Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies 2 in CpE
2nd Semester	AY 2023-2024
ACTIVITY NO.:	**Final Examination**
Names:	Garcia, Christian Andrei V.
Section:	CPE32S1
Date Performed:	5/14/24
Date Submitted:	5/19/24
Instructor:	Engr. Ryan Francisco

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install tensorflow keras scikit-learn
```

```
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: keras in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.63.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.43.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.6)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.0.3)
```

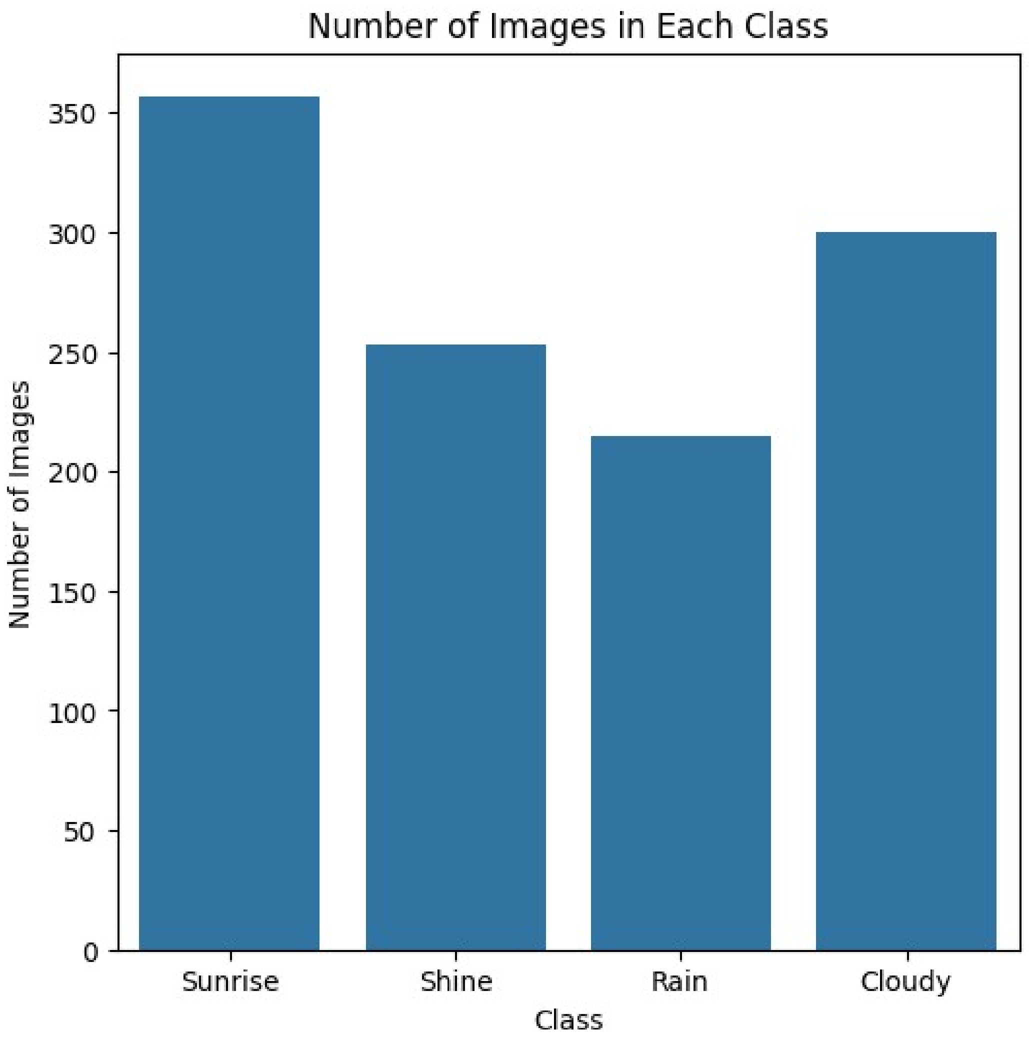
```
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (2024.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.16,>=2.15->tensorflow) (2.1.5)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.6.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5-
>tensorboard<2.16,>=2.15->tensorflow) (3.2.2)

import tensorflow as tf
from tensorflow import keras
import tensorflow_hub as hub
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import matplotlib.image as img
import PIL.Image as Image
import cv2
import os
import numpy as np
import pathlib
import glob
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data_dir = "/content/drive/MyDrive/Dataset/Multi-class Weather
Dataset"

Sunrise = glob.glob(data_dir + '/Sunrise/*')[:357]
Shine = glob.glob(data_dir + '/Shine/*')[:253]
Rain = glob.glob(data_dir + '/Rain/*')[:215]
Cloudy = glob.glob(data_dir + '/Cloudy/*')[:300]

allImages = [len(Sunrise) + len(Shine) + len(Rain) + len(Cloudy)]
fig, ax = plt.subplots(ncols = 4, figsize = (20, 4))
fig.suptitle('Weather')
plt.setp(ax, xticks=[], yticks=[])

Sunrise_image = img.imread(Sunrise[0])
Shine_image = img.imread(Shine[0])
Rain_image = img.imread(Rain[0])
Cloudy_image = img.imread(Cloudy[0])

ax[0].set_title('Sunrise')
ax[1].set_title('Shine')
ax[2].set_title('Rain')
ax[3].set_title('Cloudy')

ax[0].imshow(Sunrise_image)
ax[1].imshow(Shine_image)
ax[2].imshow(Rain_image)
ax[3].imshow(Cloudy_image)

plt.show()
print(f"\nFound {allImages} images belonging to 4 classes.\n")
print(f"Sunrise class contains {len(Sunrise)} images.")
print(f"Shine class contains {len(Shine)} images.")
print(f"Rain class contains {len(Rain)} images.")
print(f"Cloudy class contains {len(Cloudy)} images.")
```

```

Found [1125] images belonging to 4 classes.

Sunrise class contains 357 images.
Shine class contains 253 images.
```

```
Rain class contains 215 images.
Cloudy class contains 300 images.
```

```
import matplotlib.pyplot as plt
import seaborn as sns

classes = ['Sunrise', 'Shine', 'Rain', 'Cloudy']
allImages1 = [len(Sunrise), len(Shine), len(Rain), len(Cloudy)]

plt.figure(figsize=(6, 6))
sns.barplot(x = classes, y = allImages1)
plt.title('Number of Images in Each Class')
plt.xlabel('Class')
plt.ylabel('Number of Images')
plt.show()
```



```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
import os

data_dir = "/content/drive/MyDrive/Dataset/Multi-class Weather
Dataset"
img_height, img_width = 244, 244
batch_size = 32

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_gen = datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

valid_gen = datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 901 images belonging to 4 classes.
Found 224 images belonging to 4 classes.
```

```
model = Sequential([
    Flatten(input_shape=(img_height, img_width, 3)),
    Dense(512, activation='relu'),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(4, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 flatten (Flatten)           (None, 178608)            0

 dense (Dense)               (None, 512)               91447808

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 4)                 516

=================================================================
Total params: 91612548 (349.47 MB)
Trainable params: 91612548 (349.47 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________

history = model.fit(
    train_gen,
    steps_per_epoch=train_gen.samples // batch_size,
    validation_data = valid_gen,
    validation_steps = valid_gen.samples // batch_size,
    epochs = 50
)

Epoch 1/50
28/28 [==============================] - 228s 8s/step - loss: 36.2273
- accuracy: 0.4649 - val_loss: 11.2370 - val_accuracy: 0.6161
Epoch 2/50
28/28 [==============================] - 70s 2s/step - loss: 5.9126 -
accuracy: 0.6720 - val_loss: 4.0375 - val_accuracy: 0.7634
Epoch 3/50
28/28 [==============================] - 62s 2s/step - loss: 3.9105 -
accuracy: 0.7123 - val_loss: 7.8742 - val_accuracy: 0.5714
Epoch 4/50
28/28 [==============================] - 66s 2s/step - loss: 3.7844 -
accuracy: 0.7123 - val_loss: 4.2296 - val_accuracy: 0.6429
Epoch 5/50
28/28 [==============================] - 66s 2s/step - loss: 2.6511 -
accuracy: 0.7342 - val_loss: 2.3871 - val_accuracy: 0.7723
Epoch 6/50
28/28 [==============================] - 82s 3s/step - loss: 1.4761 -
accuracy: 0.7814 - val_loss: 4.9575 - val_accuracy: 0.6071
Epoch 7/50
28/28 [==============================] - 65s 2s/step - loss: 1.3261 -
accuracy: 0.8044 - val_loss: 1.5800 - val_accuracy: 0.6786
Epoch 8/50
28/28 [==============================] - 61s 2s/step - loss: 1.3647 -
accuracy: 0.7526 - val_loss: 3.1739 - val_accuracy: 0.7232
Epoch 9/50
```

```
28/28 [==============================] - 65s 2s/step - loss: 0.9300 -
accuracy: 0.8090 - val_loss: 2.6557 - val_accuracy: 0.6161
Epoch 10/50
28/28 [==============================] - 62s 2s/step - loss: 1.2845 -
accuracy: 0.7779 - val_loss: 2.0521 - val_accuracy: 0.6429
Epoch 11/50
28/28 [==============================] - 66s 2s/step - loss: 0.9193 -
accuracy: 0.7699 - val_loss: 1.9720 - val_accuracy: 0.7098
Epoch 12/50
28/28 [==============================] - 65s 2s/step - loss: 0.7711 -
accuracy: 0.7929 - val_loss: 1.2410 - val_accuracy: 0.7634
Epoch 13/50
28/28 [==============================] - 75s 3s/step - loss: 0.4684 -
accuracy: 0.8631 - val_loss: 1.1449 - val_accuracy: 0.7188
Epoch 14/50
28/28 [==============================] - 63s 2s/step - loss: 1.2450 -
accuracy: 0.7434 - val_loss: 1.7009 - val_accuracy: 0.6518
Epoch 15/50
28/28 [==============================] - 63s 2s/step - loss: 0.7750 -
accuracy: 0.8032 - val_loss: 1.2283 - val_accuracy: 0.7679
Epoch 16/50
28/28 [==============================] - 65s 2s/step - loss: 0.5723 -
accuracy: 0.8285 - val_loss: 1.2786 - val_accuracy: 0.7634
Epoch 17/50
28/28 [==============================] - 62s 2s/step - loss: 0.4656 -
accuracy: 0.8458 - val_loss: 0.9762 - val_accuracy: 0.7277
Epoch 18/50
28/28 [==============================] - 65s 2s/step - loss: 0.5297 -
accuracy: 0.8147 - val_loss: 1.1812 - val_accuracy: 0.7723
Epoch 19/50
28/28 [==============================] - 62s 2s/step - loss: 1.4203 -
accuracy: 0.7135 - val_loss: 1.0396 - val_accuracy: 0.7500
Epoch 20/50
28/28 [==============================] - 65s 2s/step - loss: 0.5787 -
accuracy: 0.8270 - val_loss: 0.9536 - val_accuracy: 0.7679
Epoch 21/50
28/28 [==============================] - 74s 3s/step - loss: 0.4152 -
accuracy: 0.8527 - val_loss: 1.4354 - val_accuracy: 0.6830
Epoch 22/50
28/28 [==============================] - 65s 2s/step - loss: 0.4181 -
accuracy: 0.8471 - val_loss: 1.0288 - val_accuracy: 0.7679
Epoch 23/50
28/28 [==============================] - 63s 2s/step - loss: 0.4850 -
accuracy: 0.8354 - val_loss: 0.7031 - val_accuracy: 0.7946
Epoch 24/50
28/28 [==============================] - 64s 2s/step - loss: 0.3093 -
accuracy: 0.8838 - val_loss: 0.8393 - val_accuracy: 0.7634
Epoch 25/50
28/28 [==============================] - 63s 2s/step - loss: 0.3060 -
```

```
accuracy: 0.8815 - val_loss: 0.7692 - val_accuracy: 0.8036
Epoch 26/50
28/28 [==============================] - 65s 2s/step - loss: 0.2330 -
accuracy: 0.9125 - val_loss: 0.8831 - val_accuracy: 0.7545
Epoch 27/50
28/28 [==============================] - 62s 2s/step - loss: 0.2408 -
accuracy: 0.9033 - val_loss: 0.9284 - val_accuracy: 0.7991
Epoch 28/50
28/28 [==============================] - 64s 2s/step - loss: 0.3424 -
accuracy: 0.8803 - val_loss: 1.7043 - val_accuracy: 0.6384
Epoch 29/50
28/28 [==============================] - 76s 3s/step - loss: 0.4387 -
accuracy: 0.8562 - val_loss: 0.8588 - val_accuracy: 0.7768
Epoch 30/50
28/28 [==============================] - 65s 2s/step - loss: 0.2877 -
accuracy: 0.8918 - val_loss: 0.7939 - val_accuracy: 0.7411
Epoch 31/50
28/28 [==============================] - 63s 2s/step - loss: 0.2711 -
accuracy: 0.8964 - val_loss: 0.6280 - val_accuracy: 0.8080
Epoch 32/50
28/28 [==============================] - 66s 2s/step - loss: 0.5576 -
accuracy: 0.8262 - val_loss: 0.7431 - val_accuracy: 0.8036
Epoch 33/50
28/28 [==============================] - 62s 2s/step - loss: 0.3893 -
accuracy: 0.8539 - val_loss: 0.6342 - val_accuracy: 0.8170
Epoch 34/50
28/28 [==============================] - 65s 2s/step - loss: 0.3579 -
accuracy: 0.8700 - val_loss: 0.9930 - val_accuracy: 0.7321
Epoch 35/50
28/28 [==============================] - 63s 2s/step - loss: 0.2921 -
accuracy: 0.8987 - val_loss: 1.1023 - val_accuracy: 0.7009
Epoch 36/50
28/28 [==============================] - 64s 2s/step - loss: 0.3161 -
accuracy: 0.8746 - val_loss: 0.7694 - val_accuracy: 0.8170
Epoch 37/50
28/28 [==============================] - 63s 2s/step - loss: 0.1808 -
accuracy: 0.9298 - val_loss: 0.7127 - val_accuracy: 0.7902
Epoch 38/50
28/28 [==============================] - 77s 3s/step - loss: 0.2150 -
accuracy: 0.9194 - val_loss: 0.7483 - val_accuracy: 0.7768
Epoch 39/50
28/28 [==============================] - 62s 2s/step - loss: 0.1951 -
accuracy: 0.9287 - val_loss: 1.2869 - val_accuracy: 0.7634
Epoch 40/50
28/28 [==============================] - 64s 2s/step - loss: 0.2983 -
accuracy: 0.9022 - val_loss: 0.7998 - val_accuracy: 0.7902
Epoch 41/50
28/28 [==============================] - 63s 2s/step - loss: 0.5041 -
accuracy: 0.8377 - val_loss: 1.4546 - val_accuracy: 0.6562
```

```
Epoch 42/50
28/28 [==============================] - 65s 2s/step - loss: 0.5083 -
accuracy: 0.8446 - val_loss: 1.1327 - val_accuracy: 0.6652
Epoch 43/50
28/28 [==============================] - 62s 2s/step - loss: 0.4862 -
accuracy: 0.8193 - val_loss: 0.9852 - val_accuracy: 0.7232
Epoch 44/50
28/28 [==============================] - 78s 3s/step - loss: 0.3077 -
accuracy: 0.8918 - val_loss: 0.6682 - val_accuracy: 0.8080
Epoch 45/50
28/28 [==============================] - 63s 2s/step - loss: 0.2597 -
accuracy: 0.9045 - val_loss: 1.1408 - val_accuracy: 0.7054
Epoch 46/50
28/28 [==============================] - 75s 3s/step - loss: 0.4801 -
accuracy: 0.8389 - val_loss: 1.5752 - val_accuracy: 0.7054
Epoch 47/50
28/28 [==============================] - 63s 2s/step - loss: 0.3276 -
accuracy: 0.8918 - val_loss: 0.9007 - val_accuracy: 0.7634
Epoch 48/50
28/28 [==============================] - 65s 2s/step - loss: 0.2267 -
accuracy: 0.9241 - val_loss: 0.6119 - val_accuracy: 0.8036
Epoch 49/50
28/28 [==============================] - 62s 2s/step - loss: 0.3506 -
accuracy: 0.8769 - val_loss: 0.7048 - val_accuracy: 0.7768
Epoch 50/50
28/28 [==============================] - 64s 2s/step - loss: 0.2774 -
accuracy: 0.8907 - val_loss: 0.7596 - val_accuracy: 0.7991
```

```
loss, accuracy = model.evaluate(valid_gen)
print(f'Loss: {loss:.2f}')
print(f'Accuracy: {accuracy:.2f}')
```

```
7/7 [==============================] - 4s 518ms/step - loss: 0.7596 -
accuracy: 0.7991
Loss: 0.76
Accuracy: 0.80
```

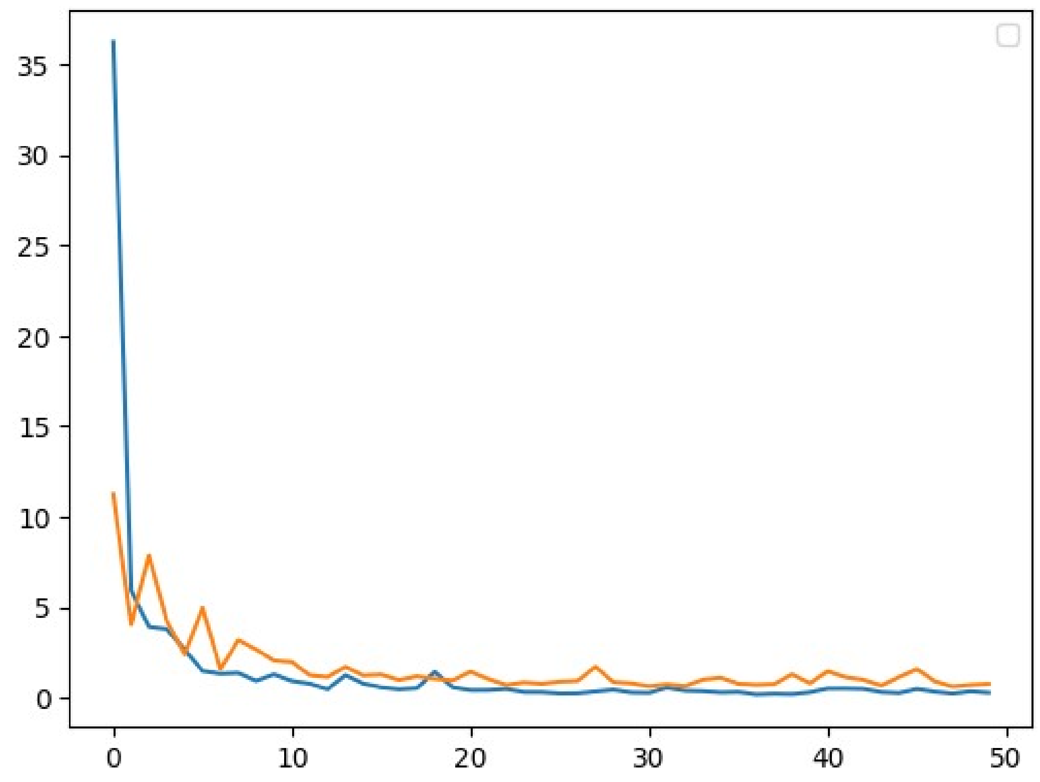
```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

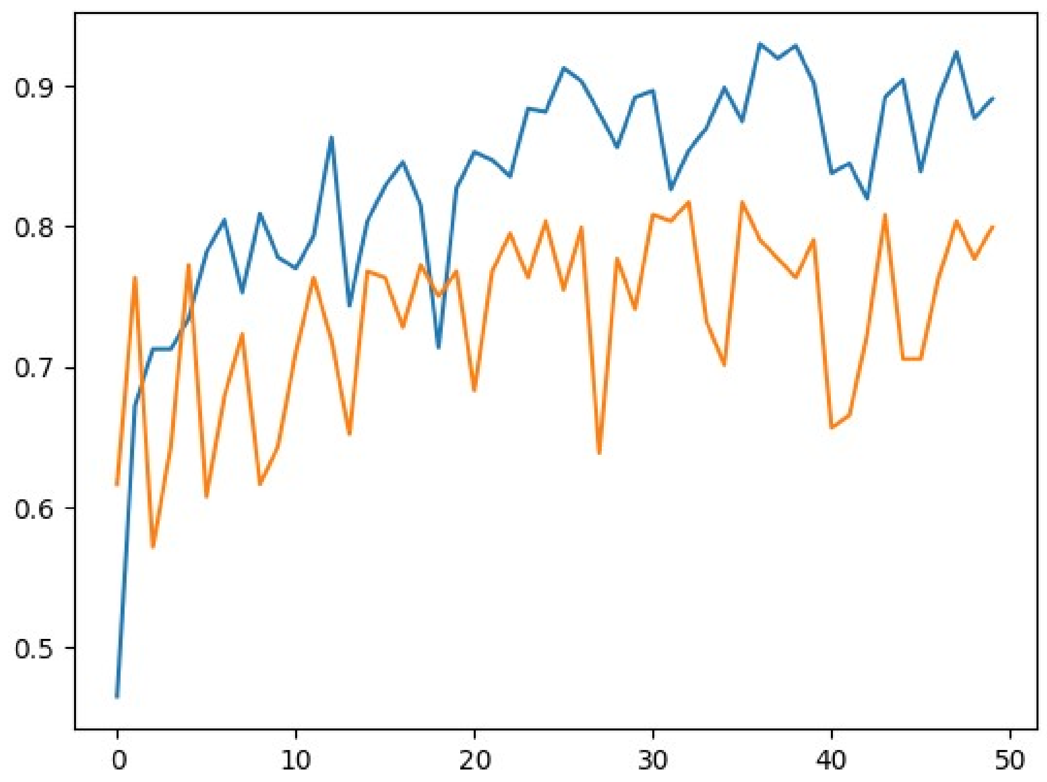
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```

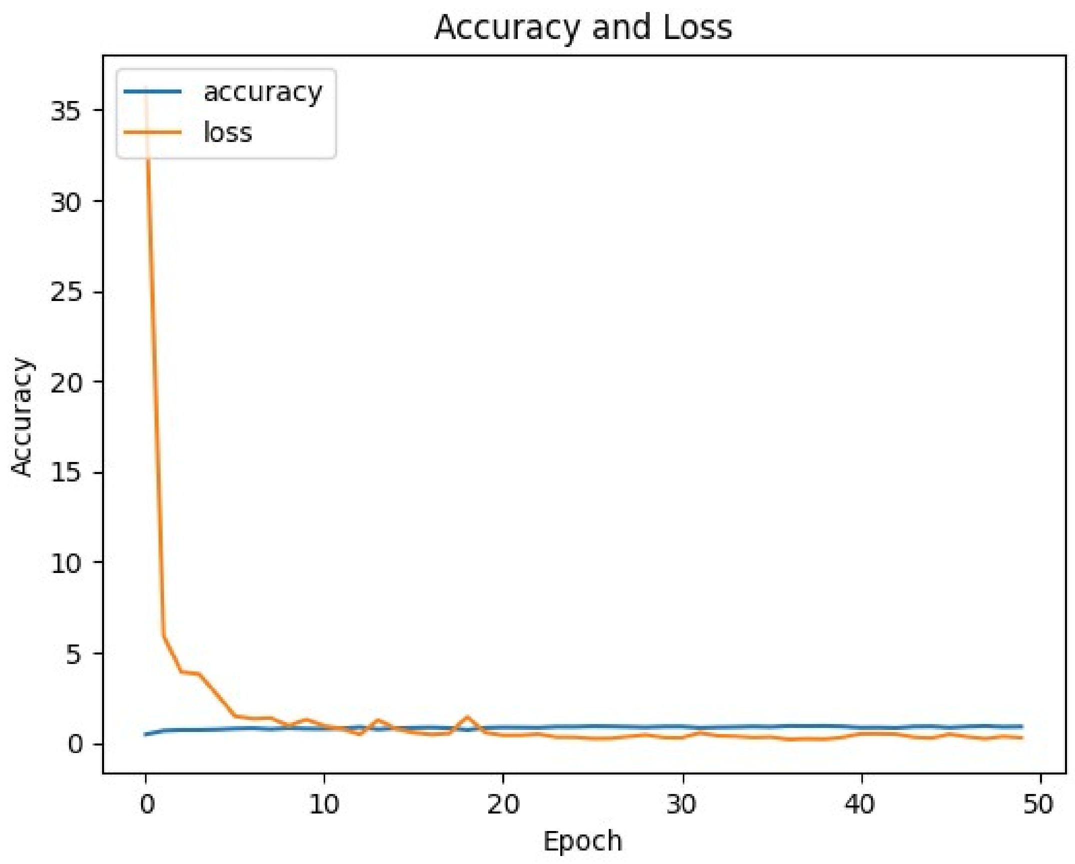
```
<matplotlib.legend.Legend at 0x7a1925ea6f20>
```



```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

[<matplotlib.lines.Line2D at 0x7a1912f04700>]
```



```python
import matplotlib.pyplot as plt

def plot_history(model):
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['loss'])
    plt.title('Accuracy and Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['accuracy', 'loss'], loc='upper left')
    plt.show()

plot_history(model)
```



Save the best model

```
from tensorflow.keras.models import Sequential, model_from_json
from tensorflow.keras.layers import Dense
import numpy
import os

model_weather = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_weather)

model.save_weights("/content/drive/MyDrive/Colab
Notebooks/model_weather.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

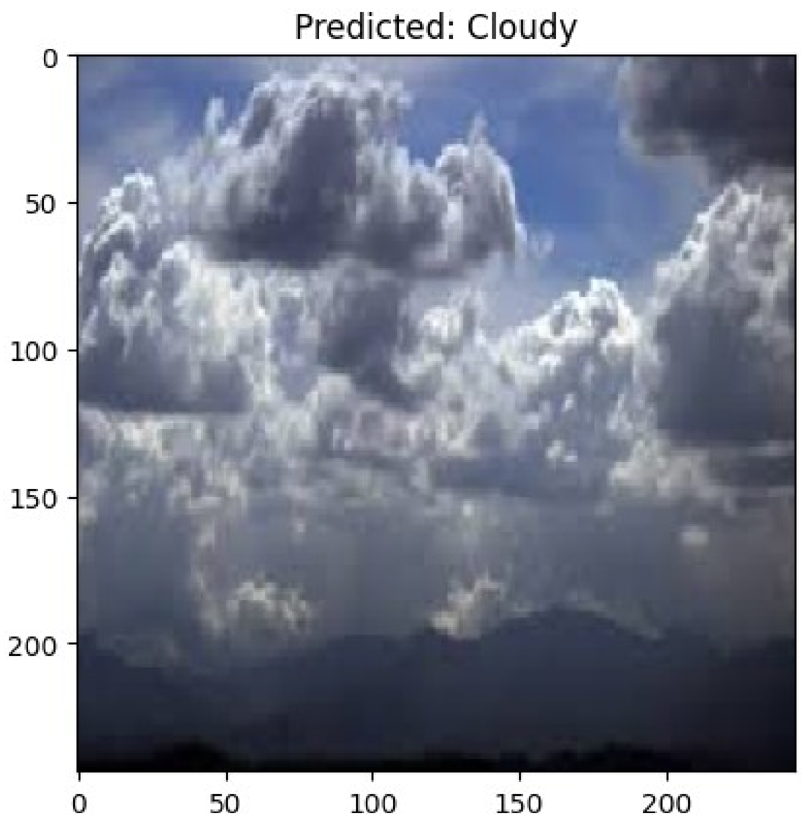
```
pip install h5py
```

```
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-
packages (3.9.0)
```

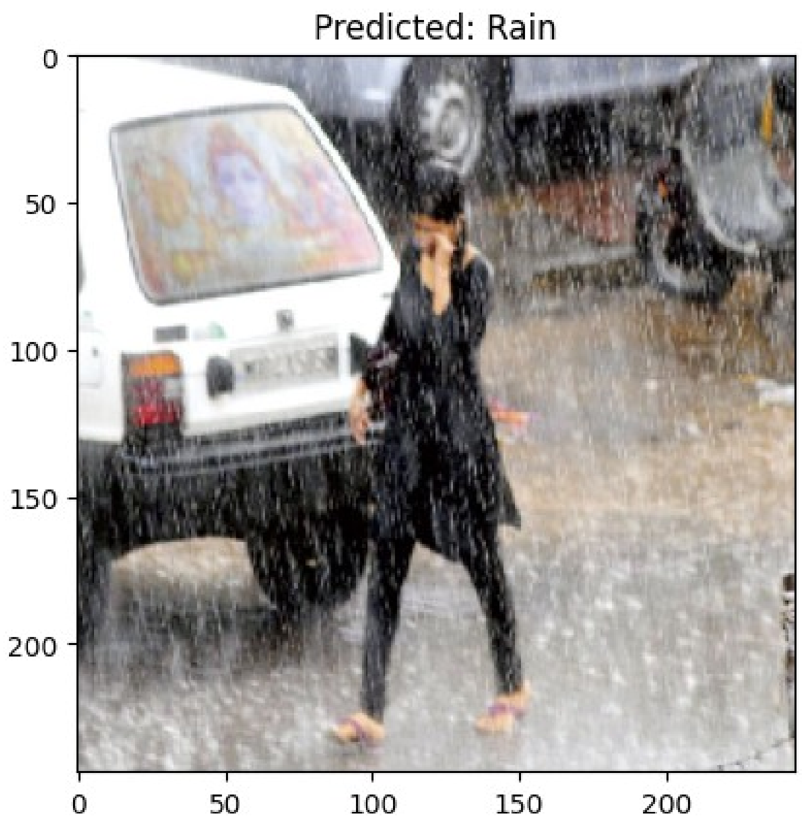
```
Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from h5py) (1.25.2)

from tensorflow.keras.models import load_model

model.save("/content/drive/MyDrive/Colab
Notebooks/model_weather.hdf5")
```

Test a sample image using the saved best model

```
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

def img_process(img_path):
    img = image.load_img(img_path, target_size=(img_height,
img_width))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array

img_path = '/content/drive/MyDrive/Dataset/sample_data/cloudy165.jpg'
img_array = img_process(img_path)

predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=1)

class_labels = list(train_gen.class_indices.keys())
predicted_label = class_labels[predicted_class[0]]
print(f'Predicted label: {predicted_label}')

plt.imshow(image.load_img(img_path, target_size=(img_height,
img_width)))
plt.title(f'Predicted: {predicted_label}')
plt.show()

1/1 [==============================] - 0s 62ms/step
Predicted label: Cloudy
```



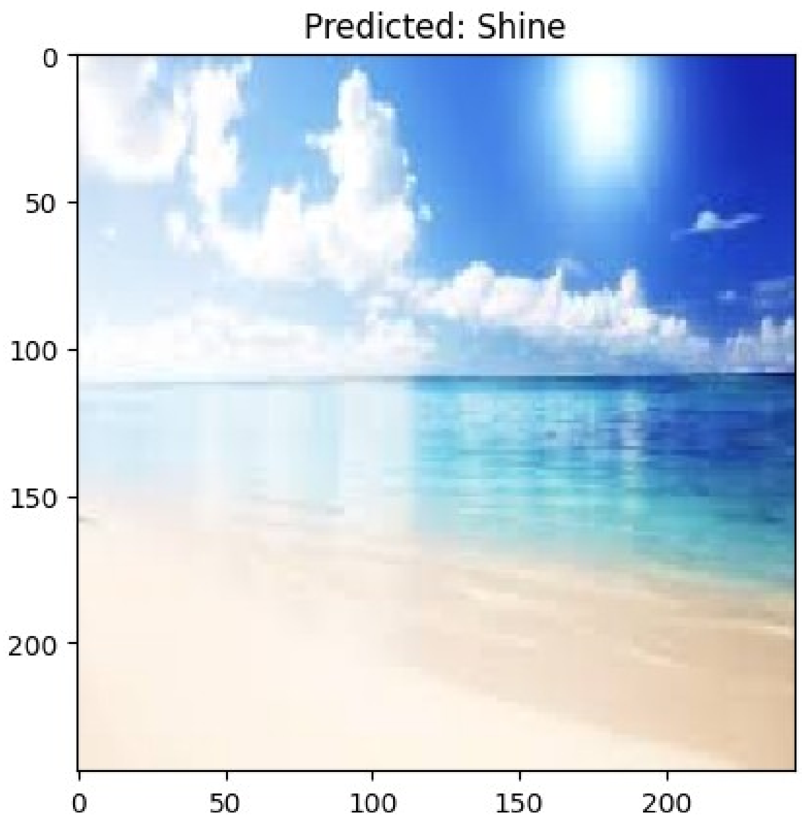
```python
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

def img_process(img_path):
    img = image.load_img(img_path, target_size=(img_height,
img_width))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array

img_path = '/content/drive/MyDrive/Dataset/Multi-class Weather
Dataset/Rain/rain104.jpg'
img_array = img_process(img_path)

predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=1)

class_labels = list(train_gen.class_indices.keys())
predicted_label = class_labels[predicted_class[0]]

print(f'Predicted label: {predicted_label}')

plt.imshow(image.load_img(img_path, target_size=(img_height,
img_width)))
plt.title(f'Predicted: {predicted_label}')
plt.show()

1/1 [==============================] - 0s 58ms/step
Predicted label: Rain
```



```
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image


def img_process(img_path):
    img = image.load_img(img_path, target_size=(img_height,
img_width))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array

img_path = '/content/drive/MyDrive/Dataset/Multi-class Weather
Dataset/Shine/shine104.jpg'
img_array = img_process(img_path)

predictions = model.predict(img_array)
```

```
predicted_class = np.argmax(predictions, axis=1)

class_labels = list(train_gen.class_indices.keys())
predicted_label = class_labels[predicted_class[0]]
print(f'Predicted label: {predicted_label}')

plt.imshow(image.load_img(img_path, target_size=(img_height,
img_width)))
plt.title(f'Predicted: {predicted_label}')
plt.show()
```

```
1/1 [==============================] - 0s 210ms/step
Predicted label: Shine
```



```
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image


def img_process(img_path):
    img = image.load_img(img_path, target_size=(img_height,
img_width))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0
    return img_array

img_path = '/content/drive/MyDrive/Dataset/Multi-class Weather
Dataset/Sunrise/sunrise104.jpg'
img_array = img_process(img_path)

predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=1)

class_labels = list(train_gen.class_indices.keys())
predicted_label = class_labels[predicted_class[0]]
print(f'Predicted label: {predicted_label}')

plt.imshow(image.load_img(img_path, target_size=(img_height,
img_width)))
plt.title(f'Predicted: {predicted_label}')
plt.show()
```

```
1/1 [==============================] - 0s 59ms/step
Predicted label: Sunrise
```