TEAM ID:PNT2022TMID45080

In [ ]:

```
import
numpy
as
np
import
seaborn
as
sns
import
pandas
```

```
d
a
s

a
s

p
d
import matplotlib.pyplot as plt
```

In [ ]:

```
from
google.co
lab
import
drive
drive.mou
nt('/cont
ent/gdriv
e')
```

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

- **SDownload 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, Flattenfrom
keras.layers import Conv2D,
MaxPooling2D
```

```
from keras.preprocessing.image
import ImageDataGeneratorfrom
keras.utils import np_utils
```

In [ ]:

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

- **Image Augmentation**

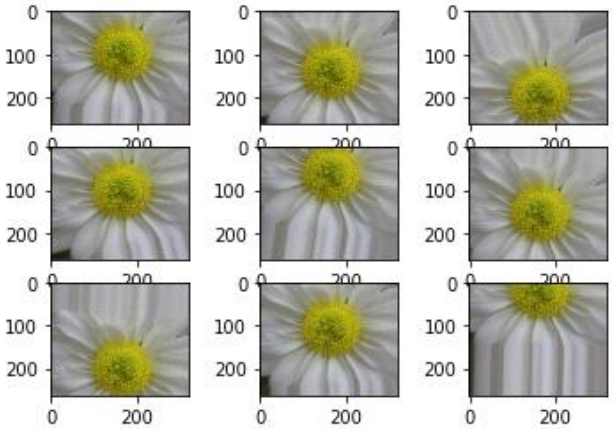
In [ ]:

```
def imgGen(img, zca=False, rotation=0., w_shift=0., h_shift=0., shear=0.,
zoom=0., h_flip=False, v_flip=False, preprocess_fcn=None, batch_size=9):
    datagen =
            ImageDataGene
            rator(
            zca_whitening
            =zca,
            rotation_rang
            e=rotation,
            width_shift_r
            ange=w_shift,
            height_shift_
            range=h_shift
            ,
            shear_range=s
            hear,
            zoom_range=zo
            om,
            fill_mode='ne
            arest',
            horizontal_fl
            ip=h_flip,
            vertical_flip
            =v_flip,
            preprocessing_funct
            ion=preprocess_fcn,
            data_format=K.imag
            e_data_format())
    d

    a

    t

    a

    g

    e

    n

    .
```

```
f
i
t
(
i
m
g
)


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

In [ ]:


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



In [ ]:

In [ ]:

```
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,dt
ype=int)y_test =
np.array(y_test[test_picks]==5,dtyp
e=int)

x_train =
x_train[train_p
icks]x_test =
x_test[test_pic
ks]

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('f
loat32')x_test =
x_test.astype('fl
oat32') 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
```

- **Create Model**

In [ ]:

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

In [ ]:

- **Compile The Model**

In [ ]:

- **Fit The Model**

In [ ]:

/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.which supports generators.	Please	use `Model.fit`,
Epoch 1/10 156/156 [==============================] - 59s 377ms/step - loss:	0.7148	- accuracy: 0.50
06 - val_loss: 0.6954 - val_accuracy: 0.5005Epoch 2/10 156/156 [==============================] - 59s 376ms/step - loss:	0.7147	- accuracy: 0.49
93 - val_loss: 0.6943 - val_accuracy: 0.5000Epoch 3/10 156/156 [==============================] - 58s 371ms/step - loss:	0.7087	- accuracy: 0.50
12 - val_loss: 0.6936 - val_accuracy: 0.4990Epoch 4/10 156/156 [==============================] - 60s 386ms/step - loss:	0.7065	- accuracy: 0.49
87 - val_loss: 0.6931 - val_accuracy: 0.5010Epoch 5/10 156/156 [==============================] - 59s 378ms/step - loss:	0.7059	- accuracy: 0.49
50 - val_loss: 0.6927 - val_accuracy: 0.5010Epoch 6/10 156/156 [==============================] - 58s 372ms/step - loss:	0.7028	- accuracy: 0.50
21 - val_loss: 0.6924 - val_accuracy: 0.5025Epoch 7/10 156/156 [==============================] - 58s 371ms/step - loss:	0.7008	- accuracy: 0.50
55 - val_loss: 0.6922 - val_accuracy: 0.5045Epoch 8/10		

156/156 [==============================] - 58s 370ms/step - loss:	0.7023	- accuracy: 0.49
91 - val_loss: 0.6921 - val_accuracy: 0.5045Epoch 9/10 156/156 [==============================] - 57s 367ms/step - loss:	0.7000	- accuracy: 0.50
36 - val_loss: 0.6920 - val_accuracy: 0.5020Epoch 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 [ ]: