TEAM ID : PNT2022TMID22410

DATE : 10/10/2022

In [ ]:

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:

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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.moun
t("/content/gdrive", force_remount=True).
```

1) Download the Dataset

In [ ]:

```
%matplotlib inline

from skimage.io import imread
from skimage import exposure, color
from skimage.transform import resize

import keras
from keras import backend as K
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import np_utils
```

In [ ]:

```
!unzip /content/gdrive/MyDrive/Flowers-Dataset.zip
```

2) Image Augmentation

In [ ]:

```
def imgGen(img, zca=False, rotation=0., w_shift=0., h_shift=0., shear=0., zoom=0., h_fli
p=False, v_flip=False,  preprocess_fcn=None, batch_size=9):
    datagen = ImageDataGenerator(
            zca_whitening=zca,
            rotation_range=rotation,
            width_shift_range=w_shift,
            height_shift_range=h_shift,
            shear_range=shear,
            zoom_range=zoom,
            fill_mode='nearest',
            horizontal_flip=h_flip,
            vertical_flip=v_flip,
            preprocessing_function=preprocess_fcn,
            data_format=K.image_data_format())

    datagen.fit(img)

    i=0
    for img_batch in datagen.flow(img, batch_size=9, shuffle=False):
        for img in img_batch:
```

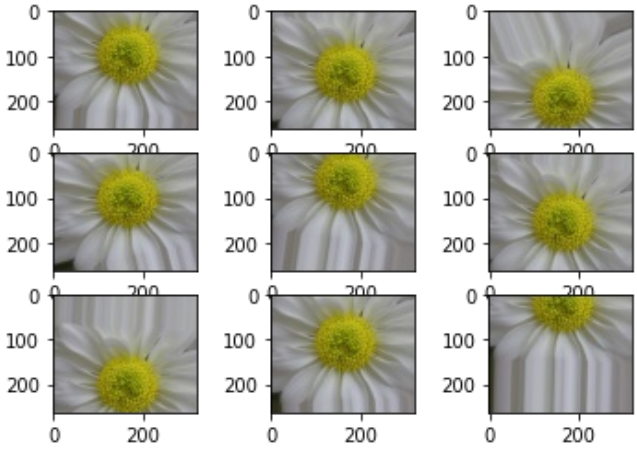
```
            plt.subplot(330 + 1 + i)
            plt.imshow(img)
            i=i+1
        if i >= batch_size:
            break
    plt.show()
```

In [ ]:

```
img = imread("/content/flowers/daisy/100080576_f52e8ee070_n.jpg")
plt.imshow(img)
plt.show()

img = img.astype('float32')
img /= 255
h_dim = np.shape(img)[0]
w_dim = np.shape(img)[1]
num_channel = np.shape(img)[2]
img = img.reshape(1, h_dim, w_dim, num_channel)
print(img.shape)
imgGen(img, rotation=30, h_shift=0.5)
```


```
(1, 263, 320, 3)
```



In [ ]:

```
def AHE(img):
    img_adapteq = exposure.equalize_adapthist(img, clip_limit=0.03)
    return img_adapteq
```

In [ ]:

```
batch_size = 64
num_classes = 2
epochs = 10

img_rows , img_cols = 32, 32

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)
```

```
train_picks = np.ravel(np.logical_or(y_train==3,y_train==5))
test_picks = np.ravel(np.logical_or(y_test==3,y_test==5))

y_train = np.array(y_train[train_picks]==5,dtype=int)
y_test = np.array(y_test[test_picks]==5,dtype=int)

x_train = x_train[train_picks]
x_test = x_test[test_picks]

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 3, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 3, img_rows, img_cols)
    input_shape = (3, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 3)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 3)
    input_shape = (img_rows, img_cols, 3)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

y_train = keras.utils.np_utils.to_categorical(np.ravel(y_train), num_classes)
y_test = keras.utils.np_utils.to_categorical(np.ravel(y_test), num_classes)
```

```
x_train shape: (50000, 32, 32, 3)
x_train shape: (10000, 32, 32, 3)
10000 train samples
2000 test samples
```

3) Create Model

In [ ]:

```
model = Sequential()
```

4) Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output)

In [ ]:

```
model.add(Conv2D(4, kernel_size=(3, 3),activation='relu'))
model.add(Conv2D(8, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

5) Compile The Model

In [ ]:

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer="adadelta",
              metrics=['accuracy'])
```

6) Fit The Model

In [ ]:

```
augmentation=True
```

```
if augmentation==True:
    datagen = ImageDataGenerator(
            rotation_range=0,
            width_shift_range=0,
            height_shift_range=0,
            shear_range=0,
            zoom_range=0,
            horizontal_flip=True,
            fill_mode='nearest',
            preprocessing_function = AHE)

    datagen.fit(x_train)

    history = model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_size),
                    steps_per_epoch=x_train.shape[0] // batch_size,
                    epochs=epochs,
                    validation_data=(x_test, y_test))
else:
    history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:19: UserWarning: `Model.fit_
generator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
```

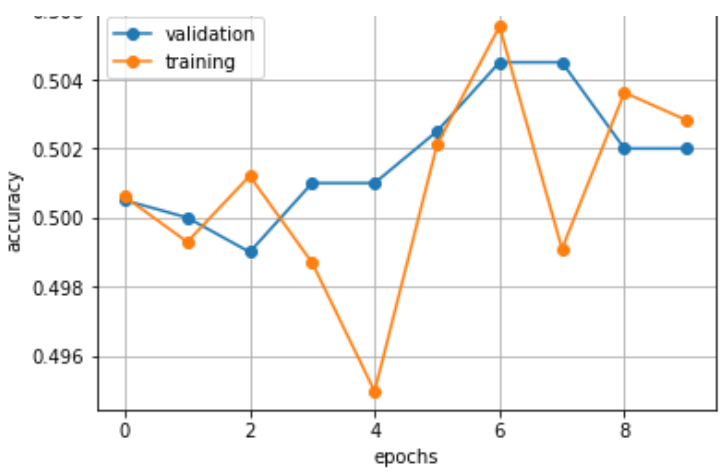
```
Epoch 1/10
156/156 [==============================] - 59s 377ms/step - loss: 0.7148 - accuracy: 0.50
06 - val_loss: 0.6954 - val_accuracy: 0.5005
Epoch 2/10
156/156 [==============================] - 59s 376ms/step - loss: 0.7147 - accuracy: 0.49
93 - val_loss: 0.6943 - val_accuracy: 0.5000
Epoch 3/10
156/156 [==============================] - 58s 371ms/step - loss: 0.7087 - accuracy: 0.50
12 - val_loss: 0.6936 - val_accuracy: 0.4990
Epoch 4/10
156/156 [==============================] - 60s 386ms/step - loss: 0.7065 - accuracy: 0.49
87 - val_loss: 0.6931 - val_accuracy: 0.5010
Epoch 5/10
156/156 [==============================] - 59s 378ms/step - loss: 0.7059 - accuracy: 0.49
50 - val_loss: 0.6927 - val_accuracy: 0.5010
Epoch 6/10
156/156 [==============================] - 58s 372ms/step - loss: 0.7028 - accuracy: 0.50
21 - val_loss: 0.6924 - val_accuracy: 0.5025
Epoch 7/10
156/156 [==============================] - 58s 371ms/step - loss: 0.7008 - accuracy: 0.50
55 - val_loss: 0.6922 - val_accuracy: 0.5045
Epoch 8/10
156/156 [==============================] - 58s 370ms/step - loss: 0.7023 - accuracy: 0.49
91 - val_loss: 0.6921 - val_accuracy: 0.5045
Epoch 9/10
156/156 [==============================] - 57s 367ms/step - loss: 0.7000 - accuracy: 0.50
36 - val_loss: 0.6920 - val_accuracy: 0.5020
Epoch 10/10
156/156 [==============================] - 57s 368ms/step - loss: 0.6976 - accuracy: 0.50
28 - val_loss: 0.6919 - val_accuracy: 0.5020
```

7) Save and Test the Model

In [ ]:

```
plt.plot(history.epoch,history.history['val_accuracy'],'-o',label='validation')
plt.plot(history.epoch,history.history['accuracy'],'-o',label='training')

plt.legend(loc=0)
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid(True)
```


validation
training
0.504
0.502
0.500
0.498
0.496
accuracy
0
2
4
6
8
epochs