In [0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
call drive.mount("/content/drive", force_remount=True).
```

Type *Markdown* and LaTeX: $\alpha^2$

# Colab uses GPU

In [3]:
```
#' ' means CPU whereas '/device:G:0' means GPU
import tensorflow as tf
tf.test.gpu_device_name()
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

Out[3]:
```
'/device:GPU:0'
```

# Testing GPU utilization

```
1
```

In [4]:
```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
print(GPUs)
#XXX: only one GPU on Colab and isn't guaranteed
gpu = GPUs[0]
def printm():
 process = psutil.Process(os.getpid())
 print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory()
 print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | To
printm()
```

```
Collecting gputil
  Downloading https://files.pythonhosted.org/packages/ed/0e/5c61eedde9f6c
87713e89d794f01e378cfd9565847d4576fa627d758c554/GPUtil-1.4.0.tar.gz (http
s://files.pythonhosted.org/packages/ed/0e/5c61eedde9f6c87713e89d794f01e37
8cfd9565847d4576fa627d758c554/GPUtil-1.4.0.tar.gz)
Building wheels for collected packages: gputil
  Building wheel for gputil (setup.py) ... done
  Created wheel for gputil: filename=GPUtil-1.4.0-cp36-none-any.whl size=
7413 sha256=c177abcedcb0977a7b4efb71616bf761a3bad716abb85b2327a42fcc6894a
064
  Stored in directory: /root/.cache/pip/wheels/3d/77/07/80562de4bb0786e5e
a186911a2c831fdd0018bda69beab71fd
Successfully built gputil
Installing collected packages: gputil
Successfully installed gputil-1.4.0
Requirement already satisfied: psutil in /usr/local/lib/python3.6/dist-pa
ckages (5.4.8)
Requirement already satisfied: humanize in /usr/local/lib/python3.6/dist-
packages (0.5.1)
[<GPUtil.GPUtil.GPU object at 0x7f94c0723358>]
Gen RAM Free: 26.2 GB  | Proc size: 457.5 MB
GPU RAM Free: 11372MB | Used: 69MB | Util   1% | Total 11441MB
```

# Import Necessary Libraries

In [57]:
```
import os
import zipfile
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
import matplotlib
import matplotlib.pyplot as plt

from keras_preprocessing.image import ImageDataGenerator,load_img
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Activation, BatchNorm
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
print(tf.__version__)

```

```
1.15.0
```

# Unzip the Dataset

In [0]:
```
zip_ref = zipfile.ZipFile('drive/My Drive/Datasets/Flowers.zip', 'r')
zip_ref.extractall('Flower') # unzip directory
zip_ref.close()
```

# Load Data and Create Data Frame

In [5]:
```
flower_class = pd.read_csv('Flower/classlabels.txt', header = None, nam
flower_class
```

Out[5]:

| | Images | Class |
|---|---|---|
| **0** | JFT_00001.jpg | 1 |
| **1** | JFT_00002.jpg | 1 |
| **2** | JFT_00003.jpg | 1 |
| **3** | JFT_00004.jpg | 1 |
| **4** | JFT_00006.jpg | 1 |
| **...** | ... | ... |
| **1474** | JFT_01467.jpg | 29 |
| **1475** | JFT_01470.jpg | 30 |
| **1476** | JFT_01472.jpg | 30 |
| **1477** | JFT_01476.jpg | 30 |
| **1478** | JFT_01477.jpg | 30 |

1479 rows × 2 columns

In [6]:
```
flower_class['N_Images']= flower_class['Images'].str.strip('JFT_0000').
flower_class
```

Out[6]:

| | Images | Class | N_Images |
|---|---|---|---|
| **0** | JFT_00001.jpg | 1 | 1 |
| **1** | JFT_00002.jpg | 1 | 2 |
| **2** | JFT_00003.jpg | 1 | 3 |
| **3** | JFT_00004.jpg | 1 | 4 |
| **4** | JFT_00006.jpg | 1 | 6 |
| **...** | ... | ... | ... |
| **1474** | JFT_01467.jpg | 29 | 1467 |
| **1475** | JFT_01470.jpg | 30 | 1470 |
| **1476** | JFT_01472.jpg | 30 | 1472 |
| **1477** | JFT_01476.jpg | 30 | 1476 |
| **1478** | JFT_01477.jpg | 30 | 1477 |

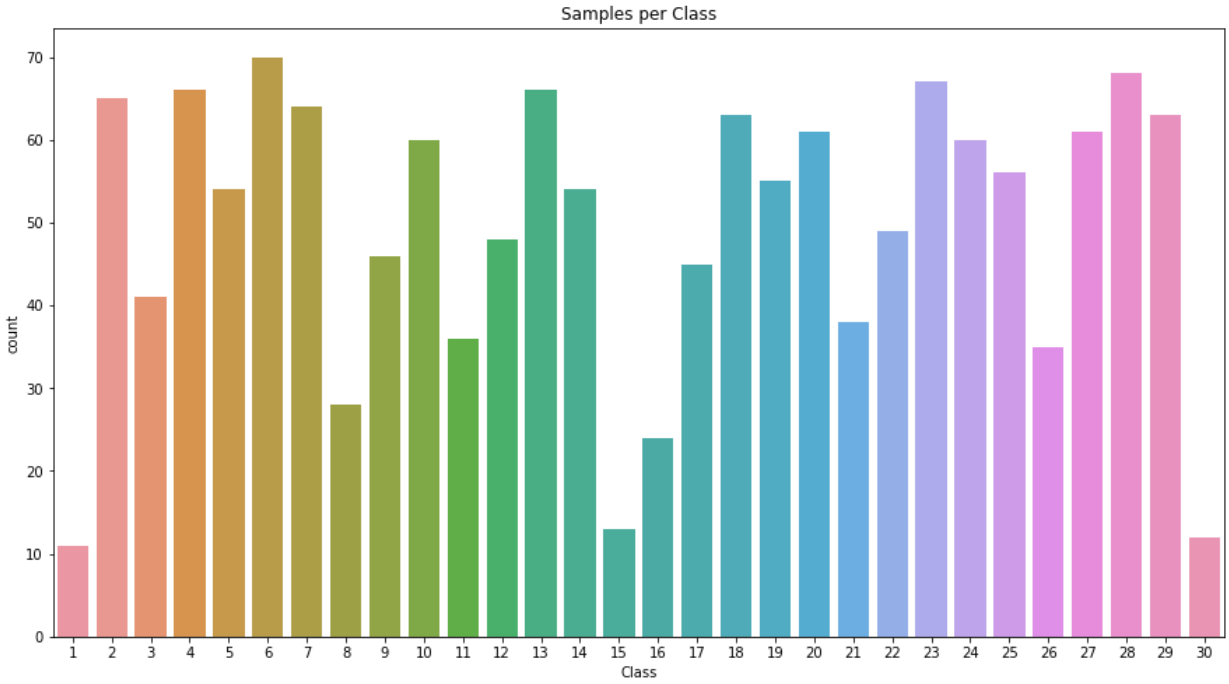
1479 rows × 3 columns

# Calculate the Baseline

In [7]:
```
base_line = 100*flower_class['Class'].value_counts()/len(flower_class)
base_line
```

Out[7]:
```
6     4.732928
28    4.597701
23    4.530088
13    4.462475
4     4.462475
2     4.394861
7     4.327248
18    4.259635
29    4.259635
20    4.124408
27    4.124408
24    4.056795
10    4.056795
25    3.786342
19    3.718729
14    3.651116
5     3.651116
22    3.313049
12    3.245436
9     3.110210
17    3.042596
3     2.772143
21    2.569304
11    2.434077
26    2.366464
8     1.893171
16    1.622718
15    0.878972
30    0.811359
1     0.743746
Name: Class, dtype: float64
```

In [12]:
```
plt.figure(figsize=(15, 8))
sns.countplot(x='Class',data=flower_class)
plt.title("Samples per Class")
plt.show()
```




In [13]:
```
train1 = pd.read_csv('Flower/tst1.txt',header= None,names=['N_Images'])
train1['N_Images'] = train1['N_Images'].apply(str)
train1df = flower_class.merge(train1, on= 'N_Images')
train1df['Class'] = train1df['Class'].apply(str)
train1df


```

Out[13]:

| | Images | Class | N_Images |
|---|---|---|---|
| **0** | JFT_00005.jpg | 1 | 5 |
| **1** | JFT_00007.jpg | 1 | 7 |
| **2** | JFT_00011.jpg | 1 | 11 |
| **3** | JFT_00012.jpg | 2 | 12 |
| **4** | JFT_00013.jpg | 2 | 13 |
| **...** | ... | ... | ... |
| **916** | JFT_01467.jpg | 29 | 1467 |
| **917** | JFT_01470.jpg | 30 | 1470 |
| **918** | JFT_01472.jpg | 30 | 1472 |
| **919** | JFT_01476.jpg | 30 | 1476 |
| **920** | JFT_01477.jpg | 30 | 1477 |

921 rows × 3 columns

In [14]:
```
val1 = pd.read_csv('Flower/val1.txt',sep = ' ',header= None,names=['N_I
val1['N_Images'] = val1['N_Images'].apply(str)
val1df = flower_class.merge(val1, on= 'N_Images')
val1df['Class'] = val1df['Class'].apply(str)
val1df
```

Out[14]:

| | Images | Class | N_Images |
|---|---|---|---|
| **0** | JFT_00001.jpg | 1 | 1 |
| **1** | JFT_00003.jpg | 1 | 3 |
| **2** | JFT_00006.jpg | 1 | 6 |
| **3** | JFT_00008.jpg | 1 | 8 |
| **4** | JFT_00018.jpg | 2 | 18 |
| **...** | ... | ... | ... |
| **274** | JFT_01462.jpg | 29 | 1462 |
| **275** | JFT_01469.jpg | 30 | 1469 |
| **276** | JFT_01471.jpg | 30 | 1471 |
| **277** | JFT_01474.jpg | 30 | 1474 |
| **278** | JFT_01479.jpg | 30 | 1479 |

279 rows × 3 columns

In [15]:
```
test1 = pd.read_csv('Flower/trn1.txt',sep = ' ',header= None,names=['N_
test1['N_Images'] = test1['N_Images'].apply(str)
test1df = flower_class.merge(test1, on= 'N_Images')
test1df['Class'] = test1df['Class'].apply(str)
test1df
```

Out[15]:

| | Images | Class | N_Images |
|---|---|---|---|
| **0** | JFT_00002.jpg | 1 | 2 |
| **1** | JFT_00004.jpg | 1 | 4 |
| **2** | JFT_00009.jpg | 1 | 9 |
| **3** | JFT_00010.jpg | 1 | 10 |
| **4** | JFT_00021.jpg | 2 | 21 |
| **...** | ... | ... | ... |
| **274** | JFT_01466.jpg | 29 | 1466 |
| **275** | JFT_01468.jpg | 30 | 1468 |
| **276** | JFT_01473.jpg | 30 | 1473 |
| **277** | JFT_01475.jpg | 30 | 1475 |
| **278** | JFT_01478.jpg | 30 | 1478 |

279 rows × 3 columns

# About the datasets

In [16]:
```
print("Total Number of Images",flower_class.shape[0])
print("Number of Training Images", train1df.shape[0])
print("Number of Validation Images", val1df.shape[0])
print("Number of Training Images", test1df.shape[0])
```

```
Total Number of Images 1479
Number of Training Images 921
Number of Validation Images 279
Number of Training Images 279
```

# Data Augmentation

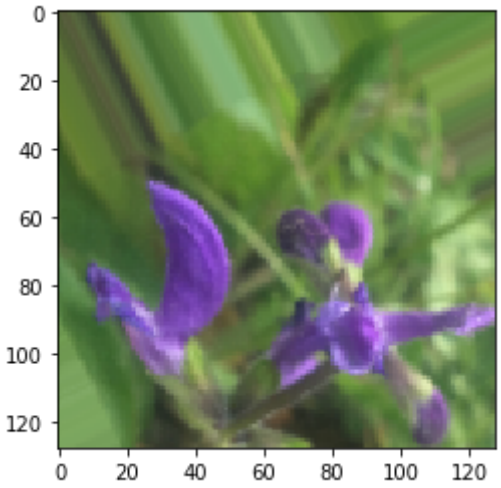
In [17]:

```
datagen=ImageDataGenerator(rescale=1./255.,
                           featurewise_center=False,
                           featurewise_std_normalization=False,
                           rotation_range=90,
                           width_shift_range=0.2,
                           height_shift_range=0.2,
                           horizontal_flip=True,
                           vertical_flip=True)

train1_generator=datagen.flow_from_dataframe(
dataframe=train1df,
directory="Flower/jpg/",
x_col="Images",
y_col="Class",
subset="training",
batch_size=128,
seed=42,
shuffle=True,
class_mode="sparse",
color_mode="rgb",
#target_size=(256,256))
target_size=(128,128))
print('Train generator created')


val1_generator=datagen.flow_from_dataframe(
dataframe=val1df,
directory="Flower/jpg/",
x_col="Images",
y_col="Class",
subset="training",
batch_size=138,
seed=42,
shuffle=True,
class_mode="sparse",
#target_size=(256,256))
target_size=(128,128))
print('Validation generator created')

test1_generator=datagen.flow_from_dataframe(
dataframe=test1df,
directory="Flower/jpg/",
x_col="Images",
y_col="Class",
subset="training",
#batch_size=128,
batch_size= 1,
color_mode="rgb",
seed=42,
shuffle=False,
#class_mode="sparse",
class_mode = None,
#target_size=(256,256))
target_size=(128,128))
print('Test generator created')
```

```
Found 921 validated image filenames belonging to 30 classes.
```

```
Train generator created
Found 279 validated image filenames belonging to 30 classes.
Validation generator created
Found 279 validated image filenames.
Test generator created
```

In [18]:
```
image, label = next(iter(train1_generator))
plt.imshow(image[0,:]);
```




In [19]:
```
# plt.figure(figsize=(10,10))
# for i in range(10):
#     plt.subplot(5,5,i+1)
#     plt.xticks([])
#     plt.yticks([])
#     plt.grid(False)
#     plt.imshow(train_generator[i])
train1_generator.image_shape
```

Out[19]: (128, 128, 3)

### Create And Train the Model

In [20]:
```
model = Sequential()
#model.add(Conv2D(32, (3, 3), padding='same', input_shape=(256,256,3)))
model.add(Conv2D(32, (3, 3), padding='same', input_shape=(128,128,3)))
#model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(32, (3, 3)))
#model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
#model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(64, (3, 3)))
#model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
#model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
#output Classes are 30
model.add(Dense(30))
model.add(Activation('softmax'))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/back
end/tensorflow_backend.py:66: The name tf.get_default_graph is deprecate
d. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/back
end/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Ple
ase use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/back
end/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated.
Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/back
end/tensorflow_backend.py:4267: The name tf.nn.max_pool is deprecated. Pl
ease use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/back
end/tensorflow_backend.py:148: The name tf.placeholder_with_default is de
precated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/back
end/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.o
ps.nn_ops) with keep_prob is deprecated and will be removed in a future v
ersion.
Instructions for updating:
```

```
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [21]:
```
1 model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 126, 126, 32)      9248
_________________________________________________________________
activation_2 (Activation)    (None, 126, 126, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 63, 63, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 63, 63, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 61, 61, 64)        36928
_________________________________________________________________
activation_4 (Activation)    (None, 61, 61, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 57600)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               29491712
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 30)                15390
_________________________________________________________________
activation_6 (Activation)    (None, 30)                0
=================================================================
Total params: 29,572,670
Trainable params: 29,572,670
Non-trainable params: 0
_________________________________________________________________
```

In [22]:
```
model.compile(optimizer = tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

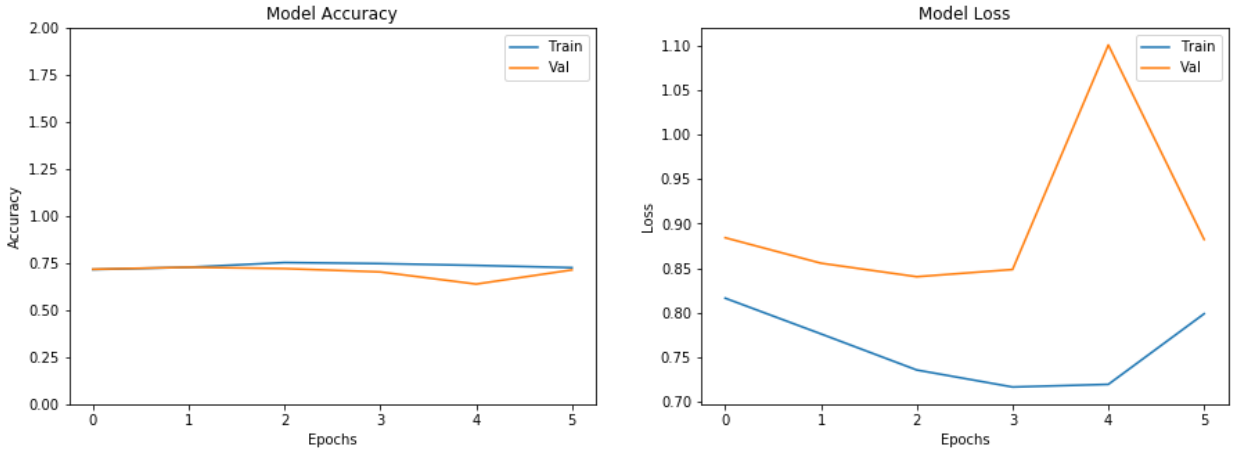
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3622: The name tf.log is deprecated. Please use tf.math.log instead.
```

### Fitting the Model

In [33]:
```
from keras.callbacks import EarlyStopping

early_stopping_monitor = EarlyStopping(patience=3)
EPOCHS = 500

# STEP_SIZE_TRAIN=train1_generator.n//train1_generator.batch_size
# STEP_SIZE_VALID=val1_generator.n//val1_generator.batch_size
# STEP_SIZE_TEST=test1_generator.n//test1_generator.batch_size

history = model.fit(train1_generator,
                    validation_data = val1_generator,
                    epochs=EPOCHS, callbacks=[early_stopping_monitor])
```

```
Epoch 1/500
8/8 [==============================] - 19s 2s/step - loss: 0.8202 - acc: 0.7088 - val_loss: 0.8842 - val_acc: 0.7168
Epoch 2/500
8/8 [==============================] - 19s 2s/step - loss: 0.7804 - acc: 0.7161 - val_loss: 0.8556 - val_acc: 0.7276
Epoch 3/500
8/8 [==============================] - 19s 2s/step - loss: 0.7301 - acc: 0.7567 - val_loss: 0.8404 - val_acc: 0.7204
Epoch 4/500
8/8 [==============================] - 19s 2s/step - loss: 0.7286 - acc: 0.7446 - val_loss: 0.8485 - val_acc: 0.7025
Epoch 5/500
8/8 [==============================] - 19s 2s/step - loss: 0.7095 - acc: 0.7357 - val_loss: 1.1007 - val_acc: 0.6380
Epoch 6/500
8/8 [==============================] - 19s 2s/step - loss: 0.7913 - acc: 0.7248 - val_loss: 0.8823 - val_acc: 0.7133
```

In [39]:
```
def plot_history(history):
  plt.figure(figsize=(15, 5))
  plt.subplot(1, 2, 1)
  plt.xlabel('Epochs')
  plt.ylabel('Accuracy')
  plt.plot(history.epoch, np.array(history.history['acc']),
           label='Train')
  plt.plot(history.epoch, np.array(history.history['val_acc']),
           label = 'Val')
  plt.title('Model Accuracy')
  plt.legend()
  plt.ylim([0, 2])

  plt.subplot(1, 2, 2)
  plt.plot(history.history['loss'],label='Train')
  plt.plot(history.history['val_loss'],label='Val')
  plt.title('Model Loss')
  plt.ylabel('Loss')
  plt.xlabel('Epochs')
  plt.legend()
  plt.show()

plot_history(history)


```




In [40]:
```
model.evaluate_generator(generator=val1_generator)
```

Out[40]:
```
[0.8677034198596913, 0.7096774280071259]
```

**Predict Output**

In [0]:
```
# Get the filenames from the generator
fnames = test1_generator.filenames

# Get the ground truth from generator
ground_truth = val1_generator.classes

# Get the label to class mapping from the generator
label2index = val1_generator.class_indices

# Getting the mapping from class index to class label
idx2label = dict((v,k) for k,v in label2index.items())

```

In [0]:
```
# Get the predictions from the model using the generator
predictions = model.predict_generator(val1_generator,verbose=1)
predicted_classes = np.argmax(predictions,axis=1)
```

In [67]:
```
errors = np.where(predicted_classes != ground_truth)[0]
print("No of errors = {}/{}".format(len(errors),val1_generator.samples))
```

```
No of errors = 265/279
```

In [81]:

```
# Show the errors
for i in range(len(errors)):
    pred_class = np.argmax(predictions[errors[i]])
    pred_label = idx2label[pred_class]

    title = 'Original label:{}, Prediction :{}, confidence : {:.3f}'.fo
        fnames[errors[i]].split('/')[0],
        pred_label,
        predictions[errors[i]][pred_class])

    original = load_img('{}/{}'.format(val1_generator.directory,fnames[
    plt.figure(figsize=[7,7])
    plt.axis('off')
    plt.title(title)
    plt.imshow(original)
    plt.show()

```

Original label:JFT_00002.jpg, Prediction :2, confidence : 0.977