ASSIGNMENT-3

Build CNN Model for Classification Of Flowers

Assignment Date	07 October 2022
Student Name	P.MANIMARAN
Student Roll Number	420619104023
Maximum marks	2 Marks

1. Download the data set : Dataset

https://drive.google.com/file/d/1xkynpL15pt6KT3YSlDimu4A5iRU9qYck/view

```
Total number of flowers in the dataset:  4326
Flowers in each category:
dandelion    1055
tulip         984
rose          784
daisy         769
sunflower     734
Name: category, dtype: int64
```

2. Image Augmentation

```
[4] from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
[5] train_datagen = ImageDataGenerator(
            featurewise_center=False,  # set input mean to 0 over the dataset
            samplewise_center=False,  # set each sample mean to 0
            featurewise_std_normalization=False,  # divide inputs by std of the dataset
            samplewise_std_normalization=False,  # divide each input by its std
            zca_whitening=False,  # apply ZCA whitening
            rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
            zoom_range = 0.1, # Randomly zoom image
            width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
            height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
            horizontal_flip=True,  # randomly flip images
            vertical_flip=False)  # randomly flip image
```

3. Create the Model

```
[6] test_datagen = ImageDataGenerator(
            featurewise_center=False,  # set input mean to 0 over the dataset
            samplewise_center=False,  # set each sample mean to 0
            featurewise_std_normalization=False,  # divide inputs by std of the dataset
            samplewise_std_normalization=False,  # divide each input by its std
            zca_whitening=False,  # apply ZCA whitening
            rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
            zoom_range = 0.1, # Randomly zoom image
            width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
            height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
            horizontal_flip=True,  # randomly flip images
            vertical_flip=False)  # randomly flip image
```

```
[ ] train = train_datagen.flow_from_directory(r"E:\SB\Dataset\Training",target_size=(64,64),batch_size=32,class_mode="categorical")
```

Found 1717 images belonging to 2 classes

```
[ ] test = test_datagen.flow_from_directory(r"E:\SB\Dataset\Testing",target_size=(64,64),batch_size=32,class_mode="categorical")
```

Found 2600 images belonging to 3 classes

```
#build the model
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras import layers
model = tf.keras.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation='relu', input_shape=(32,32,3)))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(5,  activation='softmax'))
```

4.Add Layers (Convolution, Max Pooling ,Flatten,Dense-(Hidden Layers),Output)

Convolution layer:

```
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (150,150,3)))
# 32 indicates => no of feature detectors
#(5,5)=> kernel size (feature detector size)
```

Max Pooling Layer:

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

Flatten layer:

```
[ ]  model.add(Flatten())
```

Dense (hidden layer):

```
Number of types of flowers:  5
Types of flowers:  ['daisy', 'rose', 'tulip', 'dandelion', 'sunflower']
```

Out[2]:

	category	image
0	daisy	flowersData/daisy/14167534527_781ceb1b7a_n.jpg
1	daisy	flowersData/daisy/34718882165_68cdc9def9_n.jpg
2	daisy	flowersData/daisy/5512287917_9f5d3f0f98_n.jpg
3	daisy	flowersData/daisy/476857510_d2b30175de_n.jpg
4	daisy	flowersData/daisy/521762040_f26f2e08dd.jpg

```
[ ]
model.add(Dence(units=3,kernel_intializer="random_uniform"),activation="relu")
model.add(Dence(units=2,kernel_intializer="random_uniform"),activation="relu")
```

Output layer:

```
[ ] model.add(Dence(units=5,kernel_intializer="random_uniform"),activation="softmax")
```

4. Compile the Model

```
[ ] model.compile(optimizer=Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
dandelion : 26.07 %
tulip : 21.12 %
sunflower : 20.52 %
rose : 16.19 %
daisy : 16.09 %
```

5. Fit the Model

```
history = model.fit_generator(datagen.flow(X_train,y_train, batch_size=batch_size),
                              epochs = epochs,
                              validation_data = (X_test,y_test),
                              verbose = 1)
```

```
max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
2D)

conv2d_2 (Conv2D)            (None, 37, 37, 96)       55392

max_pooling2d_2 (MaxPooling  (None, 18, 18, 96)       0
2D)

conv2d_3 (Conv2D)            (None, 18, 18, 96)       83040

max_pooling2d_3 (MaxPooling  (None, 9, 9, 96)         0
2D)

flatten (Flatten)            (None, 7776)             0

dense (Dense)                (None, 512)              3981824

activation (Activation)      (None, 512)              0

dense_1 (Dense)              (None, 5)                2565

=================================================================
Total params: 4,143,749
Trainable params: 4,143,749
Non-trainable params: 0
_________________________________________________________________
```

6. Save the Model

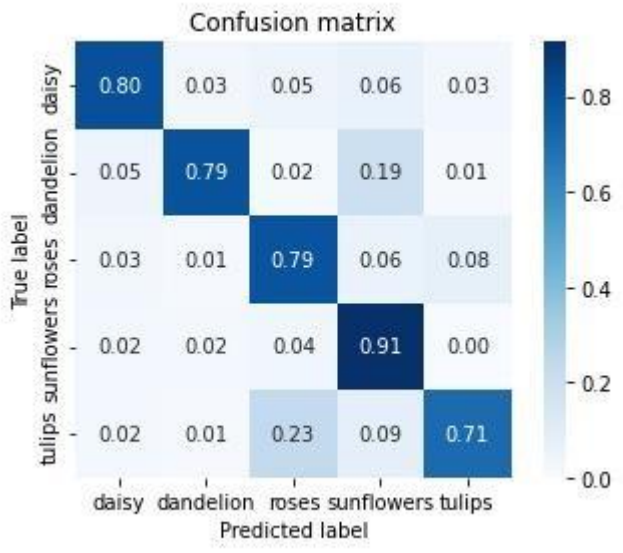
```
model.save("flowers.h5")
```

7. Test the Model Prediction Test:

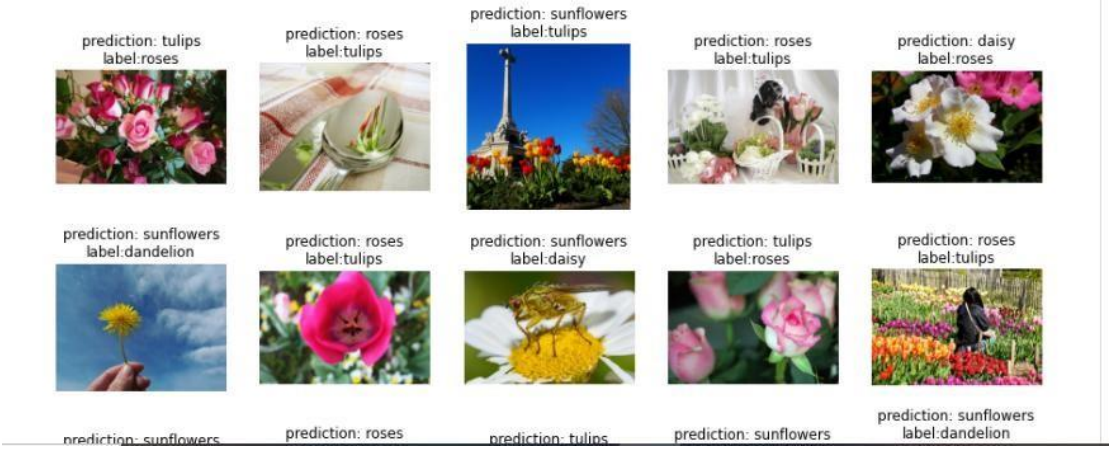
```
def show_confusion_matrix(test_labels, predictions):
  """Compute confusion matrix and normalize."""
  confusion = sk_metrics.confusion_matrix(
    np.argmax(test_labels, axis=1), predictions)
  confusion_normalized = confusion.astype("float") / confusion.sum(axis=1)
  axis_labels = list(CLASSES.values())
  ax = sns.heatmap(
      confusion_normalized, xticklabels=axis_labels, yticklabels=axis_labels,
      cmap='Blues', annot=True, fmt='.2f', square=True)
  plt.title("Confusion matrix")
  plt.ylabel("True label")
  plt.xlabel("Predicted label")

show_confusion_matrix(batch_labels, test_prediction)
```

Incorrect Prediction test:



```
incorrect = [
    (example, CLASSES[prediction])
    for example, prediction, is_correct in zip(test_batch, test_prediction, correct_predicate)
    if not is_correct
]
display_images(
  [(get_image(example), "prediction: {0}\nlabel:{1}".format(incorrect_prediction, get_class(example
   for (example, incorrect_prediction) in incorrect[:20]])
```



Count test:

In [4]:
```
# Let's do some visualization and see how many samples we have for each category
f, axe = plt.subplots(1,1,figsize=(14,6))
sns.barplot(x = flowerNum.index, y = flowerNum.values, ax = axe, palette="rocket")
axe.set_title("Flowers count for each category", fontsize=16)
axe.set_xlabel('Category', fontsize=14)
axe.set_ylabel('Count', fontsize=14)
plt.show()
```