PROBLEM STATEMENT: Build CNN Model for Classification Of Flowers

Mounting drive

```
from google.colab import drive
```

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

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

DATA AUGMENTATION

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen =
ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,
vertical_flip=False,validation_split=0.2)
```

```
test_datagen = ImageDataGenerator(rescale=1./255,validation_split=0.2)
```

```
x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/
Untitled folder/Flowers-Dataset
(4)/flowers",target_size=(64,64),class_mode='categorical',batch_size=1
00,subset = 'training')
```

```
Found 3457 images belonging to 5 classes.
```

```
x_test=test_datagen.flow_from_directory(r"/content/drive/MyDrive/
Untitled folder/Flowers-Dataset
(4)/flowers",target_size=(64,64),class_mode='categorical',batch_size=1
00,subset = 'validation')
```

```
Found 860 images belonging to 5 classes.
```

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

CNN MODEL(Adding Layers :Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output)

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import
Dense,Convolution2D,MaxPooling2D,Flatten
```

```
model=Sequential()
```

```
model.add(Convolution2D(32,
(3,3),input_shape=(64,64,3),activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
model.add(Flatten())

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

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

 flatten (Flatten)           (None, 30752)             0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

```
#hidden layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(75,activation='relu'))
model.add(Dense(5,activation='softmax'))#op layer

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics
=['accuracy'])
```

FIT THE MODEL

```
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_da
ta=x_test,validation_steps=len(x_test),epochs=10)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  """Entry point for launching an IPython kernel.

Epoch 1/10
35/35 [==============================] - 27s 750ms/step - loss: 0.9495
- accuracy: 0.6312 - val_loss: 1.0603 - val_accuracy: 0.5953
Epoch 2/10
35/35 [==============================] - 17s 501ms/step - loss: 0.8961
- accuracy: 0.6503 - val_loss: 1.0812 - val_accuracy: 0.5826
Epoch 3/10
35/35 [==============================] - 18s 505ms/step - loss: 0.8515
- accuracy: 0.6743 - val_loss: 1.0225 - val_accuracy: 0.6244
Epoch 4/10
35/35 [==============================] - 18s 521ms/step - loss: 0.7963
```

```
- accuracy: 0.6963 - val_loss: 1.0625 - val_accuracy: 0.6070
Epoch 5/10
35/35 [==============================] - 17s 502ms/step - loss: 0.7726
- accuracy: 0.7081 - val_loss: 1.0203 - val_accuracy: 0.6105
Epoch 6/10
35/35 [==============================] - 17s 499ms/step - loss: 0.7184
- accuracy: 0.7249 - val_loss: 1.0809 - val_accuracy: 0.6326
Epoch 7/10
35/35 [==============================] - 18s 530ms/step - loss: 0.6644
- accuracy: 0.7495 - val_loss: 1.1533 - val_accuracy: 0.6081
Epoch 8/10
35/35 [==============================] - 17s 500ms/step - loss: 0.6592
- accuracy: 0.7524 - val_loss: 1.0465 - val_accuracy: 0.6349
Epoch 9/10
35/35 [==============================] - 17s 500ms/step - loss: 0.6036
- accuracy: 0.7712 - val_loss: 0.9939 - val_accuracy: 0.6581
Epoch 10/10
35/35 [==============================] - 19s 531ms/step - loss: 0.5648
- accuracy: 0.7926 - val_loss: 1.0939 - val_accuracy: 0.6291

<keras.callbacks.History at 0x7f40c213dbd0>
```

SAVING THE MODEL

```
model.save('flowers.h5')
```

TESTING THE MODEL

```
import numpy as np
from tensorflow.keras.preprocessing import image

img = image.load_img('/content/drive/MyDrive/Untitled folder/Flowers-
Dataset
(4)/flowers/rose/102501987_3cdb8e5394_n.jpg' ,target_size=(64,64))

img
```

```
x=image.img_to_array(img)

x

array([[[ 6.,  6.,  4.],
        [21., 22., 16.],
        [12., 13.,  8.],
        ...,
```

```
         [ 1.,  1.,  0.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]],

        [[20., 21., 16.],
         [ 7.,  7.,  5.],
         [22., 23., 18.],
         ...,
         [ 0.,  0.,  0.],
         [ 2.,  2.,  0.],
         [ 1.,  1.,  0.]],

        [[16., 17., 12.],
         [20., 21., 16.],
         [11., 12.,  7.],
         ...,
         [ 1.,  1.,  0.],
         [ 1.,  1.,  0.],
         [ 1.,  1.,  0.]],

        ...,

        [[13., 14.,  9.],
         [ 9.,  9.,  9.],
         [ 6.,  6.,  4.],
         ...,
         [ 1.,  1.,  0.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]],

        [[ 5.,  5.,  3.],
         [ 7.,  7.,  5.],
         [14., 15., 10.],
         ...,
         [23., 23., 21.],
         [ 1.,  1.,  0.],
         [ 1.,  1.,  0.]],

        [[ 2.,  2.,  0.],
         [ 7.,  7.,  5.],
         [ 3.,  3.,  1.],
         ...,
         [22., 22., 20.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]]], dtype=float32)
```

```
x=np.expand_dims(x,axis=0)
x
```

```
array([[[[ 6.,  6.,  4.],
         [21., 22., 16.],
         [12., 13.,  8.],
         ...,
         [ 1.,  1.,  0.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]],

        [[20., 21., 16.],
         [ 7.,  7.,  5.],
         [22., 23., 18.],
         ...,
         [ 0.,  0.,  0.],
         [ 2.,  2.,  0.],
         [ 1.,  1.,  0.]],

        [[16., 17., 12.],
         [20., 21., 16.],
         [11., 12.,  7.],
         ...,
         [ 1.,  1.,  0.],
         [ 1.,  1.,  0.],
         [ 1.,  1.,  0.]],

        ...,

        [[13., 14.,  9.],
         [ 9.,  9.,  9.],
         [ 6.,  6.,  4.],
         ...,
         [ 1.,  1.,  0.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]],

        [[ 5.,  5.,  3.],
         [ 7.,  7.,  5.],
         [14., 15., 10.],
         ...,
         [23., 23., 21.],
         [ 1.,  1.,  0.],
         [ 1.,  1.,  0.]],

        [[ 2.,  2.,  0.],
         [ 7.,  7.,  5.],
         [ 3.,  3.,  1.],
         ...,
         [22., 22., 20.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]]]], dtype=float32)
```

```
model.predict(x)
```

```
array([[0., 0., 1., 0., 0.]], dtype=float32)
```

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

```
op = ['daisy','dandelion','rose','sunflower','tulip']
pred = np.argmax(model.predict(x))
op[pred]
```

```
{"type":"string"}
```

```
img = image.load_img('/content/drive/MyDrive/Untitled folder/Flowers-
Dataset
(4)/flowers/tulip/100930342_92e8746431_n.jpg',target_size=(64,64))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
pred = np.argmax(model.predict(x))
op[pred]
```

```
{"type":"string"}
```