```
#23MAI1015
#23MAI1023

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import random
import cv2
import tqdm as tqdm
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files u

import os
for dirname, _, filenames in os.walk('/content/drive/MyDrive/waether_dataset'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/content/drive/MyDrive/waether_dataset/test.csv.gsheet
/content/drive/MyDrive/waether_dataset/alien_test/foggy_10.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_7.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_5.jpg
/content/drive/MyDrive/waether_dataset/alien_test/Cloud_2.jpg
/content/drive/MyDrive/waether_dataset/alien_test/Cloud_4.jpg
/content/drive/MyDrive/waether_dataset/alien_test/Cloud_1.png
/content/drive/MyDrive/waether_dataset/alien_test/rain_2.png
/content/drive/MyDrive/waether_dataset/alien_test/foggy_6.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_3.jpg
/content/drive/MyDrive/waether_dataset/alien_test/rain_3.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_4.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_9.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_2.jpg
/content/drive/MyDrive/waether_dataset/alien_test/rain_1.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_8.jpg
/content/drive/MyDrive/waether_dataset/alien_test/foggy_1.jpg
/content/drive/MyDrive/waether_dataset/alien_test/rain_5.jpg
/content/drive/MyDrive/waether_dataset/alien_test/Cloud_3.jpeg
/content/drive/MyDrive/waether_dataset/alien_test/rain_4.jpg
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_7.jpg
/content/drive/MyDrive/waether_dataset/alien_test/rain_6.jpg
/content/drive/MyDrive/waether_dataset/alien_test/shine_1.jpg
/content/drive/MyDrive/waether_dataset/alien_test/shine_3.jpg
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_6.jpg
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_4.jpg
/content/drive/MyDrive/waether_dataset/alien_test/shine_2.jpg
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_5.jpg
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_2.jpg
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_1.jpg
```

```
/content/drive/MyDrive/waether_dataset/alien_test/sunrise_3.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy10.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy1.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy107.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy113.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy108.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy126.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy134.png
/content/drive/MyDrive/waether_dataset/foggy/foggy112.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy110.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy138.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy123.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy116.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy132.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy122.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy104.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy124.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy129.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy136.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy130.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy137.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy111.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy131.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy106.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy103.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy11.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy102.jpg
/content/drive/MyDrive/waether_dataset/foggy/foggy115.jpg
```

```
root_dir = "/content/drive/MyDrive/waether_dataset"
os.listdir(root_dir)
```

```
['alien_test',
 'foggy',
 'cloudy',
 'sunrise',
 'shine',
 'rainy',
 'test.csv.gsheet']
```

```
df_cloudy = "/content/drive/MyDrive/waether_dataset/cloudy"
df_foggy = "/content/drive/MyDrive/waether_dataset/foggy"
df_rainy = "/content/drive/MyDrive/waether_dataset/rainy"
df_shine = "/content/drive/MyDrive/waether_dataset/shine"
df_sunrise = "/content/drive/MyDrive/waether_dataset/sunrise"

print("Number of Images in Each Directory:")
print(f"Foggy: {len(os.listdir(df_foggy))}")
print(f"Sunrise: {len(os.listdir(df_sunrise))}")
print(f"Shine: {len(os.listdir(df_shine))}")
print(f"Rainy: {len(os.listdir(df_rainy))}")
print(f"Cloudy: {len(os.listdir(df_cloudy))}")
```

```
Number of Images in Each Directory:
Foggy: 300
Sunrise: 350
Shine: 250
Rainy: 300
Cloudy: 300
```

```
x = []
y = []
dataset =[]
def create_dataset(directory,dir_name):
    for i in tqdm.tqdm(os.listdir(directory)):
        full_path = os.path.join(directory,i)
        try:
            img = cv2.imread(full_path)
            img = cv2.resize(img,(150,150))
        except:
            continue
        x.append(img)
        y.append(dir_name)
    return x,y

x,y= create_dataset(df_foggy,"foggy")
x,y= create_dataset(df_sunrise,"sunrise")
x,y= create_dataset(df_shine,"shine")
x,y= create_dataset(df_rainy,"rainy")
x,y= create_dataset(df_cloudy,"cloudy")
```

```
100%|██████████| 300/300 [00:02<00:00, 100.57it/s]
100%|██████████| 350/350 [00:04<00:00, 73.59it/s]
100%|██████████| 250/250 [00:01<00:00, 131.32it/s]
100%|██████████| 300/300 [00:04<00:00, 60.24it/s]
100%|██████████| 300/300 [00:01<00:00, 197.60it/s]
```

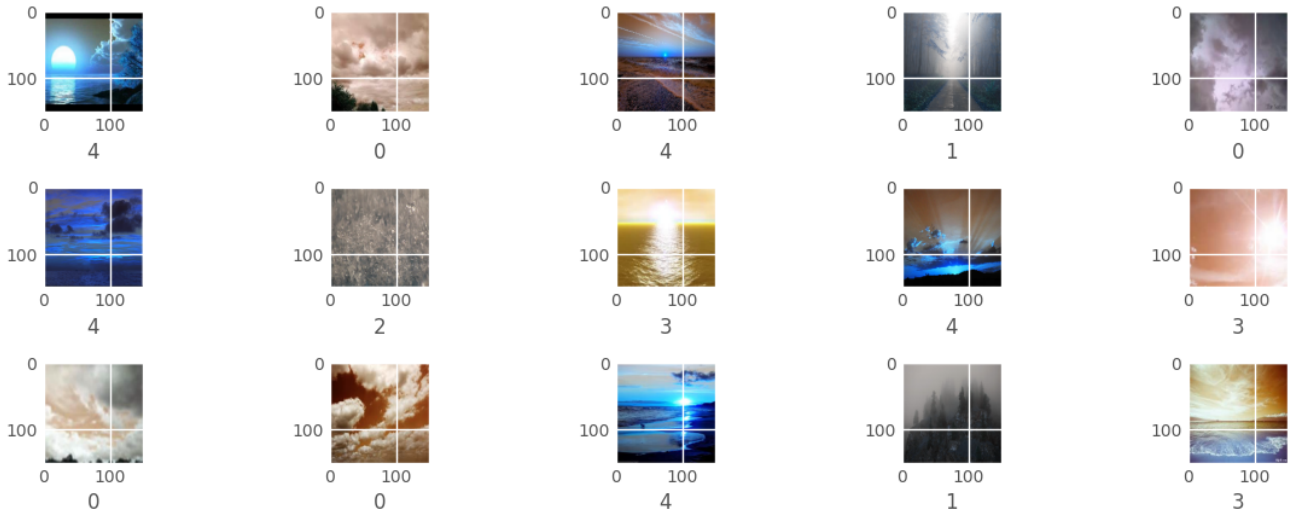
```
x =  np.array(x)
y = np.array(y)
x.shape,y.shape
```

```
((1498, 150, 150, 3), (1498,))
```

```
fig = plt.figure(figsize=(12,7))
for i in range(15):
    sample =  random.choice(range(len(x)))
    image = x[sample]
    category = y[sample]
    plt.subplot(5,5,i+1)
    plt.subplots_adjust(hspace=0.3)
    plt.imshow(image)
    plt.xlabel(category)

plt.tight_layout()
```

```
plt.show()
```




```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)

img_size =150

x_train = np.array(x_train)/255.0
x_test = np.array(x_test)/255.0


x_train = x_train.reshape(-1,img_size,img_size,3)
y_train = np.array(y_train)

x_test = x_test.reshape(-1,img_size,img_size,3)
y_test = np.array(y_test)

from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
y_train_lb = lb.fit_transform(y_train)
y_test_lb = lb.fit_transform(y_test)
```

```
y_train_lb.shape,y_test_lb.shape
```

```
((1198, 5), (300, 5))
```

```
from tensorflow.keras.applications.vgg19 import VGG19
vgg = VGG19(weights = "imagenet",include_top=False,input_shape=(img_size,img_size,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vg
80134624/80134624 [==============================] - 1s 0us/step
```

```
for layer in vgg.layers:
    layer.trainable = False
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten,Dense
model =Sequential()
model.add(vgg)
model.add(Flatten())
model.add(Dense(5,activation="softmax"))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 4, 4, 512)         20024384

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 5)                 40965

=================================================================
Total params: 20065349 (76.54 MB)
Trainable params: 40965 (160.02 KB)
Non-trainable params: 20024384 (76.39 MB)
_________________________________________________________________
```

```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics="accuracy")
```

```
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
checkpoint = ModelCheckpoint("vgg19.h5",monitor="val_accuracy",verbose=1,save_best_only=T
                             save_weights_only=False)
earlystop = EarlyStopping(monitor="val_accuracy",patience=5,verbose=1)
```

```
unique,counts = np.unique(y_train_lb,return_counts=True)
print(unique,counts)
```

```
[0 1] [4792 1198]
```

```
batch_size=32
```

```
batch_size 32
history =  model.fit(x_train,y_train_lb,epochs=15,validation_data=(x_test,y_test_lb),
                     batch_size=32 ,verbose=1,callbacks=[checkpoint,earlystop])
```

```
Epoch 1/15
38/38 [==============================] - ETA: 0s - loss: 0.4604 - accuracy: 0.8556
Epoch 1: val_accuracy improved from -inf to 0.86000, saving model to vgg19.h5
38/38 [==============================] - 491s 13s/step - loss: 0.4604 - accuracy: 0.8
Epoch 2/15
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarnin
  saving_api.save_model(
38/38 [==============================] - ETA: 0s - loss: 0.3296 - accuracy: 0.9115
Epoch 2: val_accuracy improved from 0.86000 to 0.86333, saving model to vgg19.h5
38/38 [==============================] - 413s 11s/step - loss: 0.3296 - accuracy: 0.9
Epoch 3/15
38/38 [==============================] - ETA: 0s - loss: 0.2710 - accuracy: 0.9274
Epoch 3: val_accuracy improved from 0.86333 to 0.89000, saving model to vgg19.h5
38/38 [==============================] - 422s 11s/step - loss: 0.2710 - accuracy: 0.9
Epoch 4/15
38/38 [==============================] - ETA: 0s - loss: 0.2117 - accuracy: 0.9491
Epoch 4: val_accuracy did not improve from 0.89000
38/38 [==============================] - 472s 13s/step - loss: 0.2117 - accuracy: 0.9
Epoch 5/15
38/38 [==============================] - ETA: 0s - loss: 0.1822 - accuracy: 0.9591
Epoch 5: val_accuracy did not improve from 0.89000
38/38 [==============================] - 471s 13s/step - loss: 0.1822 - accuracy: 0.9
Epoch 6/15
38/38 [==============================] - ETA: 0s - loss: 0.1550 - accuracy: 0.9683
Epoch 6: val_accuracy improved from 0.89000 to 0.89667, saving model to vgg19.h5
38/38 [==============================] - 412s 11s/step - loss: 0.1550 - accuracy: 0.9
Epoch 7/15
38/38 [==============================] - ETA: 0s - loss: 0.1403 - accuracy: 0.9716
Epoch 7: val_accuracy did not improve from 0.89667
38/38 [==============================] - 473s 13s/step - loss: 0.1403 - accuracy: 0.9
Epoch 8/15
38/38 [==============================] - ETA: 0s - loss: 0.1260 - accuracy: 0.9750
Epoch 8: val_accuracy did not improve from 0.89667
38/38 [==============================] - 412s 11s/step - loss: 0.1260 - accuracy: 0.9
Epoch 9/15
38/38 [==============================] - ETA: 0s - loss: 0.1108 - accuracy: 0.9783
Epoch 9: val_accuracy did not improve from 0.89667
38/38 [==============================] - 412s 11s/step - loss: 0.1108 - accuracy: 0.9
Epoch 10/15
38/38 [==============================] - ETA: 0s - loss: 0.1008 - accuracy: 0.9816
Epoch 10: val_accuracy did not improve from 0.89667
38/38 [==============================] - 472s 13s/step - loss: 0.1008 - accuracy: 0.9
Epoch 11/15
38/38 [==============================] - ETA: 0s - loss: 0.0871 - accuracy: 0.9841
Epoch 11: val_accuracy improved from 0.89667 to 0.90000, saving model to vgg19.h5
38/38 [==============================] - 472s 13s/step - loss: 0.0871 - accuracy: 0.9
Epoch 12/15
38/38 [==============================] - ETA: 0s - loss: 0.0792 - accuracy: 0.9891
Epoch 12: val_accuracy did not improve from 0.90000
38/38 [==============================] - 472s 13s/step - loss: 0.0792 - accuracy: 0.9
Epoch 13/15
38/38 [==============================] - ETA: 0s - loss: 0.0757 - accuracy: 0.9891
```

```
Epoch 13: val_accuracy did not improve from 0.90000
38/38 [==============================] - 412s 11s/step - loss: 0.0757 - accuracy: 0.9
Epoch 14/15
38/38 [==============================] - ETA: 0s - loss: 0.0705 - accuracy: 0.9883
Epoch 14: val_accuracy improved from 0.90000 to 0.90333, saving model to vgg19.h5
38/38 [==============================] - 472s 13s/step - loss: 0.0705 - accuracy: 0.9
```

```
loss,accuracy = model.evaluate(x_test,y_test_lb)
print(f"Loss: {loss}")
print(f"Accuracy: {accuracy}")
```

```
10/10 [==============================] - 82s 8s/step - loss: 0.3070 - accuracy: 0.896
Loss: 0.30696311593055725
Accuracy: 0.8966666460037231
```

```
y_pred = np.argmax(model.predict(x_test),axis=1)
y_pred[:15]
```

```
10/10 [==============================] - 83s 8s/step
array([2, 0, 0, 0, 3, 3, 2, 3, 4, 1, 3, 3, 0, 0, 1])
```

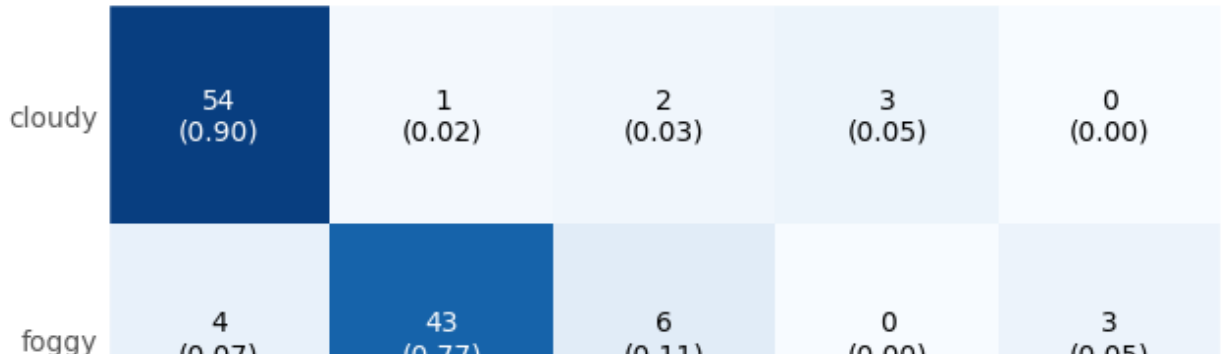
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

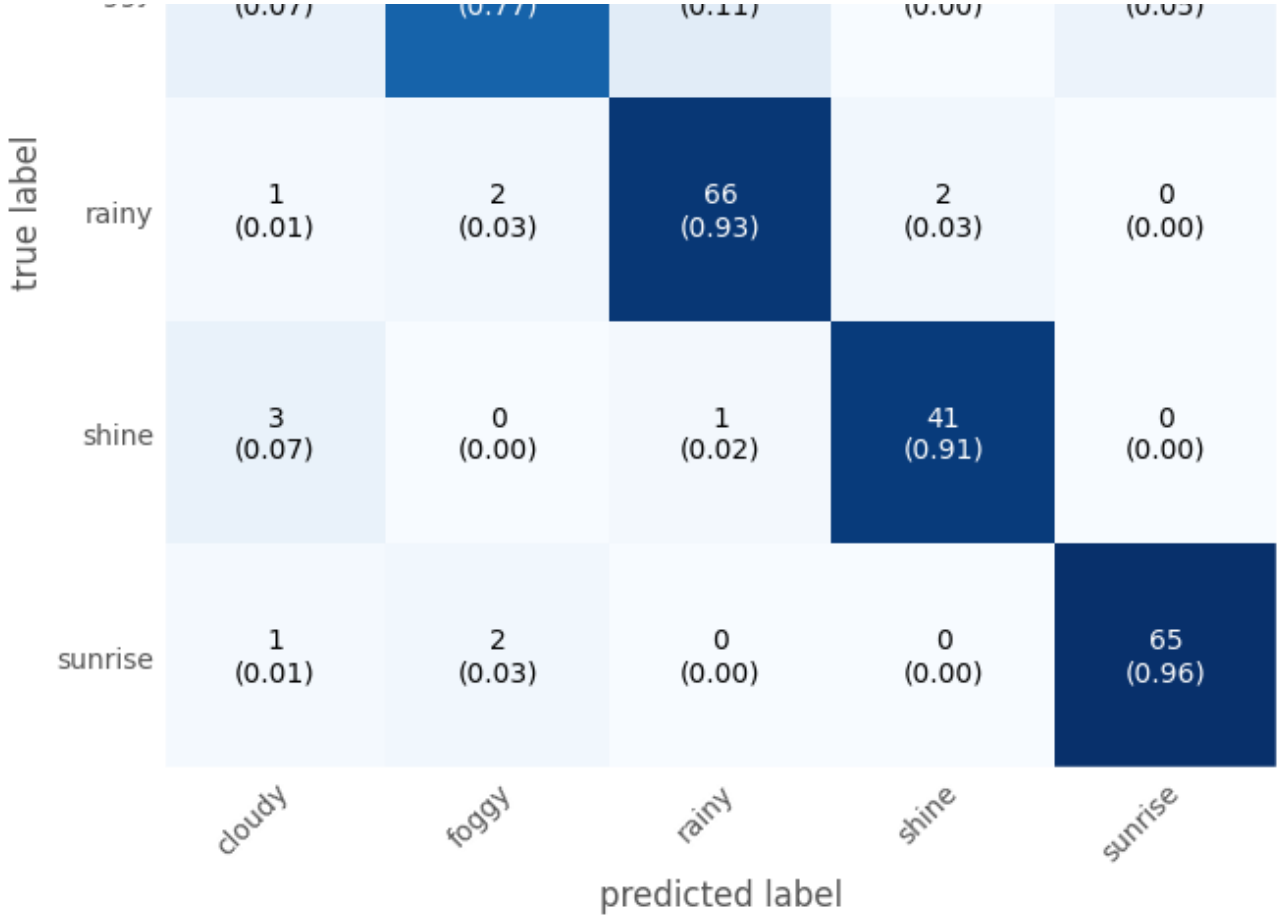
```
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        60
           1       0.90      0.77      0.83        56
           2       0.88      0.93      0.90        71
           3       0.89      0.91      0.90        45
           4       0.96      0.96      0.96        68

    accuracy                           0.90       300
   macro avg       0.90      0.89      0.89       300
weighted avg       0.90      0.90      0.90       300
```

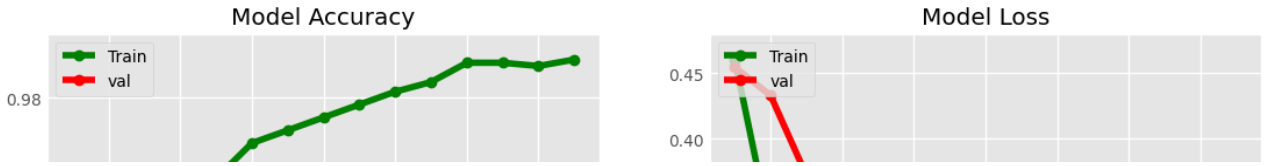
```
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
cm = confusion_matrix(y_test,y_pred)
plot_confusion_matrix(conf_mat = cm,figsize=(8,7),class_names = ["cloudy","foggy","rainy"
                      show_normed = True);
```

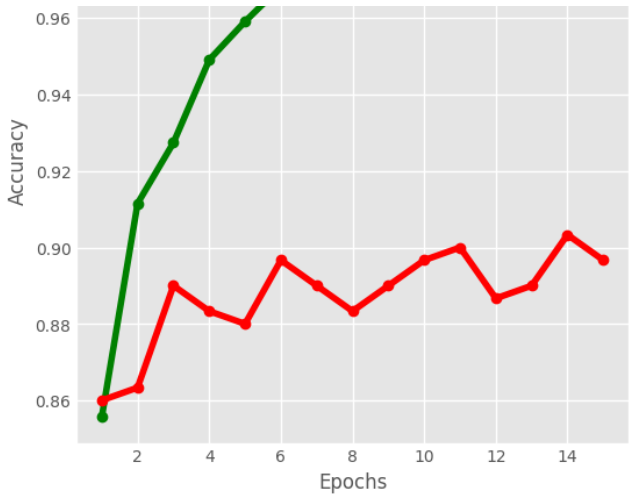


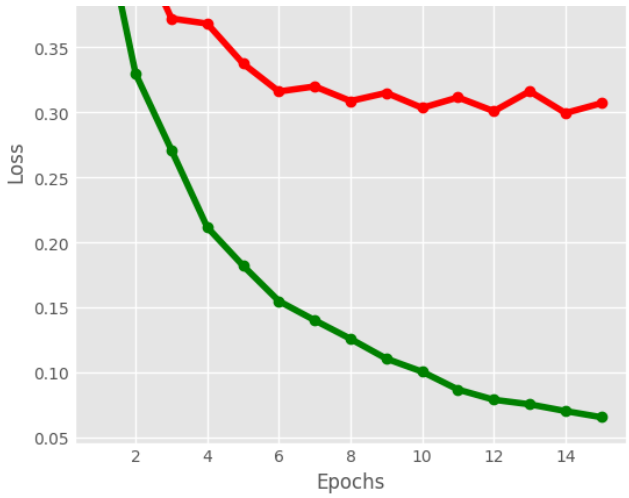



```
plt.style.use("ggplot")
fig = plt.figure(figsize=(12,6))
epochs = range(1,16)
plt.subplot(1,2,1)
plt.plot(epochs,history.history["accuracy"],"go-")
plt.plot(epochs,history.history["val_accuracy"],"ro-")
plt.title("Model Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend(["Train","val"],loc = "upper left")

plt.subplot(1,2,2)
plt.plot(epochs,history.history["loss"],"go-")
plt.plot(epochs,history.history["val_loss"],"ro-")
plt.title("Model Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(["Train","val"],loc = "upper left")
plt.show()
```

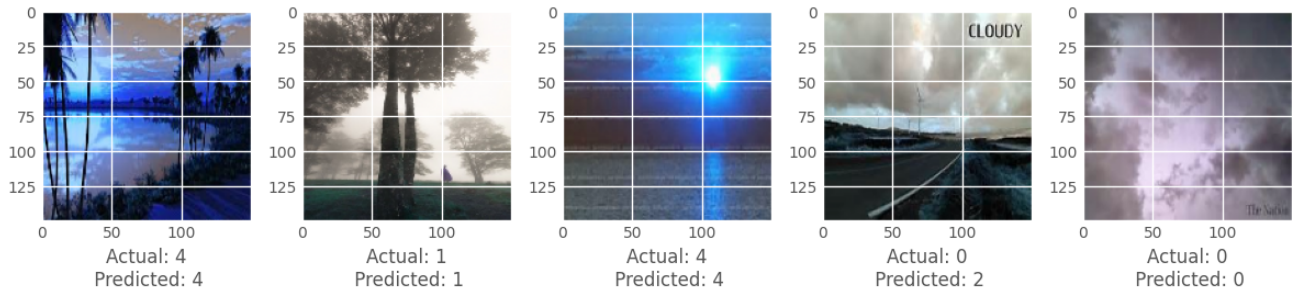


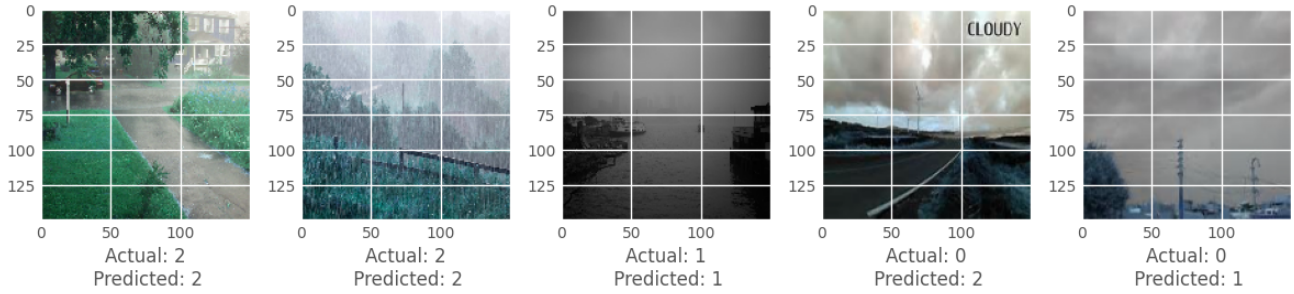





```
plt.figure(figsize=(12,9))
plt.style.use("ggplot")
for i in range(10):
    sample = random.choice(range(len(x_test)))
    plt.subplot(2,5,i+1)
    plt.subplots_adjust(hspace=0.5)
    plt.imshow(x_test[sample])
    plt.xlabel(f"Actual: {y_test[sample]}\n Predicted: {y_pred[sample]}")

plt.tight_layout()
plt.show()
```

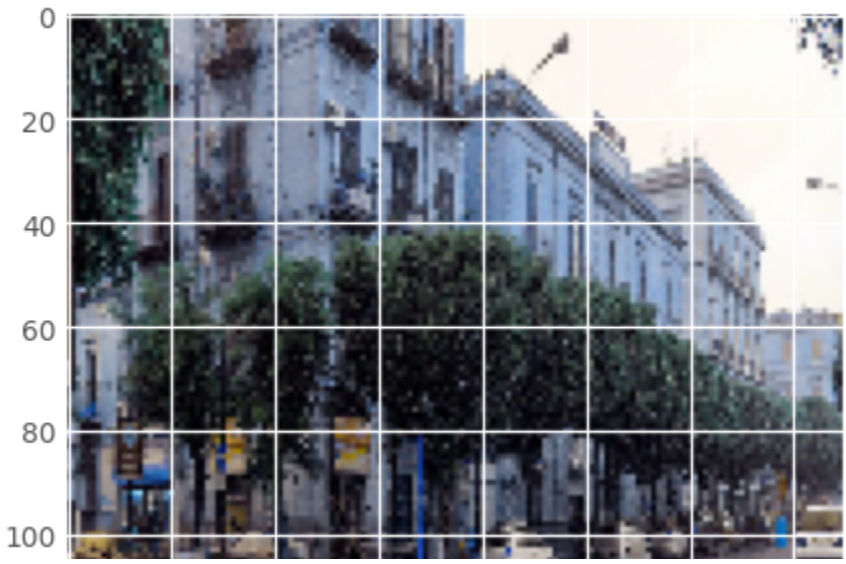


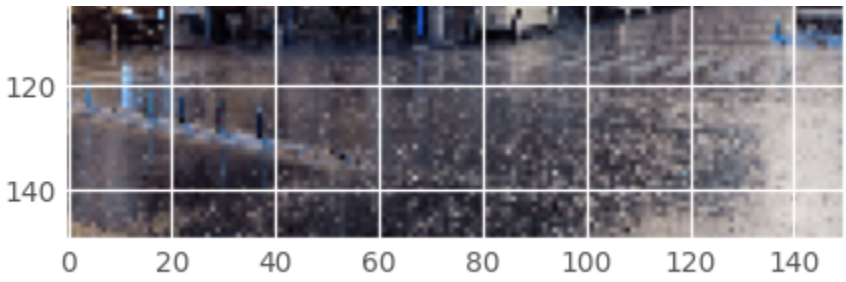



```
import matplotlib as pyplot


res = model.predict(x_test[200:201])
ind = np.argmax(res)
plt.imshow(x_test[200])
y_pred[ind]
```

```
1/1 [==============================] - 0s 269ms/step
0
```




120
140
0
20
40
60
80
100
120
140