# Raspberry Piを用いたPythonによる画像認識

電子情報科　表川奈央　川越悠生　北川航太朗
酒本優大　中村亮太　南出羅生愛

## 背　景

AI技術が発展している中で、画像認識技術を使うAIを目にすることがある。そこで昨年の先輩の発表で深層学習について興味を持ち、深層学習を用いたモノづくりを行いたいと考えた。

## 目　的

深層学習を用いて画像認識を行うことを通して画像認識やPythonについての理解を深め、卒業後に生かすこと。

## 方　法

Raspberry Piを使い、Pythonで画像認識のプログラムを作成した。画像認識を行うにあたって、事前に花の画像を用意した。

## 経　過

各自で深層学習とPythonについて調べた。学校のパソコンでは研究することが不可能であることが分かった。そのため、市販されているキットを使用することになった。その結果、プログラムの解説付きのRaspberry Pi4を見つけ、使うことにした。

## 結　果

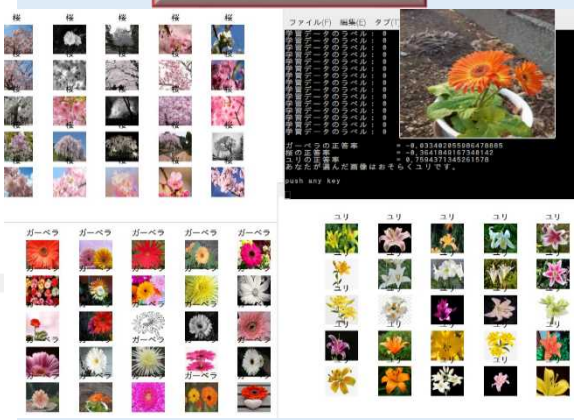
パッケージの読み込み

```
import matplotlib.pyplot as plt
import os,cv2,random,sys
from PIL import Image
import numpy as np

DATADIR = "/home/pi/Documents/python元画像/"
CATEGORIES = ["ガーベラ", "桜","ユリ"]
IMG_SIZE = 300
training_data = []
def create_training_data():                                    #データの学習
    for class_num, category in enumerate(CATEGORIES):
        path = os.path.join(DATADIR, category)
        for image_name in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, image_name))
                img_resize_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([img_resize_array, class_num])
            except Exception as e:
                pass
create_training_data()
random.shuffle(training_data)                                  # データをシャッフル
x_train = []                                                   # 画像データ使用
y_train = []                                                   # ラベル情報

# データセット作成
for feature, label in training_data:
    x_train.append(feature

        y_train.append(label)

# numpy配列に変換
x_train = np.array(x_train)
y_train = np.array(y_train)

#  画像読み込み&予測
im=input("検査ファイル: No.__")
_dir="/home/pi/Documents/python元画像/花(テスト)/"+im+".png"
img=cv2.imread(_dir)
img=cv2.resize(img,(300,300))
img_hist= cv2.calcHist([img], [0], None, [256], [0, 256])
lea_hist=cv2.calcHist([x_train], [0], None, [256], [0, 256])
for i in range(0, 25):
    print("学習データのラベル：", y_train[i])
    plt.subplot(5, 5, i+1)
    plt.axis('off')
    jp={"family":"IPAexGothic"}
    plt.title(label = 'ガーベラ' if y_train[i] == 0 else '桜' if y_train[i] == 1   else 'ユリ',**jp)
    img_array = cv2.cvtColor(x_train[i], cv2.COLOR_BGR2RGB)
    plt.imshow(img_array)
print("正答率=",cv2.compareHist(img_hist,lea_hist,cv2.HISTCMP_CORREL))
cv2.imshow("ｱｲｴｴ!ﾆﾝｼﾞｬ!ﾗｶﾝﾃﾞﾆﾝｼﾞｬ!ｳﾏ",img)
cv2.waitKey(0)
plt.show()
```

検査したい画像の読み込み

近似値の計算

## 実行結果



## 考　察

画像の近似値の出し方を色にした結果、あまり精度がよくなかったため、形の読み取りを加えることで精度が上がると思った。
また、予測結果から何の画像か判別するため、「Ａ」の画像を入力しても、「B」と出力することがあるため、改善点として修正したいと思った。