Assignment-3, authored by Guruprasath G

1. Download the dataset here (https://drive.google.com/file/d/1xkynpL15pt6KT3YSIDimu4A5iRU9qYck/view).

Importing necessary Libraries

In [1]:

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

In [2]:

```
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,MaxPo
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_
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.

In [3]:

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

In [4]:

```
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)
```

In [5]:

```
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

In [6]:

```
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'))
```

In [7]:
```
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

In [8]:
```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy
```

3. Adding callbacks to avoid overfitting

In [9]:
```
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

```
In [10]: result = model.fit(x=training_set, validation_data=testing_set, epochs=50)
```

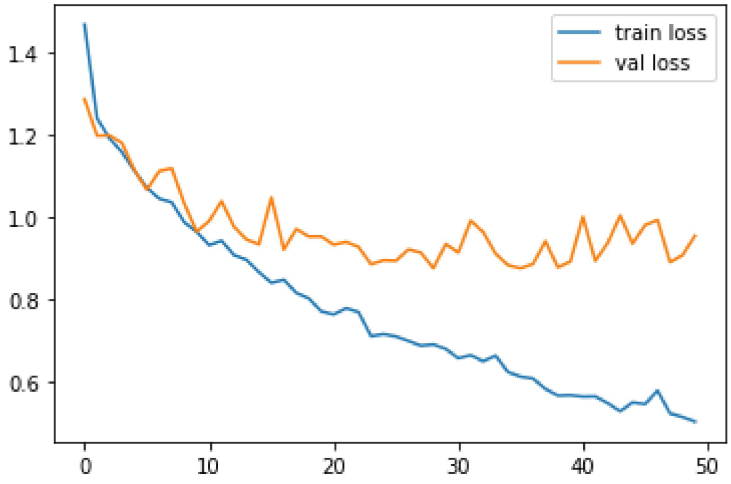
```
racy: 0.7854 - val_loss: 1.0029 - val_accuracy: 0.6821
Epoch 45/50
31/31 [==============================] - 20s 627ms/step - loss: 0.5482 - accu
racy: 0.7857 - val_loss: 0.9348 - val_accuracy: 0.6759
Epoch 46/50
31/31 [==============================] - 19s 605ms/step - loss: 0.5440 - accu
racy: 0.7930 - val_loss: 0.9796 - val_accuracy: 0.6721
Epoch 47/50
31/31 [==============================] - 20s 640ms/step - loss: 0.5768 - accu
racy: 0.7798 - val_loss: 0.9920 - val_accuracy: 0.6682
Epoch 48/50
31/31 [==============================] - 20s 625ms/step - loss: 0.5214 - accu
racy: 0.7960 - val_loss: 0.8894 - val_accuracy: 0.6845
Epoch 49/50
31/31 [==============================] - 18s 561ms/step - loss: 0.5127 - accu
racy: 0.7999 - val_loss: 0.9061 - val_accuracy: 0.6790
Epoch 50/50
31/31 [==============================] - 20s 658ms/step - loss: 0.5015 - accu
racy: 0.8079 - val_loss: 0.9536 - val_accuracy: 0.6883
```

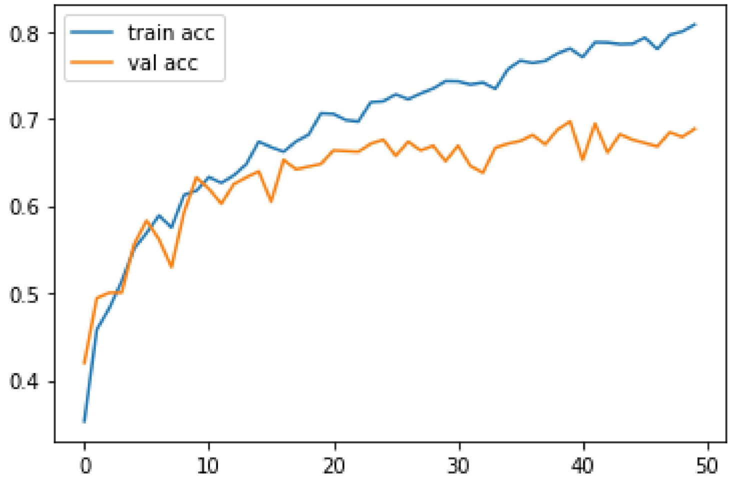
5. Loss and Accuracy check using plot

In [11]:

```
#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

In [12]:

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

Testing the Model

In [13]:

```
training_set.class_indices
```

Out[13]:

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

In [26]:

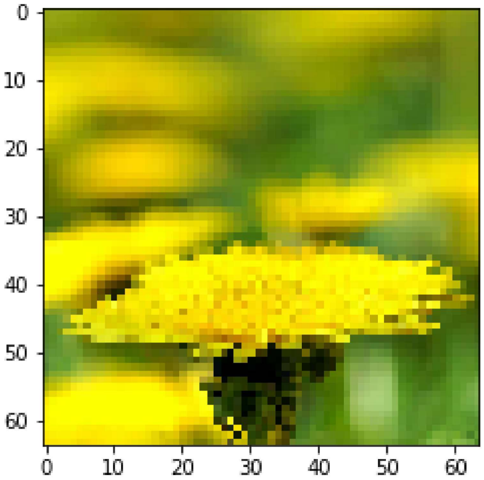
```
classes = ['Daisy','Dandelion','Rose','Sunflower','Tulip']
def testing(img):
    img = image.load_img(img,target_size=(64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x,axis=0)
    pred = np.argmax(model.predict(x))
    return print("Predicted class as:",classes[pred])

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

In [35]:

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

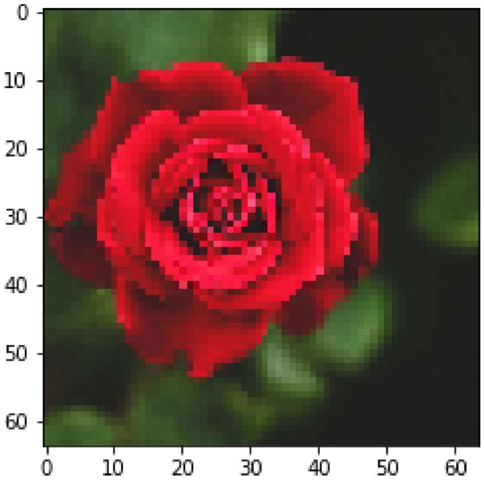
```
Predicted class as: Dandelion
```



In [31]:

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

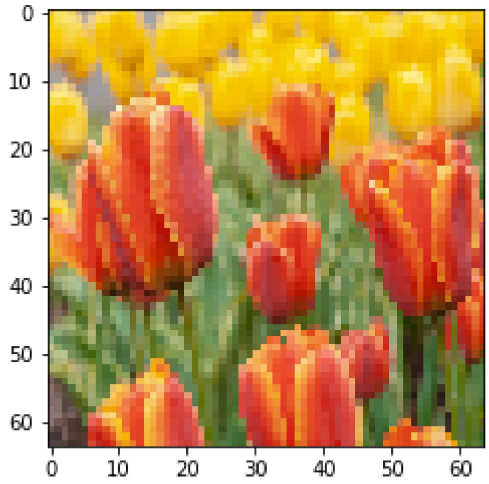
```
Predicted class as: Rose
```



In [30]:

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

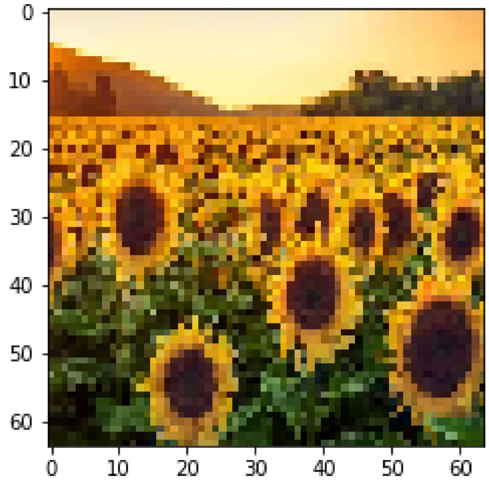
```
Predicted class as: Tulip
```



In [32]:

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

```
Predicted class as: Sunflower
```



In [33]:

```
#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.