**Flower categorisation**

## Executive Summary

Abstract: The main purpose of this project is for flower image classification using transfer learning. The study of flower classification is an important aspect due to its complex environment for extracting features, similarity of features in many species, changing features in the life of a flower having etc. An accurate classification is considered as a novel work since many classification algorithms are still newly presented.

## Goal

Implementation of a deep learning convolution neural network to classify 5 main flower categories.

We do this by achieving the following objectives:

1. Accurately detect flower images given the accuracy achieved below.

## Result comparison

| Researchers | Year | Algorithms | Dataset | Accuracy |
|---|---|---|---|---|
| DiahHarnoni et.al | 2013 | MSRM | 300 | 79.33% |
| Fadzilah et.al | 2014 | Neural Network | 180 | |
| TanakornTiay et.al | 2014 | Hu's seven and K-Nearest Neighbor Algorithm | 32 | 80% |
| Shubra Aich | 2015 | Manifold Mapping | Oxford 17 | 40% |
| Yuanyuan Liu | 2016 | Conventional Neural Network | Oxford 102 | 84.02% |
| S Krishnaveni et.al | 2017 | SVM | 500 | 83% |
| S Krishnaveni et.al | 2017 | RFT | 500 | 57% |
| Xiaolng Xia et.al | 2017 | Inception V3 | Oxford 17 | 95% |
| Xiaolng Xia et.al | 2017 | Inception V3 | Oxford 102 | 94% |
| Wei liu et.al | 2017 | Fusion Descriptor and SVM | Oxford 17 | 86.17% |

**Source:** International Journal of Electrical Electronics & Computer Science Engineering (Special Issue - NCCT-2018 | E-ISSN: 2348-2273 | P-ISSN: 2454-1222 February, 2018.)

***Data source:**
https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz

**Approach**

Transfer Learning Mobile Net (70% accuracy)

```
[14] # TODO: Build and train your network.
     URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

     feature_extractor = hub.KerasLayer(URL, input_shape=(image_size, image_size,3))

[15] feature_extractor.trainable = False

[16] layer_neurons = [650, 330, 250]

     dropout_rate = 0.2

     model = tf.keras.Sequential()

     model.add(feature_extractor)

     #for neurons in layer_neurons:
     #    model.add(tf.keras.layers.Dense(neurons, activation = 'relu'))
     #    model.add(tf.keras.layers.Dropout(dropout_rate))

     model.add(tf.keras.layers.Dense(102, activation = 'softmax'))

     model.summary()
```

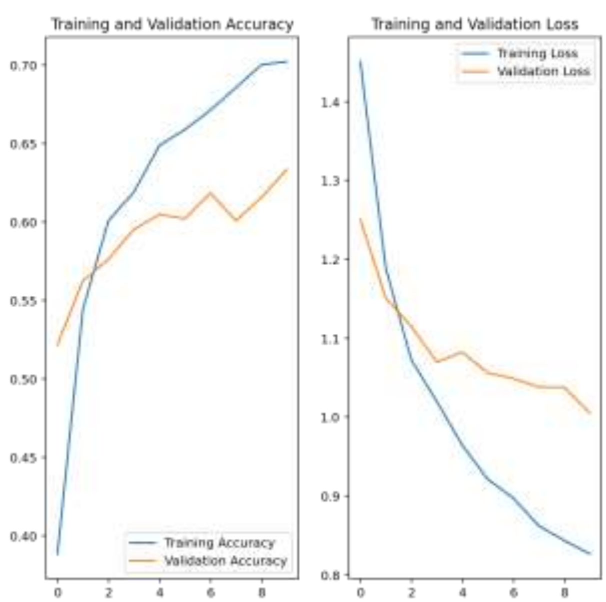
VGG16 (70% accuracy)

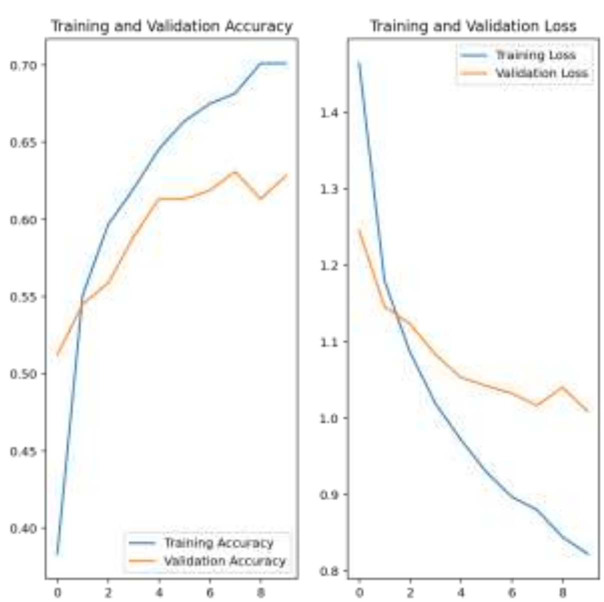
```python
from tensorflow.keras.applications.vgg16 import VGG16

base_model = VGG16(input_shape = (224, 224, 3), # Shape of our images
include_top = False, # Leave out the last fully connected layer
weights = 'imagenet')
```

```python
for layer in base_model.layers:
    layer.trainable = False
```

```python
layer_neurons = [650, 330, 250]

dropout_rate = 0.2

model = tf.keras.Sequential()

model.add(feature_extractor)

#for neurons in layer_neurons:
#    model.add(tf.keras.layers.Dense(neurons, activation = 'relu'))
#    model.add(tf.keras.layers.Dropout(dropout_rate))

model.add(tf.keras.layers.Dense(102, activation = 'softmax'))

model.summary()
```

### Performance and Results




The first graph are the results from MobileNet while the second graph are results from VGG16

**Project**: Deep Learning TensorFlow (2022.01. opencampus)

**Done by**: John Kimani