. 1–7, IEEE, Sydney, Australia, November 2017.
View at: Publisher Site | Google Scholar