brain tumar with Crop image

January 10, 2023

1 Brain_Tumor with crop image

[1]:
```
!pip install imutils
```

```
Requirement already satisfied: imutils in
/home/tanzim/anaconda3/lib/python3.9/site-packages (0.5.4)
```

[2]:
```
import numpy as np
from tqdm import tqdm
import cv2
import os
import imutils
```

[4]:
```
def crop_img(img):
        """
        Finds the extreme points on the image and crops the rectangular out of␣
 ↪them
        """
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (3, 3), 0)

        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)

        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
```

```
        extBot = tuple(c[c[:, :, 1].argmax()][0])
        ADD_PIXELS = 0
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS,␣
 ↪extLeft[0]-ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()

        return new_img

if __name__ == "__main__":
        training = "Training"
        testing = "Testing"
        training_dir = os.listdir(training)
        testing_dir = os.listdir(testing)
        IMG_SIZE = 256

        for dir in training_dir:
                save_path = 'cleaned/Training/'+ dir
                path = os.path.join(training,dir)
                image_dir = os.listdir(path)
                for img in image_dir:
                        image = cv2.imread(os.path.join(path,img))
                        new_img = crop_img(image)
                        new_img = cv2.resize(new_img,(IMG_SIZE,IMG_SIZE))
                        if not os.path.exists(save_path):
                                os.makedirs(save_path)
                        cv2.imwrite(save_path+'/'+img, new_img)

        for dir in testing_dir:
                save_path = 'cleaned/Testing/'+ dir
                path = os.path.join(testing,dir)
                image_dir = os.listdir(path)
                for img in image_dir:
                        image = cv2.imread(os.path.join(path,img))
                        new_img = crop_img(image)
                        new_img = cv2.resize(new_img,(IMG_SIZE,IMG_SIZE))
                        if not os.path.exists(save_path):
                                os.makedirs(save_path)
                        cv2.imwrite(save_path+'/'+img, new_img)
```

[5]:
```
# Necessary imports
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D,␣
 ↪Dropout
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD
```

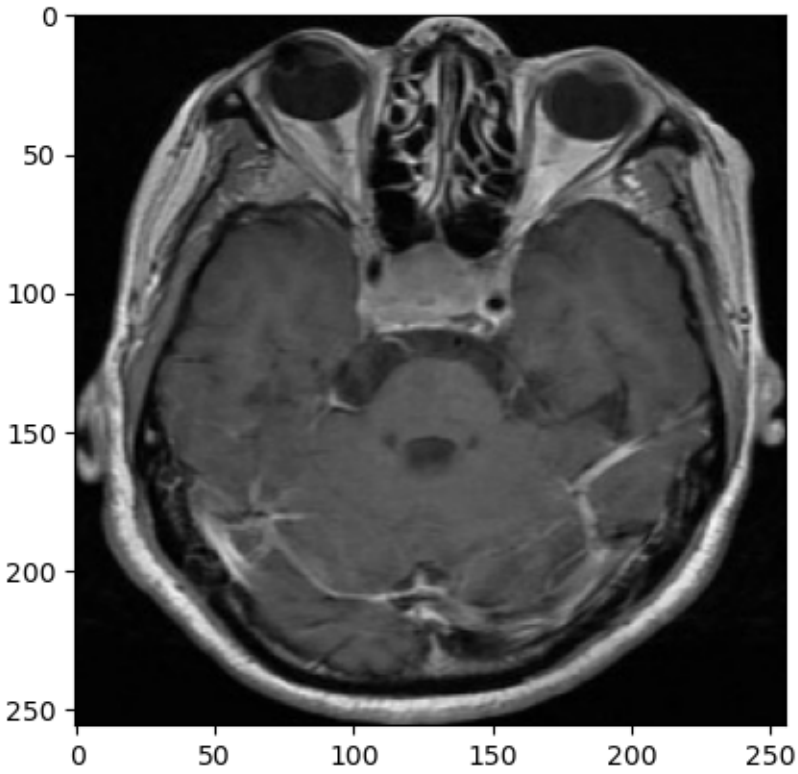
```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,␣
 ↪ModelCheckpoint

import os
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
```

```
2023-01-09 23:56:07.656752: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

[6]:
```
data_dir = ('cleaned/Training')
categories = ['glioma', 'meningioma', 'notumor', 'pituitary']
for i in categories:
    path = os.path.join(data_dir, i)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img))
```

[7]:
```
plt.imshow(img_array);
```



[8]:
```
# The image shape.
img_array.shape
```

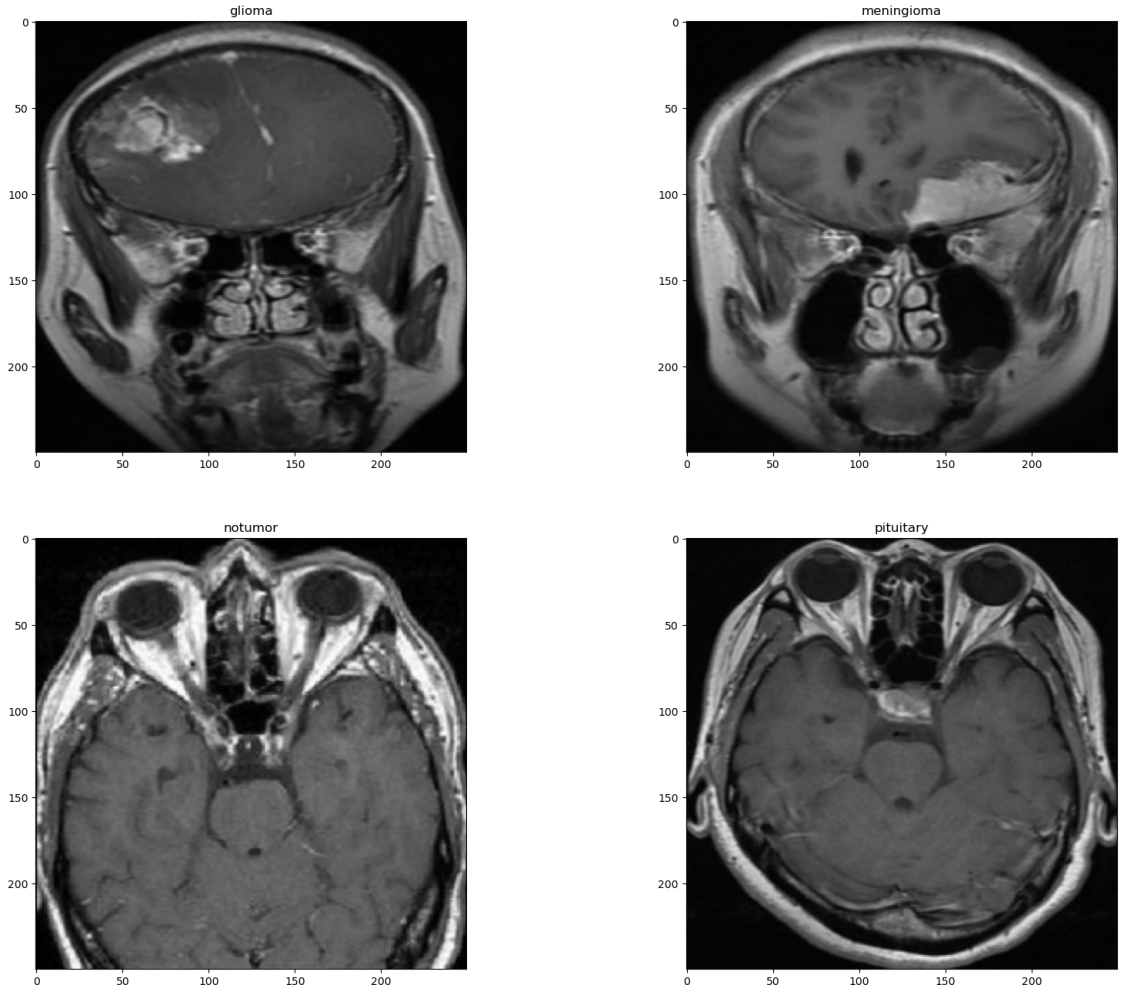
[8]:
```
(256, 256, 3)
```

2 Ploting a image of each brain tumor type

[9]:
```
plt.figure(figsize=(20, 16))

images_path = ['/glioma/Tr-glTr_0000.jpg', '/meningioma/Tr-meTr_0000.jpg', '/
 ↪notumor/Tr-noTr_0000.jpg', '/pituitary/Tr-piTr_0000.jpg']

for i in range(4):
    ax = plt.subplot(2, 2, i + 1)
    img = cv2.imread(data_dir + images_path[i])
    img = cv2.resize(img, (250, 250))
    plt.imshow(img)
    plt.title(categories[i])
```



3 Modelling: CNN

[10]:
```
model1 = Sequential()
# Convolutional layer 1
model1.add(Conv2D(32,(3,3), input_shape=(64, 64, 1), activation='relu'))
model1.add(BatchNormalization())
model1.add(MaxPooling2D(pool_size=(2,2)))

# Convolutional layer 2
model1.add(Conv2D(32,(3,3), activation='relu'))
model1.add(BatchNormalization())
model1.add(MaxPooling2D(pool_size=(2,2)))

model1.add(Flatten())
```

```
# Neural network

model1.add(Dense(units= 252, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(units=252, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(units=4, activation='softmax'))
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001, decay=0.0001,␣
 ↪clipvalue=0.5)
model1.compile(optimizer=optimizer, loss='categorical_crossentropy',
                   metrics= ['categorical_accuracy'])



# using the ImageDataGenerator to prepare the images (Resize, nomalize, etc)

generator_train = ImageDataGenerator(rescale=1./255,
                                     featurewise_center=False,
                                     samplewise_center=False,
                                     featurewise_std_normalization=False,
                                     samplewise_std_normalization=False,
                                     zca_whitening=False,
                                     rotation_range=0,
                                     zoom_range = 0,
                                     width_shift_range=0,
                                     height_shift_range=0,
                                     horizontal_flip=True,
                                     vertical_flip=False)


generator_test = ImageDataGenerator(rescale=1./255,
                                     featurewise_center=False,
                                     samplewise_center=False,
                                     featurewise_std_normalization=False,
                                     samplewise_std_normalization=False,
                                     zca_whitening=False,
                                     rotation_range=0,
                                     zoom_range = 0,
                                     width_shift_range=0,
                                     height_shift_range=0,
                                     horizontal_flip=True,
                                     vertical_flip=False)
```

```
2023-01-09 23:58:32.623119: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
```

```
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-01-09 23:58:32.623679: I
tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool
with default inter op setting: 2. Tune using inter_op_parallelism_threads for
best performance.
```

[11]:
```
# Creating the train and test data.

train = generator_train.flow_from_directory('cleaned/Training',␣
 ↪target_size=(64,64),
                                            batch_size=32, class_mode=␣
 ↪"categorical", color_mode='grayscale')

test = generator_test.flow_from_directory('cleaned/Testing',␣
 ↪target_size=(64,64),
                                            batch_size=32, class_mode=␣
 ↪"categorical", color_mode='grayscale')
```

```
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
```

[12]:
```
# Creating callbacks for the model.
# If the model dosen't continue to improve (loss), the trainning will stop.

# Stop training if loss doesn't keep decreasing.
model1_es = EarlyStopping(monitor = 'loss', min_delta = 1e-11, patience = 12,␣
 ↪verbose = 1)
model1_rlr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience =␣
 ↪6, verbose = 1)

# Automatically saves the best weights of the model, based on best val_accuracy
model1_mcp = ModelCheckpoint(filepath = 'model1_weights.h5', monitor =␣
 ↪'val_categorical_accuracy',
                      save_best_only = True, verbose = 1)

# Fiting the model.
history1 = model1.fit(train, steps_per_epoch=5712//32, epochs=100,␣
 ↪validation_data=test, validation_steps= 1311//32,
                     callbacks=[model1_es, model1_rlr, model1_mcp])
```

```
Epoch 1/100
178/178 [==============================] - ETA: 0s - loss: 0.8936 -
categorical_accuracy: 0.7018
Epoch 1: val_categorical_accuracy improved from -inf to 0.51172, saving model to
model1_weights.h5
178/178 [==============================] - 36s 199ms/step - loss: 0.8936 -
```

```
categorical_accuracy: 0.7018 - val_loss: 4.4978 - val_categorical_accuracy:
0.5117 - lr: 0.0010
Epoch 2/100
178/178 [==============================] - ETA: 0s - loss: 0.4307 -
categorical_accuracy: 0.8433
Epoch 2: val_categorical_accuracy did not improve from 0.51172
178/178 [==============================] - 49s 277ms/step - loss: 0.4307 -
categorical_accuracy: 0.8433 - val_loss: 6.3103 - val_categorical_accuracy:
0.3789 - lr: 0.0010
Epoch 3/100
178/178 [==============================] - ETA: 0s - loss: 0.3027 -
categorical_accuracy: 0.8875
Epoch 3: val_categorical_accuracy improved from 0.51172 to 0.56172, saving model
to model1_weights.h5
178/178 [==============================] - 49s 276ms/step - loss: 0.3027 -
categorical_accuracy: 0.8875 - val_loss: 2.7020 - val_categorical_accuracy:
0.5617 - lr: 0.0010
Epoch 4/100
178/178 [==============================] - ETA: 0s - loss: 0.2275 -
categorical_accuracy: 0.9208
Epoch 4: val_categorical_accuracy improved from 0.56172 to 0.77578, saving model
to model1_weights.h5
178/178 [==============================] - 48s 268ms/step - loss: 0.2275 -
categorical_accuracy: 0.9208 - val_loss: 0.7948 - val_categorical_accuracy:
0.7758 - lr: 0.0010
Epoch 5/100
178/178 [==============================] - ETA: 0s - loss: 0.1820 -
categorical_accuracy: 0.9371
Epoch 5: val_categorical_accuracy improved from 0.77578 to 0.89766, saving model
to model1_weights.h5
178/178 [==============================] - 45s 252ms/step - loss: 0.1820 -
categorical_accuracy: 0.9371 - val_loss: 0.2957 - val_categorical_accuracy:
0.8977 - lr: 0.0010
Epoch 6/100
178/178 [==============================] - ETA: 0s - loss: 0.1258 -
categorical_accuracy: 0.9583
Epoch 6: val_categorical_accuracy did not improve from 0.89766
178/178 [==============================] - 46s 261ms/step - loss: 0.1258 -
categorical_accuracy: 0.9583 - val_loss: 0.5858 - val_categorical_accuracy:
0.7961 - lr: 0.0010
Epoch 7/100
178/178 [==============================] - ETA: 0s - loss: 0.1161 -
categorical_accuracy: 0.9563
Epoch 7: val_categorical_accuracy improved from 0.89766 to 0.95000, saving model
to model1_weights.h5
178/178 [==============================] - 42s 234ms/step - loss: 0.1161 -
categorical_accuracy: 0.9563 - val_loss: 0.1326 - val_categorical_accuracy:
0.9500 - lr: 0.0010
```

```
Epoch 8/100
178/178 [==============================] - ETA: 0s - loss: 0.0940 -
categorical_accuracy: 0.9699
Epoch 8: val_categorical_accuracy did not improve from 0.95000
178/178 [==============================] - 45s 253ms/step - loss: 0.0940 -
categorical_accuracy: 0.9699 - val_loss: 0.2149 - val_categorical_accuracy:
0.9312 - lr: 0.0010
Epoch 9/100
178/178 [==============================] - ETA: 0s - loss: 0.0786 -
categorical_accuracy: 0.9754
Epoch 9: val_categorical_accuracy improved from 0.95000 to 0.95078, saving model
to model1_weights.h5
178/178 [==============================] - 41s 228ms/step - loss: 0.0786 -
categorical_accuracy: 0.9754 - val_loss: 0.1498 - val_categorical_accuracy:
0.9508 - lr: 0.0010
Epoch 10/100
178/178 [==============================] - ETA: 0s - loss: 0.0634 -
categorical_accuracy: 0.9810
Epoch 10: val_categorical_accuracy improved from 0.95078 to 0.95547, saving
model to model1_weights.h5
178/178 [==============================] - 47s 262ms/step - loss: 0.0634 -
categorical_accuracy: 0.9810 - val_loss: 0.1220 - val_categorical_accuracy:
0.9555 - lr: 0.0010
Epoch 11/100
178/178 [==============================] - ETA: 0s - loss: 0.0684 -
categorical_accuracy: 0.9778
Epoch 11: val_categorical_accuracy did not improve from 0.95547
178/178 [==============================] - 47s 265ms/step - loss: 0.0684 -
categorical_accuracy: 0.9778 - val_loss: 0.2696 - val_categorical_accuracy:
0.9086 - lr: 0.0010
Epoch 12/100
178/178 [==============================] - ETA: 0s - loss: 0.0508 -
categorical_accuracy: 0.9812
Epoch 12: val_categorical_accuracy improved from 0.95547 to 0.97266, saving
model to model1_weights.h5
178/178 [==============================] - 42s 235ms/step - loss: 0.0508 -
categorical_accuracy: 0.9812 - val_loss: 0.0784 - val_categorical_accuracy:
0.9727 - lr: 0.0010
Epoch 13/100
178/178 [==============================] - ETA: 0s - loss: 0.0579 -
categorical_accuracy: 0.9798
Epoch 13: val_categorical_accuracy did not improve from 0.97266
178/178 [==============================] - 46s 261ms/step - loss: 0.0579 -
categorical_accuracy: 0.9798 - val_loss: 0.2044 - val_categorical_accuracy:
0.9375 - lr: 0.0010
Epoch 14/100
178/178 [==============================] - ETA: 0s - loss: 0.0365 -
categorical_accuracy: 0.9877
```

```
Epoch 14: val_categorical_accuracy improved from 0.97266 to 0.97422, saving
model to model1_weights.h5
178/178 [==============================] - 46s 257ms/step - loss: 0.0365 -
categorical_accuracy: 0.9877 - val_loss: 0.1115 - val_categorical_accuracy:
0.9742 - lr: 0.0010
Epoch 15/100
178/178 [==============================] - ETA: 0s - loss: 0.0505 -
categorical_accuracy: 0.9850
Epoch 15: val_categorical_accuracy did not improve from 0.97422
178/178 [==============================] - 46s 256ms/step - loss: 0.0505 -
categorical_accuracy: 0.9850 - val_loss: 0.3364 - val_categorical_accuracy:
0.9172 - lr: 0.0010
Epoch 16/100
178/178 [==============================] - ETA: 0s - loss: 0.0505 -
categorical_accuracy: 0.9852
Epoch 16: val_categorical_accuracy did not improve from 0.97422
178/178 [==============================] - 44s 249ms/step - loss: 0.0505 -
categorical_accuracy: 0.9852 - val_loss: 0.2937 - val_categorical_accuracy:
0.9359 - lr: 0.0010
Epoch 17/100
178/178 [==============================] - ETA: 0s - loss: 0.0295 -
categorical_accuracy: 0.9907
Epoch 17: val_categorical_accuracy did not improve from 0.97422
178/178 [==============================] - 42s 236ms/step - loss: 0.0295 -
categorical_accuracy: 0.9907 - val_loss: 0.1335 - val_categorical_accuracy:
0.9680 - lr: 0.0010
Epoch 18/100
178/178 [==============================] - ETA: 0s - loss: 0.0347 -
categorical_accuracy: 0.9898
Epoch 18: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.

Epoch 18: val_categorical_accuracy did not improve from 0.97422
178/178 [==============================] - 46s 261ms/step - loss: 0.0347 -
categorical_accuracy: 0.9898 - val_loss: 0.1507 - val_categorical_accuracy:
0.9625 - lr: 0.0010
Epoch 19/100
178/178 [==============================] - ETA: 0s - loss: 0.0185 -
categorical_accuracy: 0.9933
Epoch 19: val_categorical_accuracy improved from 0.97422 to 0.98125, saving
model to model1_weights.h5
178/178 [==============================] - 72s 406ms/step - loss: 0.0185 -
categorical_accuracy: 0.9933 - val_loss: 0.0919 - val_categorical_accuracy:
0.9812 - lr: 2.0000e-04
Epoch 20/100
178/178 [==============================] - ETA: 0s - loss: 0.0103 -
categorical_accuracy: 0.9965
Epoch 20: val_categorical_accuracy did not improve from 0.98125
178/178 [==============================] - 84s 474ms/step - loss: 0.0103 -
```

```
categorical_accuracy: 0.9965 - val_loss: 0.0881 - val_categorical_accuracy:
0.9781 - lr: 2.0000e-04
Epoch 21/100
178/178 [==============================] - ETA: 0s - loss: 0.0109 -
categorical_accuracy: 0.9974
Epoch 21: val_categorical_accuracy did not improve from 0.98125
178/178 [==============================] - 69s 388ms/step - loss: 0.0109 -
categorical_accuracy: 0.9974 - val_loss: 0.0989 - val_categorical_accuracy:
0.9758 - lr: 2.0000e-04
Epoch 22/100
178/178 [==============================] - ETA: 0s - loss: 0.0073 -
categorical_accuracy: 0.9970
Epoch 22: val_categorical_accuracy did not improve from 0.98125
178/178 [==============================] - 82s 459ms/step - loss: 0.0073 -
categorical_accuracy: 0.9970 - val_loss: 0.0939 - val_categorical_accuracy:
0.9805 - lr: 2.0000e-04
Epoch 23/100
178/178 [==============================] - ETA: 0s - loss: 0.0090 -
categorical_accuracy: 0.9970
Epoch 23: val_categorical_accuracy did not improve from 0.98125
178/178 [==============================] - 82s 463ms/step - loss: 0.0090 -
categorical_accuracy: 0.9970 - val_loss: 0.0869 - val_categorical_accuracy:
0.9789 - lr: 2.0000e-04
Epoch 24/100
178/178 [==============================] - ETA: 0s - loss: 0.0110 -
categorical_accuracy: 0.9968
Epoch 24: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.

Epoch 24: val_categorical_accuracy improved from 0.98125 to 0.98281, saving
model to model1_weights.h5
178/178 [==============================] - 93s 522ms/step - loss: 0.0110 -
categorical_accuracy: 0.9968 - val_loss: 0.1023 - val_categorical_accuracy:
0.9828 - lr: 2.0000e-04
Epoch 25/100
178/178 [==============================] - ETA: 0s - loss: 0.0078 -
categorical_accuracy: 0.9975
Epoch 25: val_categorical_accuracy did not improve from 0.98281
178/178 [==============================] - 92s 517ms/step - loss: 0.0078 -
categorical_accuracy: 0.9975 - val_loss: 0.0810 - val_categorical_accuracy:
0.9828 - lr: 4.0000e-05
Epoch 26/100
178/178 [==============================] - ETA: 0s - loss: 0.0062 -
categorical_accuracy: 0.9979
Epoch 26: val_categorical_accuracy did not improve from 0.98281
178/178 [==============================] - 93s 523ms/step - loss: 0.0062 -
categorical_accuracy: 0.9979 - val_loss: 0.0887 - val_categorical_accuracy:
0.9812 - lr: 4.0000e-05
Epoch 27/100
```

```
178/178 [==============================] - ETA: 0s - loss: 0.0075 -
categorical_accuracy: 0.9975
Epoch 27: val_categorical_accuracy did not improve from 0.98281
178/178 [==============================] - 92s 519ms/step - loss: 0.0075 -
categorical_accuracy: 0.9975 - val_loss: 0.0904 - val_categorical_accuracy:
0.9828 - lr: 4.0000e-05
Epoch 28/100
178/178 [==============================] - ETA: 0s - loss: 0.0043 -
categorical_accuracy: 0.9988
Epoch 28: val_categorical_accuracy did not improve from 0.98281
178/178 [==============================] - 93s 521ms/step - loss: 0.0043 -
categorical_accuracy: 0.9988 - val_loss: 0.0924 - val_categorical_accuracy:
0.9805 - lr: 4.0000e-05
Epoch 29/100
178/178 [==============================] - ETA: 0s - loss: 0.0035 -
categorical_accuracy: 0.9991
Epoch 29: val_categorical_accuracy improved from 0.98281 to 0.98359, saving
model to model1_weights.h5
178/178 [==============================] - 91s 509ms/step - loss: 0.0035 -
categorical_accuracy: 0.9991 - val_loss: 0.0882 - val_categorical_accuracy:
0.9836 - lr: 4.0000e-05
Epoch 30/100
178/178 [==============================] - ETA: 0s - loss: 0.0046 -
categorical_accuracy: 0.9989
Epoch 30: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-06.

Epoch 30: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 93s 523ms/step - loss: 0.0046 -
categorical_accuracy: 0.9989 - val_loss: 0.0986 - val_categorical_accuracy:
0.9773 - lr: 4.0000e-05
Epoch 31/100
178/178 [==============================] - ETA: 0s - loss: 0.0058 -
categorical_accuracy: 0.9979
Epoch 31: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 89s 500ms/step - loss: 0.0058 -
categorical_accuracy: 0.9979 - val_loss: 0.0921 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 32/100
178/178 [==============================] - ETA: 0s - loss: 0.0057 -
categorical_accuracy: 0.9979
Epoch 32: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 52s 293ms/step - loss: 0.0057 -
categorical_accuracy: 0.9979 - val_loss: 0.0991 - val_categorical_accuracy:
0.9789 - lr: 8.0000e-06
Epoch 33/100
178/178 [==============================] - ETA: 0s - loss: 0.0048 -
categorical_accuracy: 0.9981
Epoch 33: val_categorical_accuracy did not improve from 0.98359
```

```
178/178 [==============================] - 53s 297ms/step - loss: 0.0048 -
categorical_accuracy: 0.9981 - val_loss: 0.0959 - val_categorical_accuracy:
0.9789 - lr: 8.0000e-06
Epoch 34/100
178/178 [==============================] - ETA: 0s - loss: 0.0064 -
categorical_accuracy: 0.9979
Epoch 34: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 44s 246ms/step - loss: 0.0064 -
categorical_accuracy: 0.9979 - val_loss: 0.0909 - val_categorical_accuracy:
0.9805 - lr: 8.0000e-06
Epoch 35/100
178/178 [==============================] - ETA: 0s - loss: 0.0054 -
categorical_accuracy: 0.9982
Epoch 35: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 48s 269ms/step - loss: 0.0054 -
categorical_accuracy: 0.9982 - val_loss: 0.0995 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 36/100
178/178 [==============================] - ETA: 0s - loss: 0.0061 -
categorical_accuracy: 0.9979
Epoch 36: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 45s 255ms/step - loss: 0.0061 -
categorical_accuracy: 0.9979 - val_loss: 0.0778 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 37/100
178/178 [==============================] - ETA: 0s - loss: 0.0050 -
categorical_accuracy: 0.9984
Epoch 37: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 47s 266ms/step - loss: 0.0050 -
categorical_accuracy: 0.9984 - val_loss: 0.0812 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 38/100
178/178 [==============================] - ETA: 0s - loss: 0.0057 -
categorical_accuracy: 0.9989
Epoch 38: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 62s 347ms/step - loss: 0.0057 -
categorical_accuracy: 0.9989 - val_loss: 0.0901 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 39/100
178/178 [==============================] - ETA: 0s - loss: 0.0040 -
categorical_accuracy: 0.9989
Epoch 39: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 76s 427ms/step - loss: 0.0040 -
categorical_accuracy: 0.9989 - val_loss: 0.1036 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 40/100
178/178 [==============================] - ETA: 0s - loss: 0.0049 -
categorical_accuracy: 0.9982
```

```
Epoch 40: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 73s 412ms/step - loss: 0.0049 -
categorical_accuracy: 0.9982 - val_loss: 0.0883 - val_categorical_accuracy:
0.9805 - lr: 8.0000e-06
Epoch 41/100
178/178 [==============================] - ETA: 0s - loss: 0.0057 -
categorical_accuracy: 0.9981
Epoch 41: val_categorical_accuracy did not improve from 0.98359
178/178 [==============================] - 63s 356ms/step - loss: 0.0057 -
categorical_accuracy: 0.9981 - val_loss: 0.0917 - val_categorical_accuracy:
0.9820 - lr: 8.0000e-06
Epoch 41: early stopping
```

[13]:
```
model1.evaluate(test)
```

```
41/41 [==============================] - 3s 72ms/step - loss: 0.0811 -
categorical_accuracy: 0.9825
```

[13]:
```
[0.08108488470315933, 0.9824561476707458]
```

[14]:
```
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 batch_normalization (BatchN  (None, 62, 62, 32)       128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 batch_normalization_1 (Batc  (None, 29, 29, 32)       128
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 252)               1580796

 dropout (Dropout)           (None, 252)               0
```

```
 dense_1 (Dense)             (None, 252)               63756

 dropout_1 (Dropout)         (None, 252)               0

 dense_2 (Dense)             (None, 4)                 1012

=================================================================
Total params: 1,655,388
Trainable params: 1,655,260
Non-trainable params: 128
_________________________________________________________________
```

[ ]: