multiclass

June 28, 2024

[ ]:
```
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 901 images belonging to 4 classes.
Found 224 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=10)
```

```
Epoch 1/10
29/29 [==============================] - 190s 7s/step - loss: 0.6077 - accuracy:
0.7858 - val_loss: 0.7867 - val_accuracy: 0.7455
Epoch 2/10
29/29 [==============================] - 118s 4s/step - loss: 0.3147 - accuracy:
0.8868 - val_loss: 0.5195 - val_accuracy: 0.8170
Epoch 3/10
29/29 [==============================] - 118s 4s/step - loss: 0.2620 - accuracy:
0.8990 - val_loss: 0.3383 - val_accuracy: 0.8616
Epoch 4/10
29/29 [==============================] - 118s 4s/step - loss: 0.2210 - accuracy:
0.9245 - val_loss: 0.4685 - val_accuracy: 0.8304
Epoch 5/10
29/29 [==============================] - 113s 4s/step - loss: 0.1719 - accuracy:
0.9390 - val_loss: 0.3048 - val_accuracy: 0.8839
Epoch 6/10
29/29 [==============================] - 115s 4s/step - loss: 0.1277 - accuracy:
0.9600 - val_loss: 0.4090 - val_accuracy: 0.8527
Epoch 7/10
29/29 [==============================] - 118s 4s/step - loss: 0.1093 - accuracy:
```

```
0.9567 - val_loss: 0.6851 - val_accuracy: 0.8527
Epoch 8/10
29/29 [==============================] - 113s 4s/step - loss: 0.1326 - accuracy:
0.9534 - val_loss: 0.3721 - val_accuracy: 0.8482
Epoch 9/10
29/29 [==============================] - 117s 4s/step - loss: 0.0985 - accuracy:
0.9656 - val_loss: 0.3870 - val_accuracy: 0.8482
Epoch 10/10
29/29 [==============================] - 115s 4s/step - loss: 0.0591 - accuracy:
0.9834 - val_loss: 0.2817 - val_accuracy: 0.8795
```

[ ]:
```
<keras.src.callbacks.History at 0x78afb1de9b40>
```

[ ]:
```
model.save('weather.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("/content/drive/MyDrive/weather/weather.h5")
print("Model Loaded")
```

```
Model Loaded
```

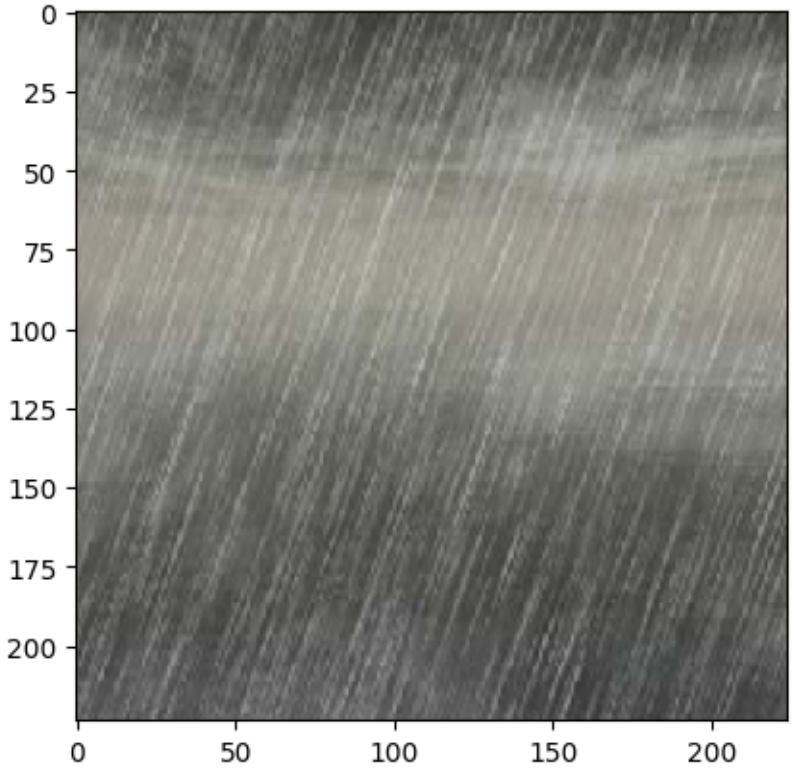
[ ]:
```
# Load and view the image
from matplotlib import pyplot as plt
test_image_path = r"/content/drive/MyDrive/weather/Multi-class Weather Dataset/
 ↪Rain/rain1.jpg"
img = image.load_img(test_image_path, target_size=(224, 224))

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 86ms/step
[[6.9388881e-04 9.9921572e-01 8.3931773e-05 6.4895585e-06]]
```

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

```
1/1 [==============================] - 0s 52ms/step
Cloudy
```