```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Define image size and batch size
IMG_SIZE = 224
BATCH_SIZE = 32

# Define data generators for train, validation and test sets
train_datagen =
ImageDataGenerator(rescale=1./255,validation_split=0.2)


train_generator = train_datagen.flow_from_directory(
    r"/content/drive/MyDrive/weather/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(
    r"/content/drive/MyDrive/weather/Multi-class Weather Dataset",
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)


Found 906 images belonging to 4 classes.
Found 225 images belonging to 4 classes.

# 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']

# 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='categorical_crossentropy',
metrics=['accuracy'])

model.fit(train_generator,validation_data=val_generator,epochs=3)
```

```
Epoch 1/3
29/29 [==============================] - 155s 5s/step - loss: 0.4005 -
accuracy: 0.8620 - val_loss: 0.6475 - val_accuracy: 0.7644
Epoch 2/3
29/29 [==============================] - 122s 4s/step - loss: 0.3088 -
accuracy: 0.8885 - val_loss: 0.5539 - val_accuracy: 0.8044
Epoch 3/3
29/29 [==============================] - 119s 4s/step - loss: 0.2241 -
accuracy: 0.9272 - val_loss: 0.5958 - val_accuracy: 0.8133

<keras.src.callbacks.History at 0x7c17c2a57cd0>
```

```
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
```

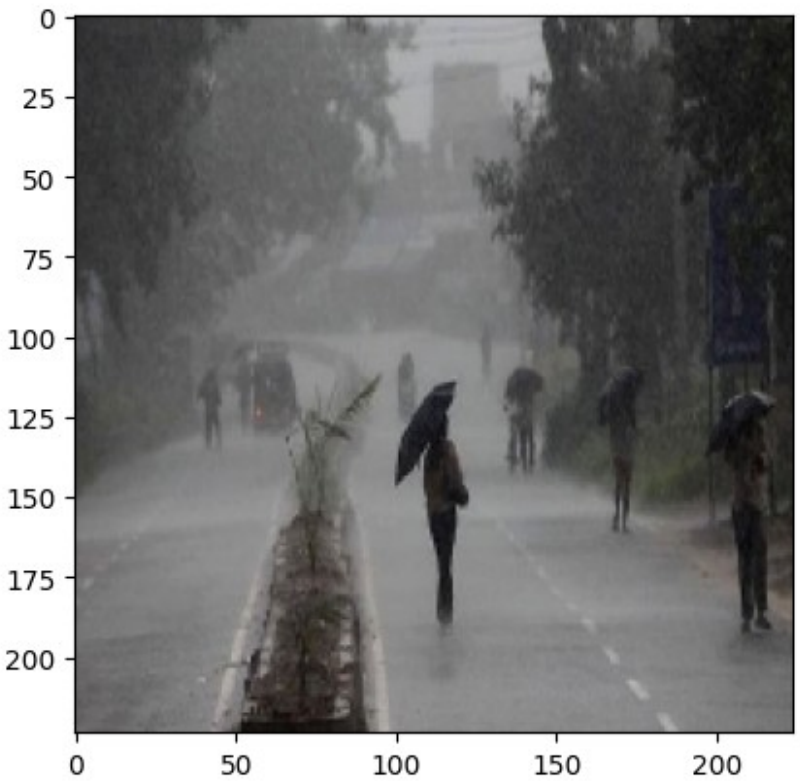
```
# Load and view the image
from matplotlib import pyplot as plt
test_image_path = r"/content/drive/MyDrive/weather/Multi-class Weather
Dataset/Rain/rain10.jpg"
img = image.load_img(test_image_path, target_size=(224, 224))

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 88ms/step
[[0.4150073  0.5355038  0.03923732 0.01025152]]

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

1/1 [==============================] - 0s 65ms/step
Rain
```