```
import warnings
warnings.filterwarnings("ignore")
```

Assignment - 3, authored by Sharan M V

1. Download the dataset here.

Importing necessary Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout,Conv2D,Flatten,MaxPool2D,Reshape
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img,img_to_array
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

Data Augumentation

- Dataset consist of 5 classes.
- **Daisy** - European Species of Aster family.
- **Sunflower** - Identified as the genus of Helianthus.
- **Tulip** - Belong to the species of spring blooming geophytes.
- **Rose** - Belongs to the family of rosaceae.
- **Dandelion** - Indentifies as the genus of Asterceae.

```
path = 'flowers/'
```

```
train_data_gen = ImageDataGenerator(rescale = 1./255,
                             shear_range = 0.2,
                             zoom_range = 0.2,
                             horizontal_flip = True,
                             validation_split = 0.30)
test_data_gen = ImageDataGenerator(rescale = 1./255,validation_split = 0.30)
```

```
training_set = train_data_gen.flow_from_directory(path,
                                                  target_size=(64,64),
                                                  batch_size=100,
                                                  class_mode='categorical',
                                                  shuffle=True,
                                                  color_mode='rgb',
                                                  subset = 'training')

testing_set = test_data_gen.flow_from_directory(path,
                                                  target_size=(64,64),
                                                  batch_size=100,
                                                  class_mode='categorical',
                                                  shuffle=True,
                                                  color_mode='rgb',
                                                  subset = 'validation')
```

```
Found 3024 images belonging to 5 classes.
Found 1293 images belonging to 5 classes.
```

Model building using CNN

1. Create the model

```
model = Sequential()

#convolution and Pooling layer 1
model.add(Conv2D(filters=48,kernel_size=3,activation='relu',input_shape=(64,64,3)))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Dropout(0.2))

#convolution and Pooling layer 2
model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Dropout(0.2))

#Flattening the images
model.add(Flatten())

#Fully Connected layers
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(5,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 48)        1344
 max_pooling2d (MaxPooling2D  (None, 31, 31, 48)       0
 )

 dropout (Dropout)           (None, 31, 31, 48)        0

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        13856

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

 dropout_1 (Dropout)         (None, 14, 14, 32)        0

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 64)                401472

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 5)                 325

=================================================================
Total params: 416,997
Trainable params: 416,997
Non-trainable params: 0
_________________________________________________________________
```

2. Compile the Model

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

3. Adding callbacks to avoid overfitting

```
early_stop = EarlyStopping(monitor='val_accuracy',
                           patience=5,verbose=1,mode='auto')

lr = ReduceLROnPlateau(monitor='val_accuracy',
                       factor=0.2,patience=5,
                       min_lr=0.00001)

callback = [early_stop,lr]
```

4. Training the Model

```
result = model.fit(x=training_set, validation_data=testing_set, epochs=50)
```

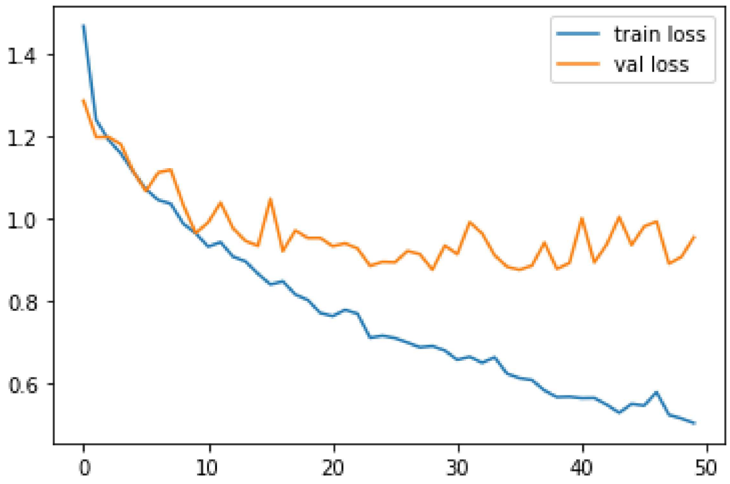
```
Epoch 1/50
31/31 [==============================] - 17s 536ms/step - loss: 1.4674 - accuracy: 0.
Epoch 2/50
```

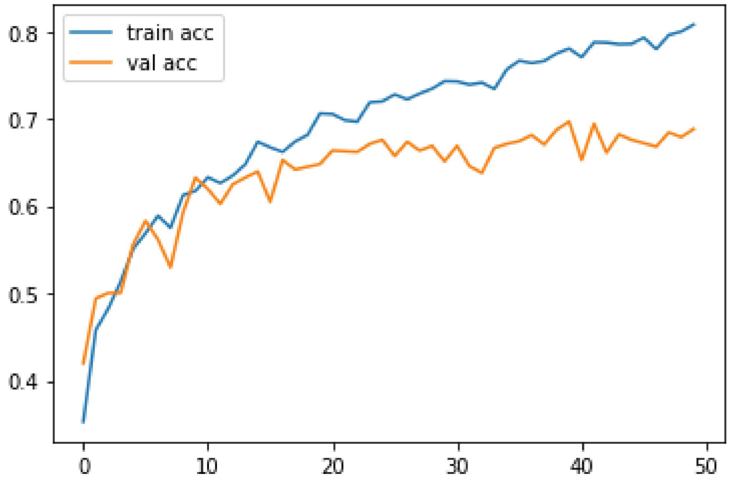
```
31/31 [==============================] - 18s 598ms/step - loss: 1.2396 - accuracy: 0.4
Epoch 3/50
31/31 [==============================] - 17s 544ms/step - loss: 1.1910 - accuracy: 0.4
Epoch 4/50
31/31 [==============================] - 20s 628ms/step - loss: 1.1573 - accuracy: 0.
Epoch 5/50
31/31 [==============================] - 17s 556ms/step - loss: 1.1126 - accuracy: 0.
Epoch 6/50
31/31 [==============================] - 19s 626ms/step - loss: 1.0707 - accuracy: 0.
Epoch 7/50
31/31 [==============================] - 20s 655ms/step - loss: 1.0441 - accuracy: 0.
Epoch 8/50
31/31 [==============================] - 44s 1s/step - loss: 1.0352 - accuracy: 0.575
Epoch 9/50
31/31 [==============================] - 19s 603ms/step - loss: 0.9866 - accuracy: 0.
Epoch 10/50
31/31 [==============================] - 19s 599ms/step - loss: 0.9633 - accuracy: 0.
Epoch 11/50
31/31 [==============================] - 19s 616ms/step - loss: 0.9307 - accuracy: 0.
Epoch 12/50
31/31 [==============================] - 18s 594ms/step - loss: 0.9415 - accuracy: 0.
Epoch 13/50
31/31 [==============================] - 18s 596ms/step - loss: 0.9065 - accuracy: 0.
Epoch 14/50
31/31 [==============================] - 19s 606ms/step - loss: 0.8950 - accuracy: 0.
Epoch 15/50
31/31 [==============================] - 19s 621ms/step - loss: 0.8649 - accuracy: 0.
Epoch 16/50
31/31 [==============================] - 19s 593ms/step - loss: 0.8388 - accuracy: 0.
Epoch 17/50
31/31 [==============================] - 19s 631ms/step - loss: 0.8464 - accuracy: 0.
Epoch 18/50
31/31 [==============================] - 19s 609ms/step - loss: 0.8144 - accuracy: 0.
Epoch 19/50
31/31 [==============================] - 19s 628ms/step - loss: 0.8010 - accuracy: 0.
Epoch 20/50
31/31 [==============================] - 20s 656ms/step - loss: 0.7695 - accuracy: 0.
Epoch 21/50
31/31 [==============================] - 21s 675ms/step - loss: 0.7618 - accuracy: 0.
Epoch 22/50
31/31 [==============================] - 18s 595ms/step - loss: 0.7771 - accuracy: 0.
Epoch 23/50
31/31 [==============================] - 19s 624ms/step - loss: 0.7676 - accuracy: 0.
Epoch 24/50
31/31 [==============================] - 19s 615ms/step - loss: 0.7089 - accuracy: 0.
Epoch 25/50
31/31 [==============================] - 20s 635ms/step - loss: 0.7140 - accuracy: 0.
Epoch 26/50
31/31 [==============================] - 20s 655ms/step - loss: 0.7082 - accuracy: 0.
Epoch 27/50
31/31 [==============================] - 20s 640ms/step - loss: 0.6974 - accuracy: 0.
Epoch 28/50
31/31 [==============================] - 17s 554ms/step - loss: 0.6858 - accuracy: 0.
Epoch 29/50
```

5. Loss and Accuracy check using plot

```
#plot the loss
plt.plot(result.history['loss'], label='train loss')
plt.plot(result.history['val_loss'], label='val loss')
plt.legend()
plt.show()

# plot the accuracy
plt.plot(result.history['accuracy'], label='train acc')
plt.plot(result.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
```





6. Save the Model

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

Testing the Model

```
training_set.class_indices
```

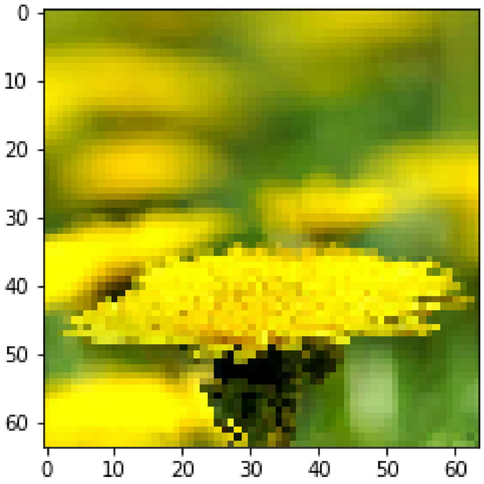
```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

```
classes = ['Daisy','Dandelion','Rose','Sunflower','Tulip']
def testing(img):
    img = image.load_img(img,target_size=(64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x,axis=0)
    pred = np.argmax(model.predict(x))
    return print("Predicted class as:",classes[pred])

def img_show(img):
    img1 = image.load_img(img,target_size=(64,64))
    plt.imshow(img1)
```

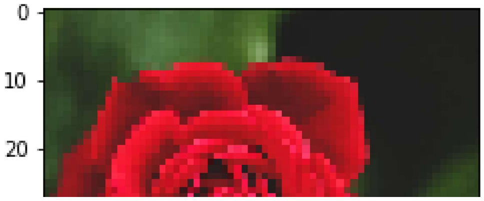
```
#test1
img_show('flower1.jpg')
testing('flower1.jpg')
```

```
Predicted class as: Dandelion
```



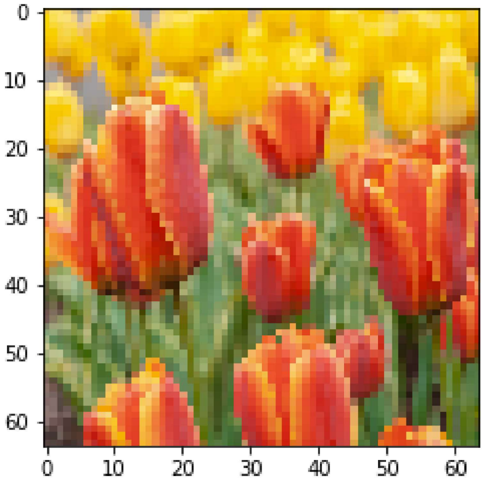
```
#test2
img_show('flower2.jpg')
testing('flower2.jpg')
```

```
Predicted class as: Rose
```



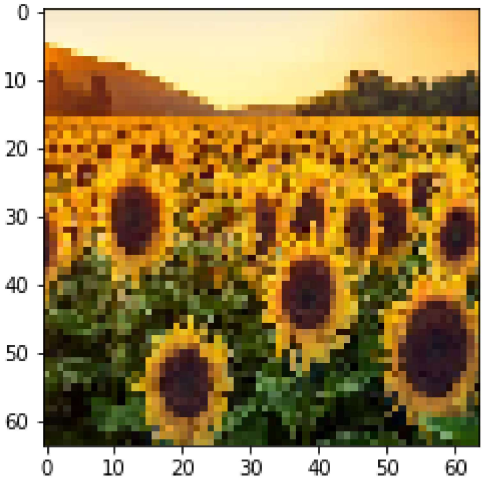
```
#test3
img_show('flower3.jpg')
testing('flower3.jpg')
```

```
Predicted class as: Tulip
```



```
#test4
img_show('flower4.jpg')
testing('flower4.jpg')
```

```
Predicted class as: Sunflower
```



```
#test5
img_show('flower5.jpg')
testing('flower5.jpg')
```

```
Predicted class as: Daisy
```

Conclusion:

- The dataset has about 4317 images from 5 different classes.
- Each classes have more than 500 images for training the data.
- 30% of the data taken for validation.
- The accuracy of the model is around 80%.
- The validation accuracy is around 70%.
- The model is built with 2 layered convolutional network considering 1344 trainable parameters.
- Testing the model with unknown images gives 95% accuracy.