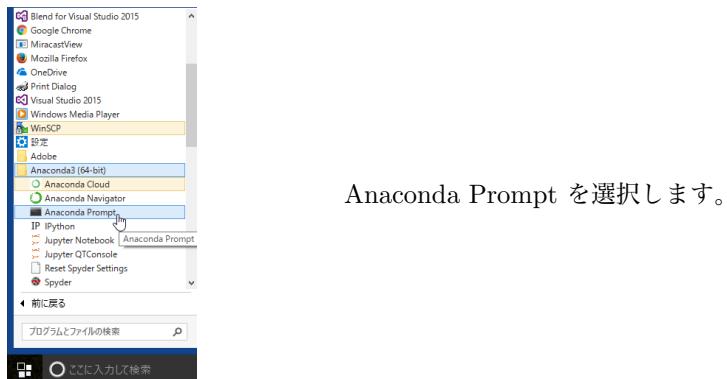


ラズベリー・パイでりんごと缶コーヒーの種類を見分ける

1 概要

前回は、訓練済みのモデルを利用しました。今回はりんごと缶コーヒーの種類を見分けるモデルを作成します。モデルの作成は各自のPC上で行います。作成したモデルをRaspberry Piへアップロードし、前回作成したプログラムを改良して、認識したものを発声を行います。

2 Anaconda Prompt の立ち上げ



Anaconda Prompt を選択します。

以前の授業で作成した tensorenv 環境に入ります。

```
activate tensorenv
```

以前の授業を欠席したために、tensorenv 環境を作成していないため、上記のコマンドが実行できない場合には、このテキストの「(参考)tensorenv 環境の作成」を参照して、環境を作成して下さい。

3 実行用のツールのダウンロード

実行用のツールをダウンロードします。そして、関連ファイルの入っているディレクトリに移動します。作業ディレクトリへ移動します。ホームディレクトリの下に「Lesson」というディレクトリがあることを前提に作業します。

```
cd %homepath%
cd Lesson
git clone https://github.com/k1nk/ch12.git
```

```
cd ch12
```

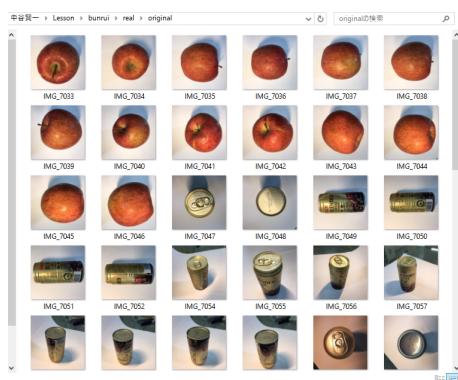
4 画像を集める

4.1 実際に写真をとる

認識させたいものの写真をとります。今回の例で撮った物体は、赤リンゴ缶コーヒー1「UCC BLEND 微糖」缶コーヒー2「GEORGIA エメラルドマウンテン」缶コーヒー3「SUNTORY BOSS 贅沢微糖」缶コーヒー4「WONDA モーニングショット」です。

今回は1つの物体について、物体の上から1枚、下から1枚、真横方向から方向を90度づつ回転して4枚、斜め上から方向を90度づつ回転して4枚、斜め下から方向を90度づつ回転して4枚の合計14枚を撮りました。

全部で $14 \times 5 = 70$ 枚となります。撮った画像（以下、サンプル画像）は、「images」フォルダの下の「apple」「boss」「GEORGIA」「UCC」「WONDA」というカテゴリ別のフォルダ内にあります。



サンプル画像とは別の画像を区分させたい場合には、「images」フォルダの名前を変更して、あたらに「images」フォルダを作成し、その中に同じようにカテゴリの名称を付けたフォルダを作成して、画像を区分して下さい。フォルダの数（カテゴリの数）は2つ以上であれば、いくつでも結構です。

5 画像を加工する

5.1 画像のサイズを調整する

画像を正方形に切り取り、 256×256 ピクセルにします。ここでは、crop_pictures3.pyを使ってクロップされたデータを作成します。crop_pictures3.pyは、指定したフォルダに下にあるファイルを、フォルダ下のフォルダ内のファイルも含めて、 256×256 ピクセルに切り取り、「指定したフォルダ名_cropped」というフォルダの中に保存します。画像ファイルは、「images」フォルダ内にあるので、指定してサイズの調整を行います。

```
python crop_pictures3.py images
```

images_cropped フォルダの中に、クロップされたデータが保存されます。

5.2 エフェクトをかけて画像を増やす

画像にエフェクトをかけて画像を増やします。また、元の画像およびエフェクトをかけた画像を左右反転させた画像を作成します。この例では、1つの画像に、8個のエフェクトをかけます。その結果、1つの画像が元の画像を含めて9枚の画像になります。その後、それらの画像を左右反転した画像を作成します。その結果元の画像を含めて18枚の画像が作成されます。

```
python increase_pictures3.py images_cropped
```

「images_cropped_trans」フォルダの中に、画像が保存されます。



図1 元画像



図2 ハイコントラスト



図3 ローコントラスト



図4 ガンマ変換 $\gamma < 1$



図5 ガンマ変換 $\gamma > 1$

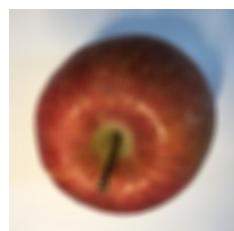


図6 平滑化



図7 ヒストグラム均一化



図8 ガウシアンノイズ



図9 ペッパーノイズ



図10 画像を左右反転

5.3 回転させて画像を増やす

画像を回転させて画像を増やします。元画像を、90度、180度、270度にさせた画像を作成します。1枚の画像が元画像を含めて4枚になります。その結果、エフェクトをかける前の画像1枚が、 $18 \times 4 = 72$ 枚となります。今回は、実際に写真に撮った画像を用いて学習を行いました。1つのカテゴリーについて、14枚の元画像を用いています。したがって、1つのカテゴリにおいて $14 \times 72 = 1008$ 枚の画像を用いて学習を行います。

```
python rotate_pictures.py images_cropped_trans
```

「images_cropped_trans_rotate」フォルダの中に、画像が保存されます。

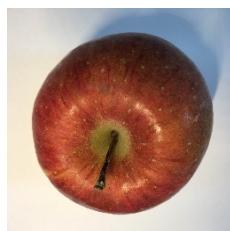


図 11 元画像



図 12 90 度回転



図 13 180 度回転

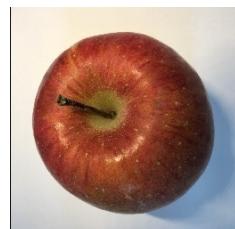


図 14 270 度回転

6 集めた画像を訓練データと検証データにわける

集めた画像にインデックスをつけ、訓練用データと検証用データに分類します。検証用データは全体の 25 % としています。

必要なパッケージのインストール

```
conda install scikit-learn  
conda install pillow  
conda install h5py  
conda install matplotlib
```

訓練データと検証データの作成

makedata.py

```
1 #!/usr/bin/python  
2 # -*- coding: utf-8 -*-  
3  
4 import os, glob  
5 import numpy as np  
6 #from sklearn import cross_validation  
7 from sklearn.model_selection import train_test_split  
8 from keras.preprocessing.image import load_img, img_to_array  
9 import argparse  
10  
11 parser = argparse.ArgumentParser(description='This program Makes train  
and test data from images that are in categorized folders.')  
12 parser.add_argument('dir', help='directory name or path that  
categorized folders are included')  
13 parser.add_argument('cat', nargs='+', help='Directory names of the  
categories to process. Directory names are indexed from zero with the  
given order.')  
14 parser.add_argument('--out', nargs='?', const='out.npy', default='out.  
npy', help='filename or path to output in npy format.')  
15 args = parser.parse_args()  
16  
17 root_dir = args.dir  
18 categories = args.cat  
19 nb_classes = len(categories)  
20  
21 out_filename = args.out  
22 image_size = 32
```

```

23
24 X = []
25 Y = []
26 for idx, cat in enumerate(categories):
27     files = filter((lambda f_or_d: os.path.isfile(f_or_d)), glob.glob(
28         root_dir + "/" + cat + "/*"))
29     print("---", cat, "を処理中")
30     for i, f in enumerate(files):
31         img = load_img(f, target_size=(image_size, image_size))
32         data = img_to_array(img)
33         X.append(data)
34         Y.append(idx)
35 X = np.array(X)
36 Y = np.array(Y)
37 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size
38 = 0.25, shuffle=True)
39 xy = (X_train, X_test, y_train, y_test)
40 np.save(out_filename, xy)
41 print("ok,", len(Y))

```

以下のコマンドを実行すると、区分した結果が入ったファイル out.npy が作成されます。最初のパラメータに、画像が入っているフォルダ名を、それ以降にそのフォルダに含まれるそれぞれのカテゴリーの画像が入ったフォルダ名を指定します。

```
python makedata.py images_cropped_trans_rotate apple BOSS GEORGIA UCC
WONDA
```

カテゴリラベルは、コマンドで「apple BOSS GEORGIA UCC WONDA」と与えた順番に、「0 1 2 3 4」となります。そしてデータを X に、ラベルを Y に保存しています。

上記のプログラムの sklearn.model_selection.train_test_split

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size
= 0.25, shuffle=True)
```

で訓練用データと検証用データを区分しています。データを区分するまえにデータの順番をシャッフルしています。詳しい使い方は、http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html を記載されています。

区分した結果を xy にまとめて、out.npy として保存します。

7 学習の実行

保存したデータを読み込んで学習を実行します。

train_keras.py

```
1 # https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py
2
3 import numpy as np
4 import keras
5 from keras.models import Sequential
6 from keras.layers import Dense, Dropout, Activation, Flatten
7 from keras.layers import Conv2D
8 from keras.layers import Convolution2D, MaxPooling2D
9 from keras.utils import np_utils
10 import argparse
11 from keras.callbacks import ModelCheckpoint
12 #import matplotlib.pyplot as plt
13
14 #root_dir = "./image/"
15 #categories = ["red+apple", "green+apple"]
16 #nb_classes = len(categories)
17 #image_size = 32
18
19 def main():
20     args = get_args()
21
22     nb_classes = args.nb_classes
23     input_file = args.input
24     model_output_hdf5_file = args.output
25     n_epoch = args.epoch
26
27     #X_train, X_test, y_train, y_test = np.load("./image/apple.npy")
28     X_train, X_test, y_train, y_test = np.load(input_file)
29     X_train = X_train.astype("float") / 256
30     X_test = X_test.astype("float") / 256
31     y_train = np_utils.to_categorical(y_train, nb_classes)
32     y_test = np_utils.to_categorical(y_test, nb_classes)
33     model = model_train(X_train, y_train, nb_classes,
34                         model_output_hdf5_file, n_epoch)
35     model_eval(model, X_test, y_test)
36     #backend.clear_session()
37
38 def build_model(in_shape, n_classes):
```

```

38     model = Sequential()
39     model.add(Conv2D(32, (3, 3), padding='same',
40                     input_shape=in_shape))
41     model.add(Activation('relu'))
42     model.add(Conv2D(32, (3, 3)))
43     model.add(Activation('relu'))
44     model.add(MaxPooling2D(pool_size=(2, 2)))
45     model.add(Dropout(0.25))

46
47     model.add(Conv2D(64, (3, 3), padding='same'))
48     model.add(Activation('relu'))
49     model.add(Conv2D(64, (3, 3)))
50     model.add(Activation('relu'))
51     model.add(MaxPooling2D(pool_size=(2, 2)))
52     model.add(Dropout(0.25))

53
54     model.add(Flatten())
55     model.add(Dense(512))
56     model.add(Activation('relu'))
57     model.add(Dropout(0.5))
58     model.add(Dense(n_classes))
59     model.add(Activation('softmax'))

60
61     # initiate RMSprop optimizer
62     opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)

63
64     # Let's train the model using RMSprop
65     model.compile(loss='categorical_crossentropy',
66                     optimizer=opt,
67                     metrics=['accuracy'])
68     return model

69
70 def model_train(X, y, n_classes, output_hdf5_file, nb_epoch):
71     model = build_model(X.shape[1:], n_classes)
72     model.summary()
73     history = model.fit(X, y, batch_size=32, nb_epoch=nb_epoch,
74                          validation_split=0.1,
75                          callbacks=[ModelCheckpoint(output_hdf5_file, monitor='val_loss',
76                                                    verbose=0, save_best_only=True, save_weights_only=False, mode='auto',
77                                                    period=1)])

```

```

75     #hdf5_file = "./image/apple-model.h5"
76     #hdf5_file = model_output_file
77     model.save_weights("final_"+output_hdf5_file)
78     print(history.history)
79     return model
80
81 def model_eval(model, X, y):
82     score = model.evaluate(X, y)
83     print('loss=', score[0])
84     print('accuracy=', score[1])
85
86 def get_args():
87     parser = argparse.ArgumentParser(description='This program Makes
88         train and test data from images that are in categorized folders
89         .')
90     parser.add_argument('nb_classes', type=int, help='Number of
91         categories')
92     parser.add_argument('--input', nargs='?', const='out.npy', default
93         ='out.npy', help='filename or path to input in npy format.')
94     parser.add_argument('--output', nargs='?', const='model.h5',
95         default='model.h5', help='filename or path of the model to
96         output in h5 format.')
97     parser.add_argument('--epoch', nargs='?', const=10, default=10,
98         type=int, help='number of epoch to train.')
99
100    return parser.parse_args()
101
102 if __name__ == "__main__":
103     main()

```

以下のコマンドで学習を実行します。

```
python train_keras.py 5
```

5は、分類するカテゴリーの数です。学習がおわると、モデルを model.h5 に保存し、テストデータにより検証を行います。

```

Anacoda Prompt
Epoch 2/10
3402/3402 [=====] - 32s - loss: 0.1863 - acc: 0.9211 - val_loss: 0.1335 - val_acc: 0.9471
Epoch 3/10
3402/3402 [=====] - 33s - loss: 0.0805 - acc: 0.9703 - val_loss: 0.0148 - val_acc: 0.9968
Epoch 4/10
3402/3402 [=====] - 28s - loss: 0.0410 - acc: 0.9835 - val_loss: 0.0229 - val_acc: 0.9894
Epoch 5/10
3402/3402 [=====] - 27s - loss: 0.0257 - acc: 0.9911 - val_loss: 0.0142 - val_acc: 0.9952
Epoch 6/10
3402/3402 [=====] - 27s - loss: 0.0247 - acc: 0.9928 - val_loss: 9.5838e-04 - val_acc: 1.0000
Epoch 7/10
3402/3402 [=====] - 27s - loss: 0.0186 - acc: 0.9942 - val_loss: 0.0218 - val_acc: 0.9926
Epoch 8/10
3402/3402 [=====] - 28s - loss: 0.0169 - acc: 0.9954 - val_loss: 3.3899e-04 - val_acc: 1.0000
Epoch 9/10
3402/3402 [=====] - 29s - loss: 0.0111 - acc: 0.9966 - val_loss: 0.0093 - val_acc: 0.9979
Epoch 10/10
3402/3402 [=====] - 32s - loss: 0.0183 - acc: 0.9962 - val_loss: 0.0171 - val_acc: 0.9958
1248/1260 [=====] - ETA: 0s loss= 0.0129243870198
accuracy= 0.99460317975
(tensorenv) C:\Users\nakatanikenichi\Lesson\bunrui\real>

```

8 画像の認識

実際に、画像の認識をしてみましょう。※サンプル画像以外のものの場合は、プログラム中の categories の部分を書き換えて下さい。

`predict_keras.py`

```

1 import train_keras as train_keras
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 import argparse
5 from keras.backend import tensorflow_backend as backend
6
7 parser = argparse.ArgumentParser(description='This program predict
     categories from the input image.')
8 parser.add_argument('image', help='filename or path of the input image
     file.')
9 parser.add_argument('--model', nargs='?', const='model.h5', default='
     model.h5', help='filename or path of the model in h5 format.')
10
11 args = parser.parse_args()
12
13 input_image_file = args.image
14 model_file = args.model
15
16 image_size = 32
17
18 #サンプル画像以外のものの場合は、以下の
     categoriesの部分を書き換えて下さい。
19 categories = ["apple", "BOSS", "GEORGIA", "UCC", "WONDA"]
20

```

```
21 nb_classes = len(categories)
22
23 def main():
24     X = []
25     img = load_img(input_image_file, target_size=(image_size, image_size
26                     ))
27     in_data = img_to_array(img)
28     X.append(in_data)
29     X = np.array(X)
30     X = X.astype("float") / 256
31
32     model = train_keras.build_model(X.shape[1:], nb_classes)
33     #model.load_weights("./image/apple-model.h5")
34     model.load_weights(model_file)
35
36     pre = model.predict(X)
37     print(pre)
38     for i in range(len(categories)):
39         if pre[0][i] > 0.5:
40             print(categories[i])
41
42     backend.clear_session()
43
44 if __name__ == '__main__':
45     main()
```

訓練に用いたものとは別に写真を撮り、それを認識させます。りんごも訓練に使ったものとは、別のものです。



図 15 test0.jpg



図 16 test1.jpg



図 17 test2.jpg



図 18 test3.jpg



図 19 test4.jpg

```
python predict_keras.py test0.jpg
[[ 9.9999981e-01    1.44897010e-08    2.95493219e-25    1.75999522e-23
  1.20560941e-07]]
apple
```

りんごは、他のものとは計上もことなるため、9.9999981e-01 とかなりはつきり認識されるようです。

```
python predict_keras.py test1.jpg
[[ 9.90515179e-14    6.78819716e-01    6.57590499e-06    3.21173072e-01
  5.61466607e-07]]
BOSS
```

BOSS は、正しく認識されましたが、6.78819716e-01 となっており、UCC が、2 番目の 3.21173072e-01 となっています。缶の色が似ていためではと思います。

```
python predict_keras.py test2.jpg
[[ 0.0000000e+00    8.28396824e-31    1.0000000e+00    4.09071975e-32
  1.96705918e-24]]
GEORGIA
```

GEORGIA は、色が青で他と違うためか、1.0000000e+00 と他のものと正確に区別しています。

```

python predict_keras.py test3.jpg
[[ 1.24014774e-12    6.86096326e-02    8.85054305e-06    9.31380928e-01
  5.61855416e-07]]
UCC

```

UCC のほうは、9.31380928e-01 と判別されています。色が似ている BOSS は、6.86096326e-02 となっています。

```

python predict_keras.py test4.jpg
[[ 1.26062186e-14    4.02524120e-06    4.83060887e-07    1.16335996e-05
  9.99983907e-01]]
WONDA

```

WONDA も、9.99983907e-01 と正しく判別されています。

判別ができることが確認できましたので、でき上がったモデルを Raspberry Pi へアップロードします。

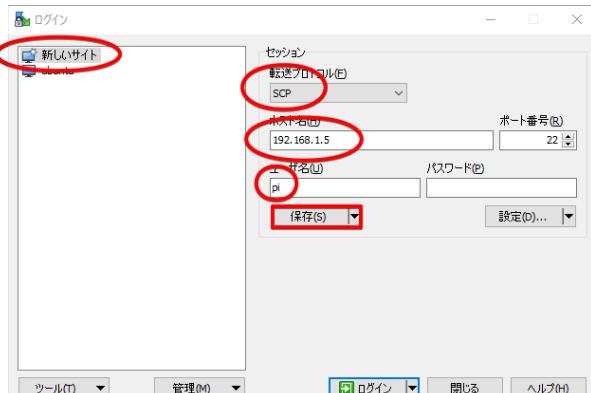
上記の数値は、訓練時のパラメータの初期設定や、訓練の際にドロップアウトを行っていることなどのために、訓練ごとに異なるものとなります。今回の例の場合、リンゴと缶コーヒーの区別や、缶コーヒーでも色が大きく異なるものの区別はうまく行われるようです。しかし、例えば、BOSS と UCC のように、形状も色も似ているもの同士の区別は誤ることがあります。

これらについてより正確に区別を行うには、訓練の際に与える画像データの解像度や与える画像データの量、モデルの複雑さ等についての検討が必要です。

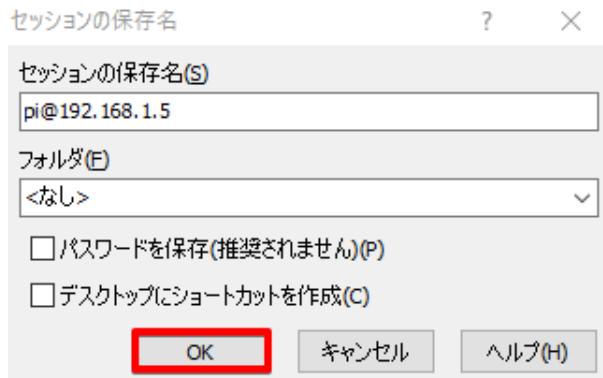
また、カテゴリーの区分についても、検討が必要だと思います。今回は、リンゴと缶コーヒーのそれぞれの銘柄を別々のカテゴリーに区分してみました。しかし、「リンゴ」「缶コーヒー」の2つのカテゴリーに分類して訓練を行うことの方がより一般的なようにも思います。これらのデータの与え方はそれぞれの事例ごとに、その利用目的によっても変わってくるものであると思います。

9 Raspberry Pi へのモデルのコピー

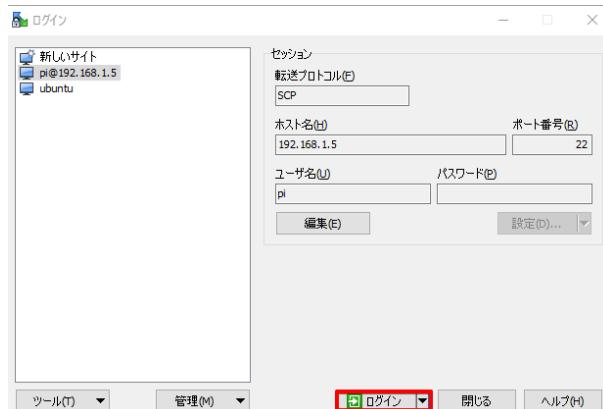
作成したモデル (model.h5) を Raspberry Pi へアップロードします。今回は、WinSCP を利用してアップロードを行います。まず、WinSCP を起動します。



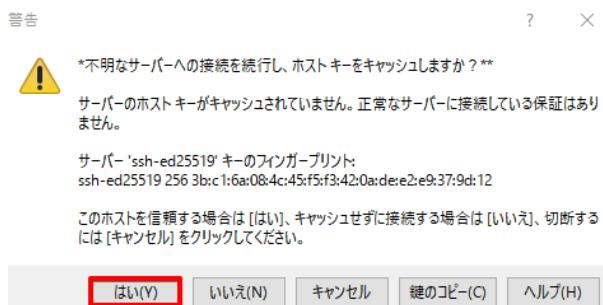
- ①「新しいサイト」を選択します。
- ②「転送プロトコル」として「SCP」を選択します。
- ③「ホスト名」に Raspberry Pi の IP アドレスを入力します。
- ④「ユーザ名」を「pi」とします。
- ⑤今回の設定を保存するため、「保存」ボタンを押します。



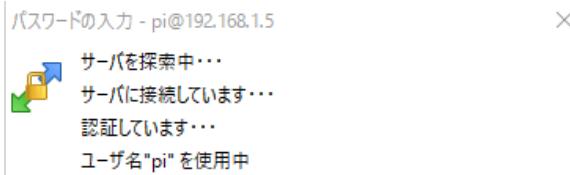
セッションの保存名を確認し、「OK」ボタンを押します。



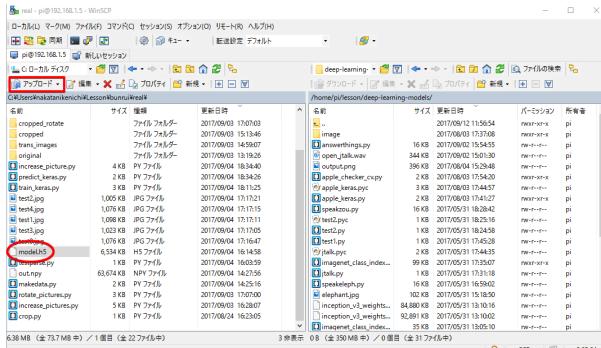
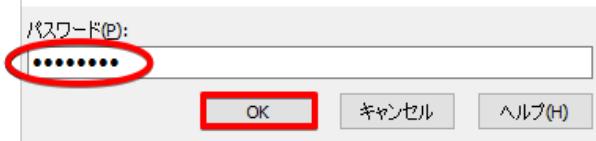
保存したセッションを選択し、「ログイン」ボタンを押します。



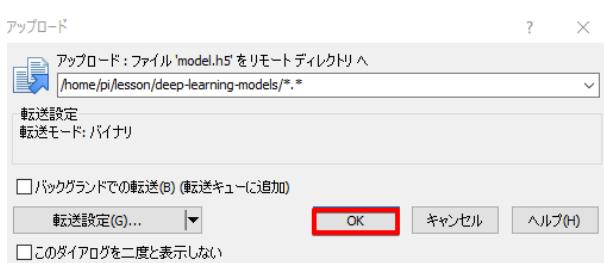
「はい」ボタンを押します。



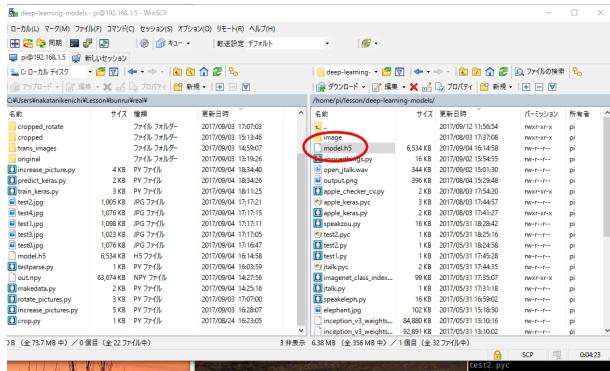
パスワードを入力し、「OK」ボタンを押します。



①画面の左側でアップロードするファイル「model.h5」を選択します。②画面の右側でアップロード対象のディレクトリ「/home/pi/lesson/deep-learning-models」を選択します。③「アップロード」ボタンを押して、ファイルをアップロードします。



「OK」ボタンを押します。



アップロード完了後、画面の左側に、アップロードしたファイルがあることを確認します。

同様に、「ac_checker.py」「train_keras.py」を同じ場所にアップロードします。train_keras.py、ac_checker.pyがすでに Raspberry Pi 上にある場合でも、最新版を使うために上書きしてアップロードして下さい。

10 Raspberry Pi でリンゴとコーヒーの種類を見分ける

アップロードしたモデルを使って、見せたものを見分けます。※サンプル画像以外のものの場合は、プログラム中の categories の部分を書き換えて下さい。

ac_checker.py

```

1 # -*- coding: utf-8 -*-
2 import train_keras as tr
3 import sys, os
4 import numpy as np
5 import subprocess
6 import cv2
7 from keras.preprocessing.image import load_img, img_to_array
8 import jtalk
9
10 cam = cv2.VideoCapture(0)
11 image_size = 32
12
13 #サンプル画像以外のものの場合は、以下の
14 #categoriesの部分を書き換えて下さい。
14 categories = [u"りんご", u"ボス", u"ジョージア", u"ユーシーシー", u"ワン
15 ダ"]
15 nb_classes = len(categories)
16
17 def main():
18
19     while(True):

```

```

20     ret, frame = cam.read()
21     cv2.imshow("Show FLAME Image", frame)
22
23     k = cv2.waitKey(1)
24     if k == ord('s'):
25         cv2.imwrite("output.png", frame)
26         cv2.imread("output.png")
27
28     X = []
29     img = load_img("./output.png", target_size=(image_size,
30                     image_size))
30     in_data = img_to_array(img)
31     X.append(in_data)
32     X = np.array(X)
33     X = X.astype("float") / 256
34
35     model = tr.build_model(X.shape[1:], nb_classes)
36     model.load_weights(".model.h5")
37
38     pre = model.predict(X)
39     print(pre)
40     for i in range(len(categories)):
41         if pre[0][i] > 0.5:
42             print(categories[i])
43             text = u'これは' + categories[i] + u'だよ'
44             text = text.encode('utf-8')
45             jtalk.jtalk(text)
46
47     elif k == ord('q'):
48         break
49
50     cam.release()
51     cv2.destroyAllWindows()
52
53 if __name__ == '__main__':
54     main()

```

VNC ビューワーで Raspberry pi に接続します。端末を立ち上げます。

```

source ~/.profile
workon cv

```

```
cd ~/Lesson/deep-learning-models  
python ac_checker.py
```

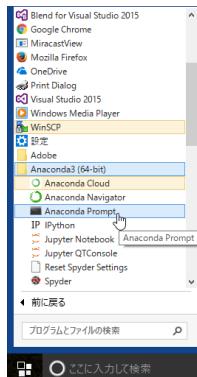
「s」キーを押すと、認識を開始します。缶コーヒーなどを見せて正しく認識されているか確認してみましょう。サンプル画像でうまくいったら、サンプル画像以外の画像でも試してみましょう。画像データの集め方は、実際に写真を撮る以外に、WEBサイトからAPIを使ってデータを集める方法もあります。「(参考) WEBサイトからAPIを使ってデータを集める」を参考にしてみて下さい。

11 (参考)tensorenv 環境の作成

以前の授業を休んだ等のため、tensorenv環境を作成していない方は、以下に従ってtensorenv環境を作成して下さい。ここでは、TensorFlowを自分のPC上で使うために必要な環境設定を行います。

また、第1回目の授業を欠席した人は、このテキストの最後の章「(参考) Widnowsの環境設定」を参照して、Anacondaをインストールしてから、以下の作業を行って下さい。

11.1 Anaconda Prompt の立ち上げ



Anaconda Prompt を選択します。

11.2 tensorenv 環境の作成

作業ディレクトリへ移動します。ホームディレクトリの下に「Lesson」というディレクトリがあることを前提に作業します。ない場合には作成して下さい。

```
cd %homepath%  
mkdir Lesson  
cd Lesson
```

python3.5を使う環境を作成します。環境の名前を「tensorenv」とします。環境の作成が終ったら、その環境に入ります。

```
conda create --name=tensorenv python=3.5  
activate tensorenv
```

環境に入ると、コマンドプロンプトの先頭に環境名(tensorenv)が表示されます。

11.3 TensorFlow のインストール

作成した環境に、TensorFlow やその他、プログラムの実行に必要なモジュール等をインストールします。この TensorFlow をバックエンドに、容易に素早くディープラーニングプロトタイプの迅速な実験を可能にすることに重点を置いて開発されたライブラリが Keras です。Keras もあわせてインストールしておきます。インストール中に、「Proceed ([y]/n)?」と聞かれたら「y」を入力します。

```
pip install --upgrade pip  
CPU版  
pip install tensorflow  
pip install jupyter  
conda install scipy  
pip install keras  
conda install matplotlib  
conda install -c menpo opencv3
```

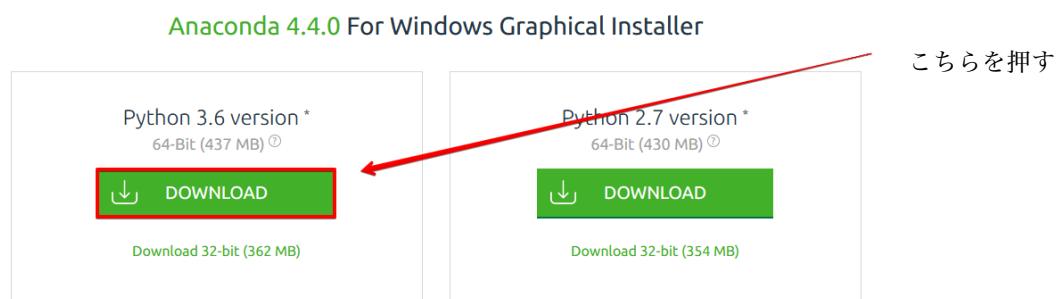
12 (参考) Widnows の環境設定

第1回目の授業を欠席して、Widnows の環境設定を行っていない場合には、以下の設定を行ってから、本テキストの内容を実行して下さい。

12.1 Anaconda のインストール

<https://www.continuum.io/downloads#windows>

からダウンロードします。



ダウンロードしたファイルをダブルクリックして実行し、インストールを行います。

※すでに Anaconda (Python2.7 version を含む) がインストールされている場合には、そのまま、次に進んで問題ありません。

13 (参考) WEB サイトから API を使ってデータを集める

課題を行うにあたって、皆さんのが選択した課題に種類によっては、画像データ等のデータを多くを集める必要が出てくる場合があります。今回は、WEB サイトの API を使って画像データを集める例を説明致します。WEB サイトから集めることができるデータには、画像データ以外にも多くのデータがあります。

13.1 「フォト蔵」からダウンロードする

13.1.1 概要

フォト蔵（フォトぞう）は簡単に写真を投稿・共有できる無料のフォトアルバムサービスです。スマートフォンやデジタルカメラで撮った写真をみんなに見てもらったり、友達や家族でお互いの写真を見せあつたりすることができます。フォト蔵の URL は以下の通りです。

<http://photozou.jp/>

13.1.2 API

フォト蔵 API とは、他のプログラムからフォト蔵にアクセスするために提供している外部プログラムインターフェースです。API によるアクセスは 1 リクエスト/秒を目安とすることとなっています。このように、API にアクセスする際には、サービスに影響の出る過剰なアクセスは避ける必要があります。

フォト蔵 API の仕様は、

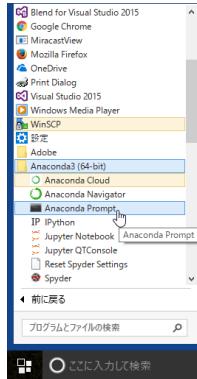
<http://photozou.jp/basic/api>

に記載されています。今回は、これらの API の中から、「search_public」という API を利用しています。

http://photozou.jp/basic/api_method_search_public

API の仕様について各自確認してみて下さい。

13.1.3 Anaconda Prompt の立ち上げ



Anaconda Prompt を選択します。

13.1.4 実行用のツールのダウンロード

実行用のツールをダウンロードします。授業関連ファイル lesson に入っていますので、ダウンロードします。そして、関連ファイルの入っているディレクトリに移動します。

```
cd %homepath%
```

```
cd Lesson  
git clone https://github.com/k1nk/ch8.git  
cd ch8
```

13.1.5 モジュールのインストール

実行に必要なモジュールをインストールします。以前に授業で作成した tensorenv 環境で作業を行います。

```
activate tensorenv  
conda install urllib3  
conda install beautifulsoup4  
conda install requests
```

授業に欠席したなどの理由で、「activate tensorenv」で環境に入れない場合には、このテキストの「(参考) tensorenv 環境の作成」を行ってから続けて下さい。

13.1.6 プログラム

phototkura_downloader.py

```
1 #!/usr/bin/env python  
2 # -*- coding: utf-8 -*-  
3  
4 import sys, os, re, time  
5 import urllib.request as req #python3  
6 import urllib.parse as parse #python3  
7  
8 import json  
9 import argparse  
10  
11 args = sys.argv  
12 parser = argparse.ArgumentParser(description='This script download  
    files from photo_kura.')  
13 parser.add_argument('kwd', \  
    action='store', \  
    nargs='?', \  
    const="apple", \  
    default="apple", \  
    type=str, \  
    choices=None, \  
    help='Keyword to search photos.', \  
    metavar=None)  
22
```

```

23 parser.add_argument('dir', \
24     action='store', \
25     nargs='?', \
26     const='./image', \
27     default='./image', \
28     type=str, \
29     choices=None, \
30     help='Directory path where your taken photo files are located \
31     .', \
32     metavar=None)
33
34
35 # APIのURLを指定
36 PHOTOZOU_API = "https://api.photozou.jp/rest/search_public.json"
37 CACHE_DIR = "./image/cache"
38
39 # フォト蔵のAPIを利用して画像を検索する --- (※1)
40 def search_photo(keyword, offset=0, limit=100):
41     # APIのクエリを組み立てる
42     keyword_enc = parse.quote_plus(keyword)
43     q = "keyword={0}&offset={1}&limit={2}".format(
44         keyword_enc, offset, limit)
45     url = PHOTOZOU_API + "?" + q
46     # キャッシュ用のディレクトリを作る
47     if not os.path.exists(CACHE_DIR):
48         os.makedirs(CACHE_DIR)
49     cache = CACHE_DIR + "/" + re.sub(r'[^a-zA-Z0-9\%\#]+', '_', url)
50     if os.path.exists(cache):
51         return json.load(open(cache, "r", encoding="utf-8"))
52     print("[API] " + url)
53     req.urlretrieve(url, cache)
54     time.sleep(1) # --- 礼儀として1秒スリープ
55     return json.load(open(cache, "r", encoding="utf-8"))
56
57 # 画像をダウンロードする --- (※2)
58 def download_thumb(info, save_dir):
59     if not os.path.exists(save_dir): os.makedirs(save_dir)
60     if info is None: return
61     if not "photo" in info["info"]:

```

```

62     print("[ERROR] broken info")
63     return
64 photolist = info["info"]["photo"]
65 for photo in photolist:
66     title = photo["photo_title"]
67     photo_id = photo["photo_id"]
68     url = photo["thumbnail_image_url"]
69     path = save_dir + "/" + str(photo_id) + "_thumb.jpg"
70     if os.path.exists(path): continue
71     try:
72         print("[download]", title, photo_id)
73         req.urlretrieve(url, path)
74         time.sleep(1) # --- 礼儀として1秒スリープ
75     except Exception as e:
76         print("[ERROR] failed to download url=", url)
77
78 # 検索結果を全部取得する --- (※3)
79 def download_all(keyword, save_dir, maxphoto = 1000):
80     offset = 0
81     limit = 100
82     while True:
83         # APIを呼び出す
84         info = search_photo(keyword, offset=offset, limit=limit)
85         if info is None:
86             print("[ERROR] no result"); return
87         if (not "info" in info) or (not "photo_num" in info["info"]):
88             print("[ERROR] broken data"); return
89         photo_num = info["info"]["photo_num"]
90         if photo_num == 0:
91             print("photo_num = 0, offset=", offset)
92             return
93         # 写真情報が含まれていればダウンロード
94         print("*** download offset=", offset)
95         download_thumb(info, save_dir)
96         offset += limit
97         if offset >= maxphoto: break
98
99 if __name__ == '__main__':
100     # モジュールとして使わないで単独で実行する時
101     query = args.kwd

```

```

102
103     #save_dir
104     DIR = args.dir
105     query_p='+'.join(query.split())
106
107     if not os.path.exists(DIR):
108         os.mkdir(DIR)
109     DIR = os.path.join(DIR, query_p.split()[0])
110     if not os.path.exists(DIR):
111         os.mkdir(DIR)
112
113     download_all(query,DIR)

```

13.1.7 ダウンロードの方法

以下のコマンドで、写真をダウンロードします。

```
python phototkura_downloader.py "青リンゴ" green_apple_kura
```

この例では「青リンゴ」というキーワードで、「green_apple_kura」というディレクトリの中に画像をダウンロードします。デフォルトで maxphoto = 1000 まで、写真をダウンロードします。ダウンロードした画像から、学習に適したものを手作業で振り分けます。

キーワード等を自分の課題に関するものに変更して、ダウンロードを行って見ましょう。

13.2 「Goole 画像検索」からダウンロードする

google_download.py

```

1  # -*- coding: utf-8 -*-
2
3  from bs4 import BeautifulSoup
4  import requests
5  import re
6  #import urllib2 #python2
7  import urllib.request
8  import urllib.parse as parse #python3
9  import os
10 #import cookielib
11 import json
12
13 import argparse
14

```

```

15 #args = sys.argv
16 parser = argparse.ArgumentParser(description='This script download
17     files from photo_kura.')
18 parser.add_argument('kwd', \
19     action='store', \
20     nargs='?', \
21     const="red apple", \
22     default="red apple", \
23     type=str, \
24     choices=None, \
25     help='Keyword to search photos.', \
26     metavar=None)
27
28 parser.add_argument('dir', \
29     action='store', \
30     nargs='?', \
31     const=".\\Pictures_google", \
32     default=".\\Pictures_google", \
33     type=str, \
34     choices=None, \
35     help='Directory path where your taken photo files are located
36         .', \
37     metavar=None)
38
39 args = parser.parse_args()
40
41 def get_soup(url,header):
42     return BeautifulSoup(urllib.request.urlopen(urllib.request.Request(
43         url,headers=header)), 'html.parser')
44 #return BeautifulSoup(urllib2.urlopen(urllib2.Request(url,headers=
45         header)), 'html.parser')
46
47 query_org = args.kwd
48 label="0"
49 print(query_org)
50
51 #query= query.split()
52 #query='+'.join(query)
53 query = parse.quote_plus(query_org)

```

```

51 url="https://www.google.co.in/search?q="+query+"&source=lnms&tbo=isch"
52 print(url)
53 #add the directory for your image here
54 #DIR="Pictures_google"
55 DIR=args.dir
56 header={'User-Agent':"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 Safari/537.36"
57 }
58 soup = get_soup(url,header)
59
60 ActualImages=[]# contains the link for Large original images, type of
61 #image
62 for a in soup.find_all("div", {"class": "rg_meta"}):
63     link , Type = json.loads(a.text)[ "ou"] , json.loads(a.text)[ "ity"]
64     ActualImages.append((link,Type))
65
66 print("there are total" , len(ActualImages),"images")
67
68 if not os.path.exists(DIR):
69     os.mkdir(DIR)
70 DIR = os.path.join(DIR, query_org.split()[0])
71
72 if not os.path.exists(DIR):
73     os.mkdir(DIR)
74 ###print images
75 for i , (img , Type) in enumerate(ActualImages):
76     try:
77         req = urllib.request.Request(img,data=None,headers=header)
78         #req = urllib2.Request(img, headers={'User-Agent' : header}) #
79             python2
80         response = urllib.request.urlopen(req)
81         raw_img = response.read()
82         #raw_img = urllib2.urlopen(req).read() #python2
83
84         cntr = len([i for i in os.listdir(DIR) if label in i]) + 1
85         print(cntr)
86         if len(Type)==0:
87             f = open(os.path.join(DIR , label + "_" + str(cntr)+".jpg") ,
88                  'wb')
89         else :

```

```
87     f = open(os.path.join(DIR , label + "_" + str(cntr) + ". " + Type
88                           ), 'wb')
89
90     f.write(raw_img)
91     f.close()
92 except Exception as e:
93     print("could not load : "+img)
94     print(e)
```

以下のコマンドで、写真をダウンロードします。

```
activate tensorenv #python3で実行
python google_download.py "リンゴ" apple_google
```

取得できるのは、100枚までです。