```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE = 244
BATCH_SIZE = 32

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

train_datagen =
ImageDataGenerator(rescale=1./255,validation_split=0.2)
train_generator = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/1sv21cs040/archive (2)/Multi-class Weather
Dataset',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)
val_generator = train_datagen.flow_from_directory(
    '/content/drive/MyDrive/1sv21cs040/archive (2)/Multi-class Weather
Dataset',
    target_size=(IMG_SIZE,IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 915 images belonging to 4 classes.
Found 228 images belonging to 4 classes.
```

```python
# Get the class indices from the training generator
class_indices = train_generator.class_indices

# Extract class names
class_names = list(class_indices.keys())

print("Class indices:", class_indices)
print("Class names:", class_names)
```

```
Class indices: {'Cloudy': 0, 'Rain': 1, 'Shine': 2, 'Sunrise': 3}
Class names: ['Cloudy', 'Rain', 'Shine', 'Sunrise']
```

```python
# Define a Sequential model
model = keras.Sequential([
    layers.Conv2D(32, (3,3), activation='relu',
input_shape=(IMG_SIZE,IMG_SIZE,3)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(4, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

model.fit(train_generator,validation_data=val_generator,epochs=10)

Epoch 1/10
29/29 [==============================] - 410s 14s/step - loss: 0.3617
- accuracy: 0.6798 - val_loss: 0.4575 - val_accuracy: 0.7149
Epoch 2/10
29/29 [==============================] - 124s 4s/step - loss: 0.1749 -
accuracy: 0.8699 - val_loss: 0.2026 - val_accuracy: 0.8684
Epoch 3/10
29/29 [==============================] - 125s 4s/step - loss: 0.1248 -
accuracy: 0.9082 - val_loss: 0.2748 - val_accuracy: 0.7851
Epoch 4/10
29/29 [==============================] - 128s 4s/step - loss: 0.1002 -
accuracy: 0.9333 - val_loss: 0.1553 - val_accuracy: 0.8509
Epoch 5/10
29/29 [==============================] - 132s 5s/step - loss: 0.0925 -
accuracy: 0.9322 - val_loss: 0.1801 - val_accuracy: 0.8465
Epoch 6/10
29/29 [==============================] - 125s 4s/step - loss: 0.0680 -
accuracy: 0.9464 - val_loss: 0.1701 - val_accuracy: 0.8553
Epoch 7/10
29/29 [==============================] - 124s 4s/step - loss: 0.0910 -
accuracy: 0.9322 - val_loss: 0.2554 - val_accuracy: 0.7763
Epoch 8/10
29/29 [==============================] - 127s 4s/step - loss: 0.0629 -
accuracy: 0.9596 - val_loss: 0.2562 - val_accuracy: 0.8377
Epoch 9/10
29/29 [==============================] - 131s 5s/step - loss: 0.0263 -
accuracy: 0.9858 - val_loss: 0.2303 - val_accuracy: 0.8421
Epoch 10/10
29/29 [==============================] - 125s 4s/step - loss: 0.0222 -
accuracy: 0.9880 - val_loss: 0.2417 - val_accuracy: 0.8465

<keras.src.callbacks.History at 0x7d997693c730>

model.save('Alzheimer.h5')

/usr/local/lib/python3.10/dist-packages/keras/src/engine/
training.py:3103: UserWarning: You are saving your model as an HDF5
```

```
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
model = load_model('Alzheimer.h5')
print("Model Loaded")
```

```
Model Loaded
```

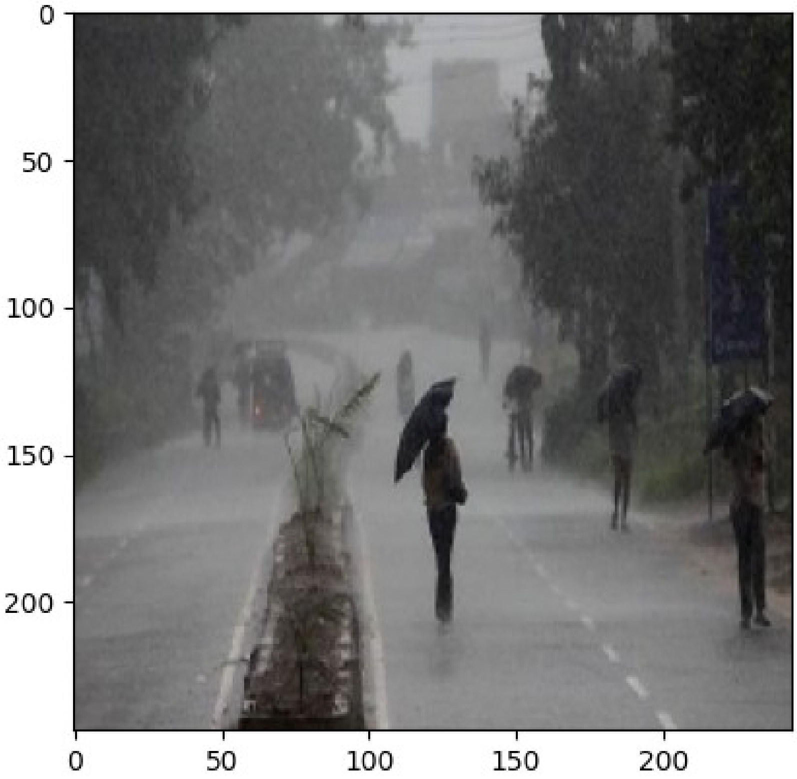
```
# Load and view the image
from matplotlib import pyplot as plt
import os

# Verify the file exists
test_image_path = r"/content/drive/MyDrive/1sv21cs040/archive
(2)/Multi-class Weather Dataset/Rain/rain10.jpg"
if not os.path.exists(test_image_path):
    print(f"Error: File not found at {test_image_path}")
    # Handle the error appropriately, e.g., exit the script or prompt
for a different path
else:
    from tensorflow.keras.preprocessing import image
    # Load the image with the target size the model expects
    img = image.load_img(test_image_path, target_size=(244, 244))  #
Change target_size to (244, 244)

    plt.imshow(img)
    plt.axis()
    plt.show()

    #convert image into array
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.  # Normalize the pixel values


    # Make predictions
    prediction = model.predict(img_array)
    # Print the prediction
    print(prediction)
```



```
1/1 [==============================] - 0s 95ms/step
[[1.2296852e-01 8.7700039e-01 2.1082580e-05 9.9407334e-06]]
```

```
#interprete the results
prediction = model.predict(img_array)
ind = np.argmax(prediction[0])
print(class_names[ind])
```

```
1/1 [==============================] - 0s 80ms/step
Rain
```