Reference

https://www.youtube.com/watch?v=AwOIgOwaLI0 (https://www.youtube.com/watch?v=AwOIgOwaLI0)

Imports

In [1]:

```
import tensorflow_datasets as tfds
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
```

Load Data

Explore more about datset: https://www.tensorflow.org/datasets/catalog/tf_flowers (https://www.tensorflow.org/datasets/catalog/tf_flowers)

In [2]:

```
## Loading images and labels
(train_ds, train_labels), (test_ds, test_labels) = tfds.load("tf_flowers",
    split=["train[:70%]", "train[:30%]"], ## Train test split
    batch_size=-1,
    as_supervised=True,  # Include labels
)
```

```
Downloading and preparing dataset Unknown size (download: Unknown size, gene
rated: Unknown size, total: Unknown size) to C:\Users\Shubham Dhamal\tensorf
low_datasets\tf_flowers\3.0.1...
```

```
Dl Completed...: 100%       1/1 [00:49<00:00, 49.19s/ url]
```

```
Dl Size...: 100%       218/218 [00:49<00:00, 5.76 MiB/s]
```

```
Dataset tf_flowers downloaded and prepared to C:\Users\Shubham Dhamal\tensor
flow_datasets\tf_flowers\3.0.1. Subsequent calls will reuse this data.
```

Image Preprocessing

In [3]:

```
## check existing image size
train_ds[0].shape
```

Out[3]:

```
TensorShape([442, 1024, 3])
```

In [4]:

```
## Resizing images
train_ds = tf.image.resize(train_ds, (150, 150))
test_ds = tf.image.resize(test_ds, (150, 150))
```

In [5]:

```
train_labels
```

Out[5]:

```
<tf.Tensor: shape=(2569,), dtype=int64, numpy=array([2, 3, 3, ..., 0, 2, 0],
dtype=int64)>
```

In [6]:

```
## Transforming labels to correct format
train_labels = to_categorical(train_labels, num_classes=5)
test_labels = to_categorical(test_labels, num_classes=5)
```

In [7]:

```
train_labels[0]
```

Out[7]:

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

Use Pretrained VGG16 Image Classification model

Load a pre-trained CNN model trained on a large dataset

In [8]:

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

In [9]:

```
train_ds[0].shape
```

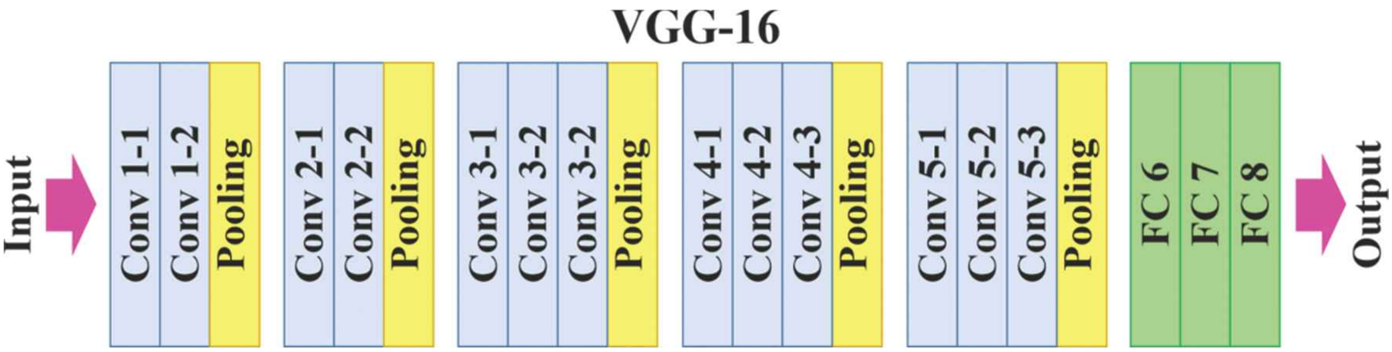
Out[9]:

```
TensorShape([150, 150, 3])
```

In [10]:

```
## Loading VGG16 model
base_model = VGG16(weights="imagenet", include_top=False, input_shape=train_ds[0].shape
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applic
ations/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5 (https://stor
age.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_
ordering_tf_kernels_notop.h5)
58889256/58889256 [==============================] - 14s 0us/step
```



In [11]:

```
## will not train base mode
# Freeze Parameters in model's lower convolutional layers
base_model.trainable = False
```

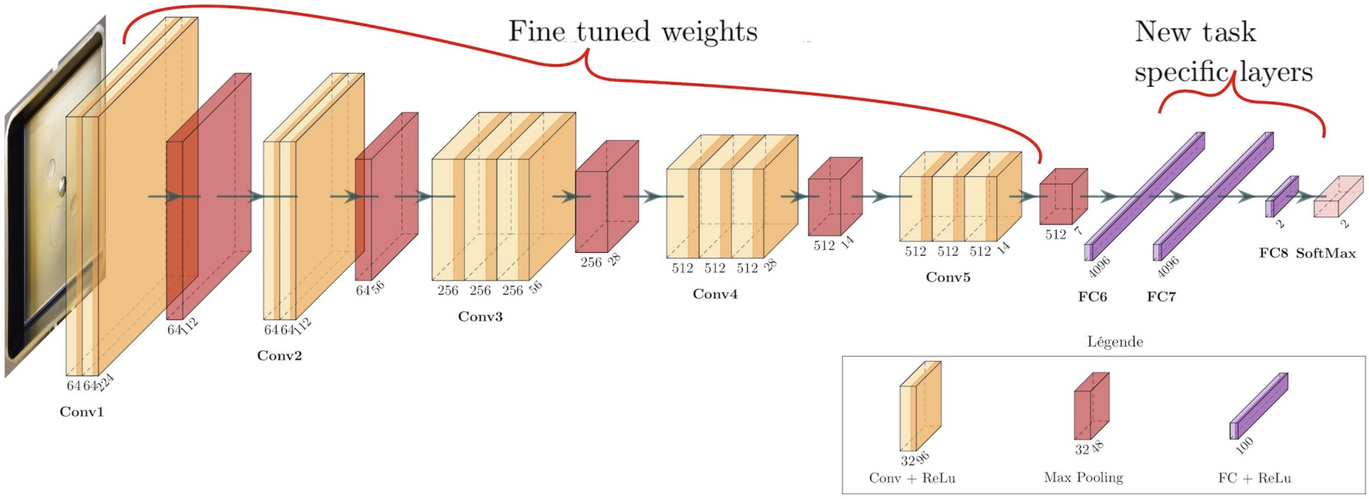
In [12]:

```
## Preprocessing input
train_ds = preprocess_input(train_ds)
test_ds = preprocess_input(test_ds)
```

In [13]:

```
## model details
base_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 150, 150, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 150, 150, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 150, 150, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 75, 75, 64)        0         
                                                                 
 block2_conv1 (Conv2D)       (None, 75, 75, 128)       73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 75, 75, 128)       147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 37, 37, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 37, 37, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 37, 37, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 37, 37, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 18, 18, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 18, 18, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 18, 18, 512)       2359808   
                                                                 
 block4_conv3 (Conv2D)       (None, 18, 18, 512)       2359808   
                                                                 
 block4_pool (MaxPooling2D)  (None, 9, 9, 512)         0         
                                                                 
 block5_conv1 (Conv2D)       (None, 9, 9, 512)         2359808   
                                                                 
 block5_conv2 (Conv2D)       (None, 9, 9, 512)         2359808   
                                                                 
 block5_conv3 (Conv2D)       (None, 9, 9, 512)         2359808   
                                                                 
 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0         
                                                                 
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```



Add custom classifier with two dense layers of trainable parameters to model

In [14]:

```
#add our layers on top of this model
from tensorflow.keras import layers, models

flatten_layer = layers.Flatten()
dense_layer_1 = layers.Dense(50, activation='relu')
dense_layer_2 = layers.Dense(20, activation='relu')
prediction_layer = layers.Dense(5, activation='softmax')


model = models.Sequential([
    base_model,
    flatten_layer,
    dense_layer_1,
    dense_layer_2,
    prediction_layer
])
```

Train classifier layers on training data available for task

In [15]:

```
from tensorflow.keras.callbacks import EarlyStopping

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

In [16]:

```
es = EarlyStopping(monitor='val_accuracy', mode='max', patience=5,  restore_best_weight
```

In [19]:

```
history=model.fit(train_ds, train_labels, epochs=1, validation_split=0.2, batch_size=32
```

```
65/65 [==============================] - 383s 6s/step - loss: 1.3275 - accur
acy: 0.4642 - val_loss: 1.1292 - val_accuracy: 0.5720
```

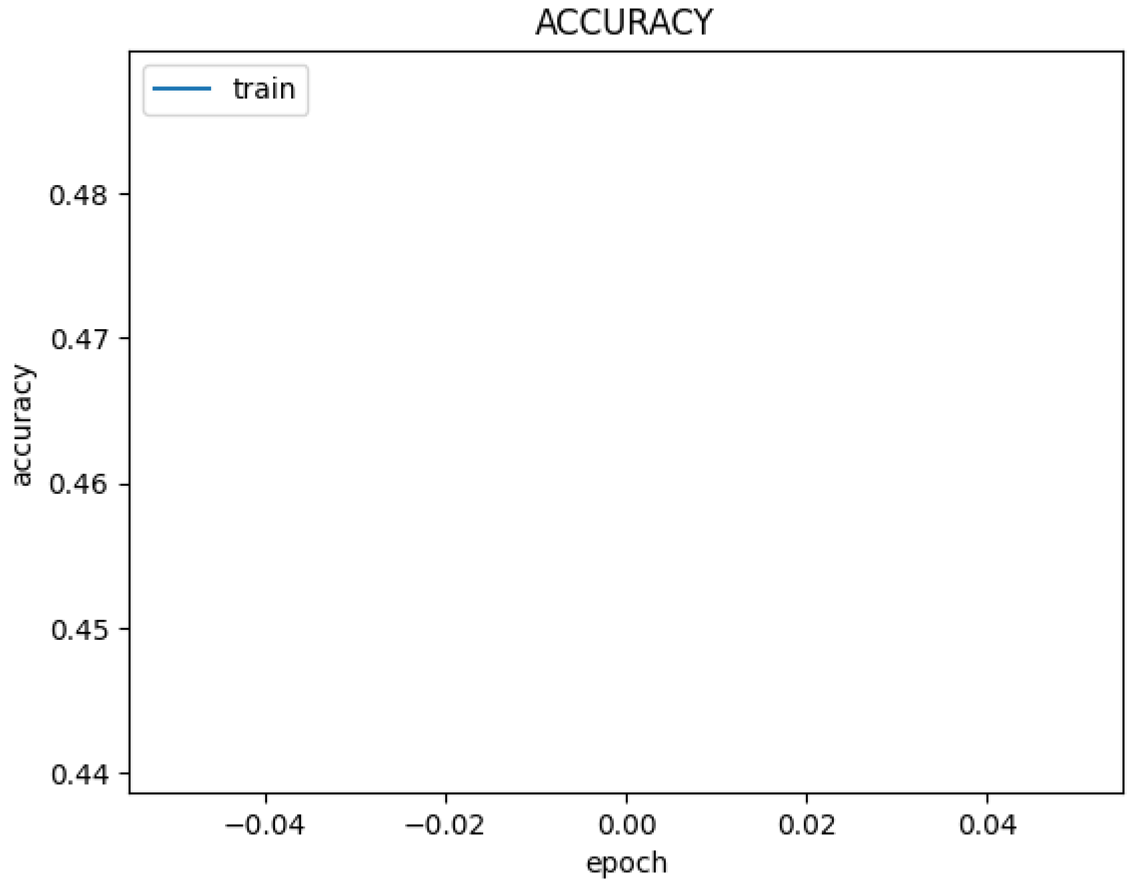
In [20]:

```
los,accurac=model.evaluate(test_ds,test_labels)
print("Loss: ",los,"Accuracy: ", accurac)
```

```
35/35 [==============================] - 182s 5s/step - loss: 0.9499 - accur
acy: 0.6222
Loss:  0.9498578310012817 Accuracy:  0.6221616864204407
```

In [21]:

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.title('ACCURACY')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train'],loc='upper left')
plt.show()
```



In [22]:

```python
import numpy as np
import pandas as pd
y_pred = model.predict(test_ds)
y_classes = [np.argmax(element) for element in y_pred]
#to_categorical(y_classes, num_classes=5)
#to_categorical(test_labels, num_classes=5)
print(y_classes[:10])
print("\nTest")
print(test_labels[:10])
```

```
35/35 [==============================] - 181s 5s/step
[4, 3, 3, 2, 3, 2, 0, 0, 0, 2]

Test
[[0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]]
```

In [ ]:

```

```