Load Dataset by Botnoi Image Scaping

```
!pip install botnoi==0.2.1
from botnoi import scrape as sc
from botnoi import cv
import os
```

```
Collecting botnoi==0.2.1
  Downloading botnoi-0.2.1-py3-none-any.whl (16.0 MB)
     |████████████████████████████████| 16.0 MB 148 kB/s
Installing collected packages: botnoi
Successfully installed botnoi-0.2.1
```

Test Image Scraping

```
[ ] sc.get_image_urls('bird',100)
```

```
'https://s1.manualzz.com/store/data/006566233_1-d1105d070e9b1aa769ffd0410cf9ea20-595x842.png',
'https://cdn.britannica.com/91/173391-131-E2106C95.jpg',
'https://image.shutterstock.com/image-illustration/man-doing-bird-dog-exercise-260nw-1592796499.jpg',
'https://orafitnessandyoga.com/wp-content/uploads/2018/08/blog_birddog.jpg',
'https://www.bird-dog.tv/wp-content/uploads/2019/04/MINI-19-def-1.png',
'https://cdn.cnn.com/cnnnext/dam/assets/190919131522-01-north-american-bird-loss.jpg',
'https://parade.com/wp-content/uploads/2020/06/iStock-481057316.jpg',
'https://geographical.co.uk/media/k2/items/cache/ba357ac7afbad42ac8b25041c150b8d1_XL.jpg',
'https://news.usc.edu/files/2019/11/Taiwan-Blue-Magpie-web-824x549.jpg',
'https://www.nsf.gov/news/mmg/media/images/birdsticks_lyme_f.jpg',
'https://www.picclickimg.com/d/l400/pict/224623895345_/Perfect-Vision-Birdog-Bird-Dog-Ultra-Satellite-Signal.jpg',
'https://hubats.com/wp-content/uploads/2019/02/Ni-MH-SC-3300mAh-7.2v-rechargeable-battery-pack-for-Birdog.jpg',
'https://nas-national-prod.s3.amazonaws.com/aud_gbbc-2016_american-robin_32533_kk_ca_photo-donald_metzner.jpg',
'https://www.summitsource.com/Assets/ProductImages/BDGULT.jpg',
'https://www.americansatellite.us/pub/media/catalog/product/b/i/birdog-ultra-large_1.jpg',
'https://static01.nyt.com/images/2020/04/03/science/03TB-KINGFISHER1/03TB-KINGFISHER1-mediumSquareAt3X.jpg',
'https://www.mindlttv.net/blog/wp-content/uploads/2019/09/BirdDog-Cloud.jpg',
'https://storage.googleapis.com/ares-profile-pictures/hd/birdog_official-218d7db0cbeaa4297cc7524a43cb4a45_hd.jpg',
'https://live.staticflickr.com/2802/4503310485_3be50047dc_b.jpg',
'https://whyy.org/wp-content/uploads/2021/03/BIRD_TALK_1200_800-web.jpg',
'https://www.birdsandblooms.com/wp-content/uploads/2021/07/252705471_1_Trisha_Snider_BNB_BYPC2020.jpg',
'https://images-na.ssl-images-amazon.com/images/I/71iND2IIwnL.jpg',
'https://nationaltoday.com/wp-content/uploads/2020/01/Bird-Day-1-640x514.jpg',
'https://www.birdwatchersdigest.com/bwdsite/wp-content/uploads/2020/03/prairie-warbler-male.jpg',
'https://www.allaboutbirds.org/news/wp-content/uploads/2009/04/WKingbird-James.jpg',
'https://brooklyneagle.com/wp-content/uploads/2021/08/10-1G2A7104-gray-catbird-hwolf-1024x768.jpg',
'https://www.imperial.ac.uk/ImageCropToolT4/imageTool/uploaded-images/newseventsimage_1529346275459_mainnews2012_x1.jpg',
'https://i.ytimg.com/vi/-LRjkbEy-qU/maxresdefault.jpg',
'https://cdn.birdwatchingdaily.com/2020/02/Tufted_Titmouse_62126_Jenny_Burdette_GA2019_Overall1-e1573593877167.jpeg',
'https://pbs.twimg.com/profile_images/1443808321820143640/ta1aizW7.jpg',
'https://cdn.birdwatchingdaily.com/2020/06/Mountain-Bluebird-nr-Millarville-29-May-2020.jpg',
'https://blog.solidsignal.com/wp-content/uploads/2012/03/DSCN0452.png',
'https://cdn.louisianamusicfactory.com/lmf/media/wp-content/uploads/2019/02/03041450/birdog-small-band.jpg',
'https://earthsky.org/upl/2021/02/Golden-crowned-sparrow-bird-San-Francisco-CA-Stephanie-Becker-Feb-1-2021-sq-e1612960448574.jpg',
'https://i.ytimg.com/vi/YFHsVcF9lgk/maxresdefault.jpg',
'https://i.natgeofe.com/n/6f9b6d9e-5797-4867-a859-7b0c2a66cd3b/02-bird-of-paradise-A012_C010_1029SF_0001575.jpg',
'https://www.batterydepot.com/media/catalog/product/cache/2/image/0ed4bf167095b2977f33d989a5e50f07/d/c/dc-61.jpg',
'https://content.propertyroom.com/listings/sellers/seller1/images/origimgs/birdog-perfect-vision-with-ddsi-satellite-meter-1_20420172146435982534.jpg',
'https://m.media-amazon.com/images/I/612zbep35UL._AC_SY355_.jpg',
'https://i2.wp.com/bdfunhouse.wordpress.com/files/2009/03/birdog.jpg',
'https://i.ebayimg.com/images/g/2TEAAOSwgQ9VilsH/s-l300.jpg',
'https://ichef.bbci.co.uk/news/976/cpsprodpb/67CF/production/_108857562_mediaitem108857561.jpg',
'https://i.imgur.com/1xWPqHY.jpg',
'https://newscdn2.weigelbroadcasting.com/6apN3-1624657042-199655-blog-northerncardinal_.jpg',
'https://abcbirds.org/wp-content/uploads/2021/07/Blue-Jay-on-redbud-tree-by-Tom-Reichner_news.png',
'https://i.pinimg.com/originals/df/1c/6d/df1c6dd089dcb9efc0aa30c6bc837440.png',
'https://angelelectronics.ca/shop/image/cache/data/products/birdog-usb-plus-v4__2-700x700.jpg',
'https://i.ytimg.com/vi/RIN8pKgKUN0/maxresdefault.jpg',
'https://9b16f79ca967fd0708d1-2713572fef44aa49ec323e813b06d2d9.ssl.cf2.rackcdn.com/1140x_a10-7_cTC/birdplague0701-1625605893.jpg',
'https://i.ytimg.com/vi/wiFNA3sqjCA/maxresdefault.jpg',
'https://www.bird-dog.tv/wp-content/uploads/2018/09/BirdDog_logo_NEW-GREEN.png',
'https://www.perkypet.com/media/Articles/Perky-Pet/What-to-Do-If-You-Find-An-Injured-Bird.jpg',
'https://www.picclickimg.com/d/l400/pict/233273193326_/Perfect-Vision-Birdog-Ultra-Satellite-Signal-Meter-And.jpg',
'https://climate.nasa.gov/system/news_items/main_images/3047_main_image.jpg',
'https://arc-anglerfish-washpost-prod-washpost.s3.amazonaws.com/public/R2V7YZM4KFDSNIYMFFMPDNO5GI.jpg',
'https://i1.sndcdn.com/avatars-000124133632-iveo0h-t500x500.jpg',
'https://ichef.bbci.co.uk/news/976/cpsprodpb/4FB7/production/_116970402_a20-20sahas20barve20-20parrotbill_chavan.jpg',
'https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/img-2335-jpg-1573672702.jpg',
'https://www.picclickimg.com/d/l400/pict/174222455816_/Used-Birdog-Ultra-Satellite-Signal-Meter-Finder-Bird.jpg',
'https://www.hakaimagazine.com/wp-content/uploads/facebook-gulf-birds.jpg']
```

Scape images to dataset directory

```
[2] import urllib.request
    import socket
    import cv2
    socket.setdefaulttimeout(5)

    def extractimagefeat(query,nimg=20):
      #create folder
      foldername = 'images/'+query
      isdir = os.path.isdir(foldername)
      #check if folder exist
      if not isdir:
        #create directory
        os.makedirs(foldername)
      #get images from google search
      imglist = sc.get_image_urls(query,nimg)
      i = 1
      for img in imglist[0:nimg]:
        try:
          #extract image features from each images and save to files
          #create image path
```

```
            filename, file_extension = os.path.splitext(img)
            savepath = foldername + '/' + str(i)+file_extension
            urllib.request.urlretrieve(img, savepath)
            if(cv2.imread(savepath)==None):
              os.remove(savepath)
          except:
            pass
          i = i + 1

        return 'complete'
```

```
[3] extractimagefeat('cat',nimg=200)
```

```
complete: 0.00%
complete: 24.50%
complete: 54.00%
completed
'complete'
```

```
[4] extractimagefeat('giraffe',nimg=200)
```

```
complete: 0.00%
complete: 29.00%
complete: 50.00%
complete: 61.50%
complete: 89.00%
completed
'complete'
```

```
[5] extractimagefeat('elephant',nimg=200)
```

```
complete: 0.00%
complete: 30.50%
complete: 62.50%
complete: 96.00%
completed
'complete'
```

check every image vaild

```
[6] def check_images( s_dir, ext_list):
        bad_images=[]
        bad_ext=[]
        s_list= os.listdir(s_dir)
        for klass in s_list:
            klass_path=os.path.join (s_dir, klass)
            print ('processing class directory ', klass)
            if os.path.isdir(klass_path):
                file_list=os.listdir(klass_path)
                for f in file_list:
                    f_path=os.path.join (klass_path,f)
                    index=f.rfind('.')
                    ext=f[index+1:].lower()
                    if ext not in ext_list:
                        print('file ', f_path, ' has an invalid extension ', ext)
                        bad_ext.append(f_path)
                    if os.path.isfile(f_path):
                        try:
                            img=cv2.imread(f_path)
                            shape=img.shape
                        except:
                            print('file ', f_path, ' is not a valid image file')
                            bad_images.append(f_path)
                    else:
                        print('*** fatal error, you a sub directory ', f, ' in class directory ', klass)
            else:
                print ('*** WARNING*** you have files in ', s_dir, ' it should only contain sub directories')
        return bad_images, bad_ext

    source_dir =r'/content/images'

    good_exts=['jpg', 'png', 'jpeg', 'gif', 'bmp' ] # list of acceptable extensions
    bad_file_list, bad_ext_list=check_images(source_dir, good_exts)
    if len(bad_file_list) !=0:
        print('improper image files are listed below')
        for i in range (len(bad_file_list)):
            print (bad_file_list[i])
    else:
        print(' no improper image files were found')
```

```
processing class directory  cat
processing class directory  giraffe
processing class directory  elephant
 no improper image files were found
```

```
[ ] #remove if some image not ok image
    !rm /content/images/bird/11.jpg
    !rm /content/images/bird/160.jpg
```

Import Python Other Lib

```
[7] import numpy as np
    import time

    import PIL.Image as Image
    import matplotlib.pylab as plt

    import tensorflow as tf
    import tensorflow_hub as hub

    import datetime
```

```
import datetime
%load_ext tensorboard
```

Batch Size

```
batch_size = 16
img_height = 224
img_width = 224
```

Setting Network

```
mobilenet_v2 ="https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4"

classifier_model = mobilenet_v2

IMAGE_SHAPE = (224, 224)

classifier = tf.keras.Sequential([
    hub.KerasLayer(classifier_model, input_shape=IMAGE_SHAPE+(3,))
])
```

Transfer learning

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  str('/content/images'),
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  str('/content/images'),
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size
)
```

```
Found 469 files belonging to 3 classes.
Using 376 files for training.
Found 469 files belonging to 3 classes.
Using 93 files for validation.
```

```
class_names = np.array(train_ds.class_names)
print(class_names)
print('All classes : ',len(class_names))
```

```
['cat' 'elephant' 'giraffe']
All classes :  3
```

```
# Data_augmentation
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip("horizontal_and_vertical"),
  tf.keras.layers.RandomRotation(0.4),
  tf.keras.layers.RandomContrast(0.3),
  tf.keras.layers.RandomZoom(0.4,0.4),
])
```

```
normalization_layer = tf.keras.layers.Rescaling(1./255)
train_ds = train_ds.map(lambda x, y: (normalization_layer(data_augmentation(x)), y)) # Where x—images, y—labels.
val_ds = val_ds.map(lambda x, y: (normalization_layer(data_augmentation(x)), y)) # Where x—images, y—labels.
```

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

```
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

```
(16, 224, 224, 3)
(16,)
```

```
mobilenet_v2 = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

feature_extractor_model = mobilenet_v2
```

```
    feature_extractor_model,
    input_shape=(224, 224, 3),
    trainable=False)
```

```
feature_batch = feature_extractor_layer(image_batch)
print(feature_batch.shape)
```

```
(16, 1280)
```

```
num_classes = len(class_names)

model = tf.keras.Sequential([
  feature_extractor_layer,
  tf.keras.layers.Dropout(0.2),        # add dropout
```

```
  tf.keras.layers.Dense(num_classes)
])

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_1 (KerasLayer)   (None, 1280)              2257984
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 3)                 3843
=================================================================
Total params: 2,261,827
Trainable params: 3,843
Non-trainable params: 2,257,984
_________________________________________________________________
```

```
[20] predictions = model(image_batch)
     predictions.shape
```

```
TensorShape([16, 3])
```

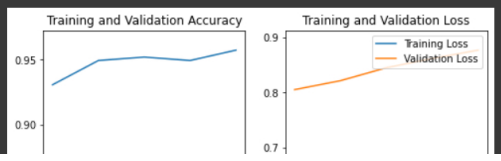
```
model.compile(
  optimizer=tf.keras.optimizers.Adam(),
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])

checkpoint_filepath = '/content/saved-model-epoch{epoch:03d}.pb' # pb
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=False,
    monitor='accuracy',
    mode='max',
    save_freq=16,
    save_best_only=False,)

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(
    log_dir=log_dir,
    histogram_freq=1) # Enable histogram computation for every epoch.
```

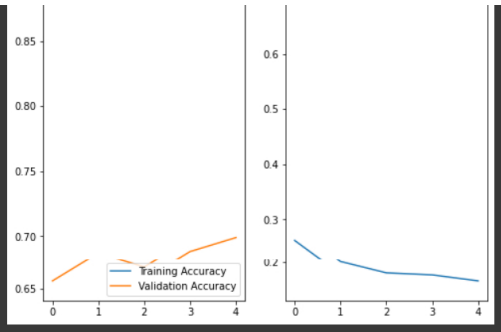
```
[38] NUM_EPOCHS = 5

     history = model.fit(train_ds,
                         validation_data=val_ds,
                         epochs=NUM_EPOCHS,
                         callbacks=[tensorboard_callback,model_checkpoint_callback])
```

```
Epoch 1/5
15/24 [=================>............] - ETA: 0s - loss: 0.2696 - accuracy: 0.9292INFO:tensorflow:Assets written to: /content/saved-model-epoch001.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch001.pb/assets
24/24 [==============================] - 11s 416ms/step - loss: 0.2619 - accuracy: 0.9309 - val_loss: 0.8053 - val_accuracy: 0.6559
Epoch 2/5
 7/24 [=======>......................] - ETA: 0s - loss: 0.1906 - accuracy: 0.9375INFO:tensorflow:Assets written to: /content/saved-model-epoch002.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch002.pb/assets
23/24 [===========================>..] - ETA: 0s - loss: 0.2029 - accuracy: 0.9484INFO:tensorflow:Assets written to: /content/saved-model-epoch002.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch002.pb/assets
24/24 [==============================] - 14s 607ms/step - loss: 0.2008 - accuracy: 0.9495 - val_loss: 0.8216 - val_accuracy: 0.6774
Epoch 3/5
15/24 [=================>............] - ETA: 0s - loss: 0.1652 - accuracy: 0.9625INFO:tensorflow:Assets written to: /content/saved-model-epoch003.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch003.pb/assets
24/24 [==============================] - 8s 345ms/step - loss: 0.1802 - accuracy: 0.9521 - val_loss: 0.8450 - val_accuracy: 0.6667
Epoch 4/5
 7/24 [=======>......................] - ETA: 0s - loss: 0.1616 - accuracy: 0.9643INFO:tensorflow:Assets written to: /content/saved-model-epoch004.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch004.pb/assets
23/24 [===========================>..] - ETA: 0s - loss: 0.1791 - accuracy: 0.9484INFO:tensorflow:Assets written to: /content/saved-model-epoch004.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch004.pb/assets
24/24 [==============================] - 15s 639ms/step - loss: 0.1765 - accuracy: 0.9495 - val_loss: 0.8617 - val_accuracy: 0.6882
Epoch 5/5
16/24 [===================>..........] - ETA: 0s - loss: 0.1632 - accuracy: 0.9492INFO:tensorflow:Assets written to: /content/saved-model-epoch005.pb/assets
INFO:tensorflow:Assets written to: /content/saved-model-epoch005.pb/assets
24/24 [==============================] - 8s 326ms/step - loss: 0.1656 - accuracy: 0.9574 - val_loss: 0.8769 - val_accuracy: 0.6989
```

Training Report

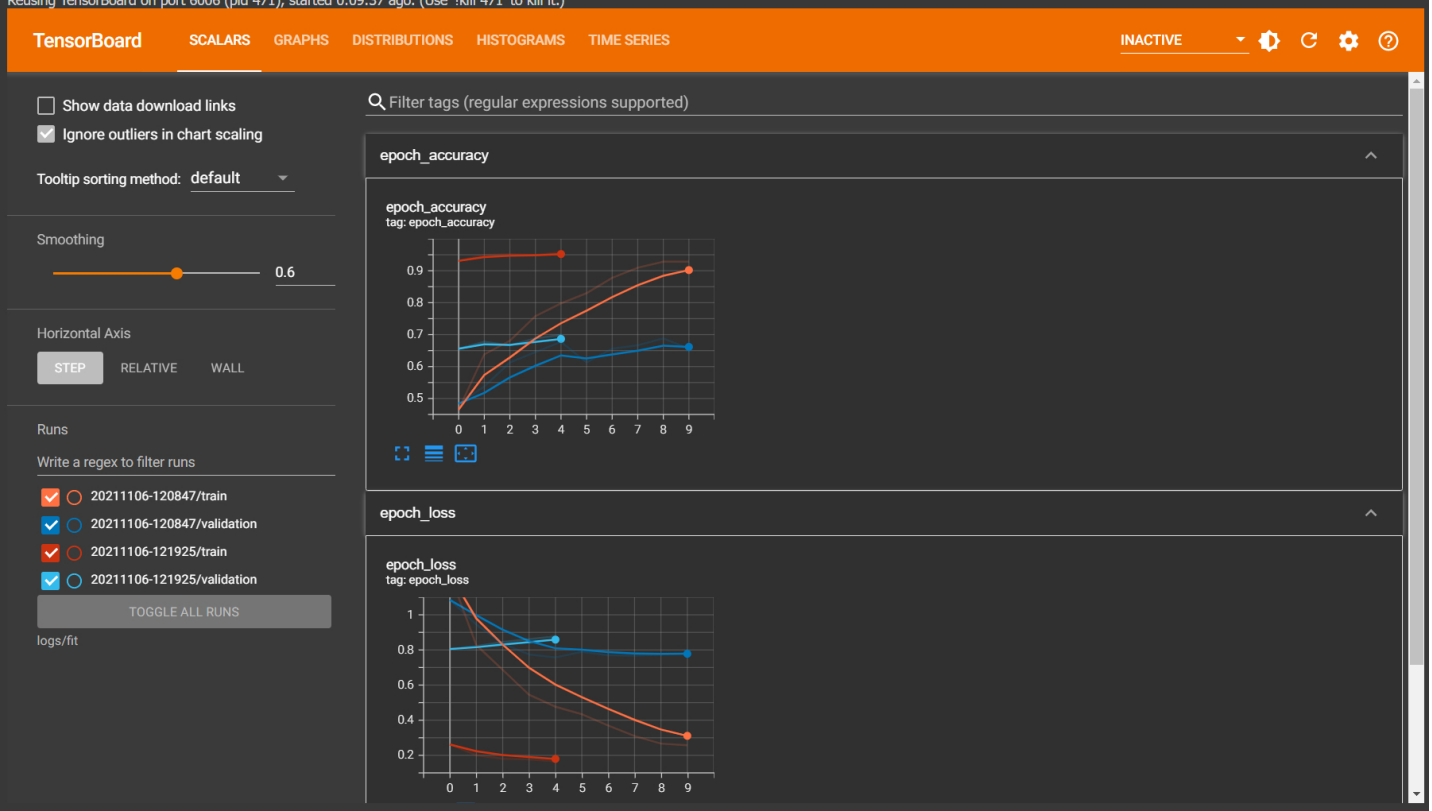
```
[39] acc = history.history['accuracy']
     val_acc = history.history['val_accuracy']

     loss = history.history['loss']
     val_loss = history.history['val_loss']

     epochs_range = range(NUM_EPOCHS)

     plt.figure(figsize=(8, 8))
     plt.subplot(1, 2, 1)
     plt.plot(epochs_range, acc, label='Training Accuracy')
     plt.plot(epochs_range, val_acc, label='Validation Accuracy')
     plt.legend(loc='lower right')
     plt.title('Training and Validation Accuracy')

     plt.subplot(1, 2, 2)
     plt.plot(epochs_range, loss, label='Training Loss')
     plt.plot(epochs_range, val_loss, label='Validation Loss')
     plt.legend(loc='upper right')
     plt.title('Training and Validation Loss')
     plt.show()
```





[40] %tensorboard --logdir logs/fit

Reusing TensorBoard on port 6006 (pid 471), started 0:09:37 ago. (Use '!kill 471' to kill it.)



[41] print(tf.version.VERSION)

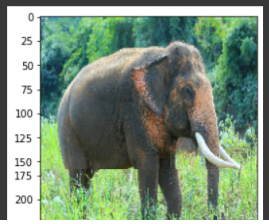
2.6.0

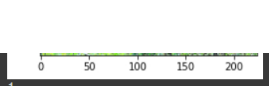
[43] reloaded_model = tf.keras.models.load_model('/content/saved-model-epoch005.pb')

[33]
```
def loadTestImageAndPredictFromPath(path):
    test_img1 = tf.keras.utils.load_img(path, target_size=(224, 224))
    plt.imshow(test_img1)
    plt.show()
    img_array = tf.keras.utils.img_to_array(test_img1).astype('float32')/255
    img_array = tf.expand_dims(img_array, 0) # Create a batch

    predictions = reloaded_model.predict(img_array)
    #print(predictions)
    score = tf.nn.softmax(predictions[0])
    print(np.argmax(score).item())
    print(
        "Predict -> {} with a {:.2f} percent confidence."
        .format(class_names[np.argmax(score).item()], 100 * np.max(score))
    )
```

[ ]
```
!wget https://reidparkzoo.org/wp-content/uploads/2018/03/American-Alligator-banner-900x350.jpg
!mv American-Alligator-banner-900x350.jpg Alligator01.jpg
```

[49] loadTestImageAndPredictFromPath('/content/test01.jpg')

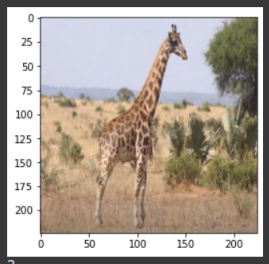




1
Predict -> elephant with a 94.99 percent confidence.

```
[ ] !wget https://see.news/wp-content/uploads/2020/12/UK_wildbirds-01-robin-750x375.jpg
    !mv /content/UK_wildbirds-01-robin-750x375.jpg /content/Bird01.jpeg
```

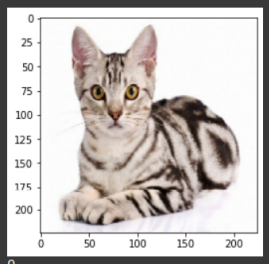
```
[50] loadTestImageAndPredictFromPath('/content/test02.jpg')
```



2
Predict -> giraffe with a 99.01 percent confidence.

```
[ ] !wget https://i.insider.com/5de5782179d75715a15a1fd3?width=1000&format=jpeg
    !mv /content/5de5782179d75715a15a1fd3?width=1000 /content/Dog01.jpeg
```

```
loadTestImageAndPredictFromPath('/content/test03.jpg')
```



0
Predict -> cat with a 71.97 percent confidence.

```
[ ] # อาจจะเพิ่ม Data Augmentation / เพิ่มรอบ / เพิ่มจำนวน Dataset
```

0 วินาที เสร็จสมบูรณ์เมื่อ 19:23