Python を用いた犬の画像識別

池本　響　　遠藤　大暉
木原　隆樹　　小林　那緒

１．研究概要

プログラミング言語の Python で OpenCV を用いて、機械学習データを作成し犬の顔を識別する。

用語の解説

（１） Python について

AI 分野のプログラミングや機械学習で多く使用されていることから将来性が高い言語

（２） OpenCV について

画像・動画に関する処理機能をまとめたオープンソースのライブラリ

２．研究の具体的内容

【１】制作環境の準備

まず初めに Python を使うための環境構築をした。Anaconda と呼ばれる Python パッケージを使用し、その中にある統合開発環境の Spyder で Python の開発を行った。

次に OpenCV を使用するためにアナコンダプロンプトで OpencvPython をダウンロードする。それにより OpenCV を使用できる。

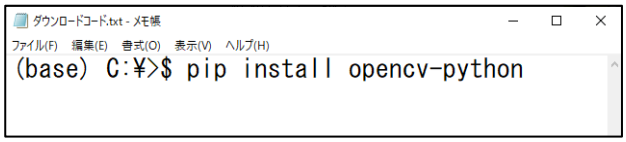

図１　ダウンロードコード

【２】画像収集

画像収集では Imagedownloader という、GoogleChrome の拡張機能を使用して画像収集をした。初めに、拡張機能のダウンロード画面で Imagedownloader のインストールを行った。

そして、GoogleChrome を開き、収集したい画像があるウィンドウを表示させて、右上にある拡張機能マークをクリックすると（図２）のような画面が表示される。そこで保存フォルダの名前などを設定して DOWNLOAD をクリックすると、PC のダウンロードに集めた画像が保存される。

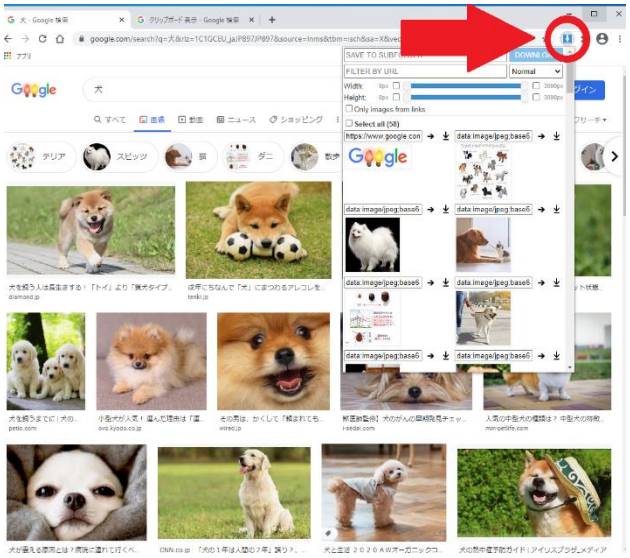

図２　Imagedownloader 実行画面

【３】画像加工について

機械学習に使う画像は集めた画像をそのまま使うことはできず、学習しやすいように画像を加工しなければならない。まずは画像の中心に識別対象（今回は犬の顔）が来るように、Windows のフォトでトリミング処理をして画像の中央に犬の画像がくるように加工した。すべての学習データのトリミング処理が完了した後は、すべての画像を 200×200 のサイズにリサイズすることでポジリストの値を同じようにすることができる。

【４】カスケード分類器の作成

カスケード分類器という学習画像すべての特徴をまとめたデータを作成する。

まず初めに作業フォルダの作成と、必要なファイルの準備をした。そのフォルダは、cascade, pos, neg, vec を作成した。

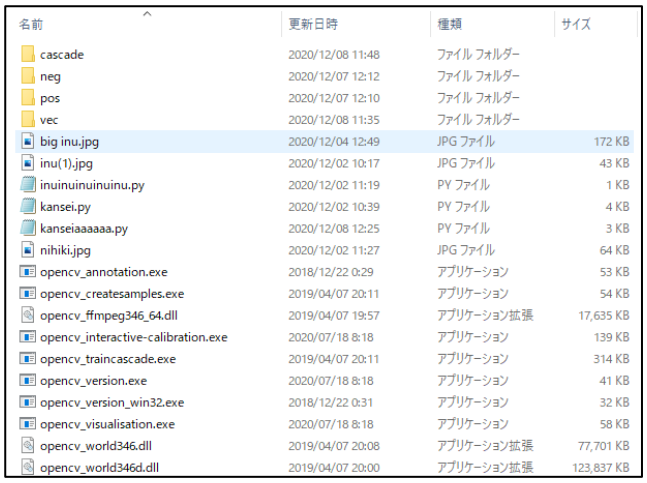

図３　作業フォルダとファイル一覧

またファイルは OpenCV の公式からダウンロードする。

その後画像ファイルの準備をする。Neg のフォルダに不正解画像を入れる。また Pos に正解画像を入れる。不正解画像 444 枚正解画像 746 枚を準備する。次にコマンドプロンプトで、不正解画像ファイル、正解画像ファイルのリストを作る。

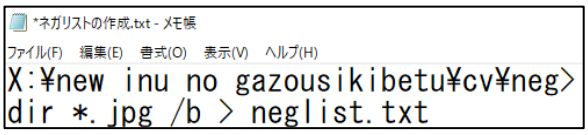

図４　ネガリストの作成

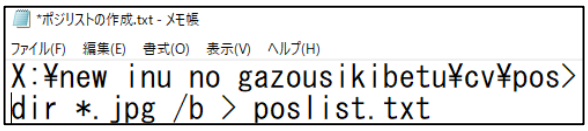

図５　ポジリストの作成

作ったネガリストの中の名前にすべてフルパスを追記する。

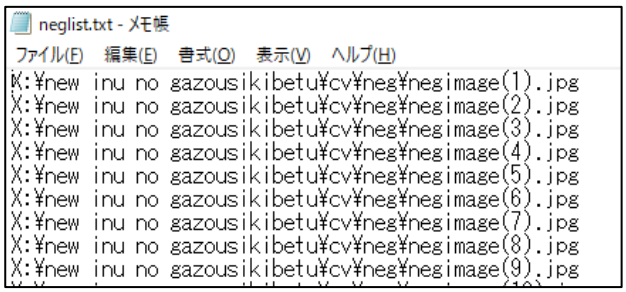

図６　ネガリスト

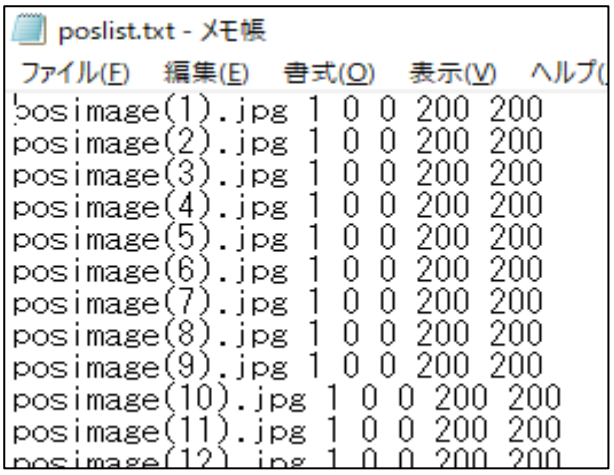

図７　ポジリスト

ポジリストの 1 0 0 200 200 という数値の 1 は正解画像に写っている犬の顔の数、
0 0 は犬の顔を囲む始点の x,y 座標、
200 200 は終点の x,y 座標を表している。

その後 opencv_createsamples によるベクトルファイルを作成する。
opencv_createsamples を使うことによって一枚の画像から角度やサイズを変更した大量のサンプルを自動生成ことができる。これも、同じようにコマンドプロンプトで作成する。

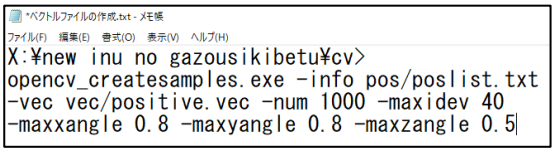

図８　ベクトルファイルの作成

ここまでの下準備を使用し、カスケード分類器を作成する。Cascade のフォルダに出る。

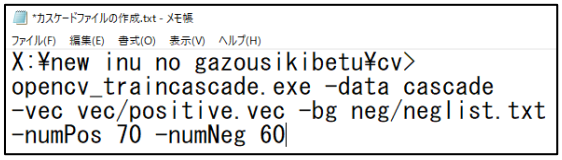

図９　カスケード分類器の作成

コードを実行すると計算が始まる。コマンドプロンプトには以下のことが表示される。

N	HR	FA
1	1	1
2	1	1
3	1	1
4	1	1
5	0.99665	0.952113
6	0.998325	0.95493

図 10　実行中

N：繰り返し回数
HR：犬の画像を正しく犬と認識した確率
FA：その他の画像を犬と誤って認識した確率

枚数が多いほど計算がかかる。今回は処理時間が 11 分かかったが、多いと数時間かかる。またエラーが出ると cascade.xml が出てこない。正解画像と不正解画像の枚数をぴったりにするとエラー発生するので９割や８割の数を設定する。

名前	更新日時	種類	サイズ
cascade.xml	2020/12/08 11:48	XML ドキュメント	140 KB
params.xml	2020/12/08 11:37	XML ドキュメント	1 KB
stage0.xml	2020/12/08 11:37	XML ドキュメント	2 KB
stage1.xml	2020/12/08 11:37	XML ドキュメント	3 KB
stage2.xml	2020/12/08 11:38	XML ドキュメント	3 KB
stage3.xml	2020/12/08 11:38	XML ドキュメント	3 KB
stage4.xml	2020/12/08 11:38	XML ドキュメント	3 KB
stage5.xml	2020/12/08 11:39	XML ドキュメント	3 KB
stage6.xml	2020/12/08 11:39	XML ドキュメント	4 KB
stage7.xml	2020/12/08 11:40	XML ドキュメント	4 KB
stage8.xml	2020/12/08 11:41	XML ドキュメント	4 KB
stage9.xml	2020/12/08 11:41	XML ドキュメント	5 KB
stage10.xml	2020/12/08 11:42	XML ドキュメント	5 KB
stage11.xml	2020/12/08 11:43	XML ドキュメント	4 KB
stage12.xml	2020/12/08 11:43	XML ドキュメント	5 KB
stage13.xml	2020/12/08 11:44	XML ドキュメント	5 KB
stage14.xml	2020/12/08 11:45	XML ドキュメント	5 KB
stage15.xml	2020/12/08 11:45	XML ドキュメント	5 KB
stage16.xml	2020/12/08 11:46	XML ドキュメント	5 KB
stage17.xml	2020/12/08 11:47	XML ドキュメント	5 KB
stage18.xml	2020/12/08 11:48	XML ドキュメント	5 KB
stage19.xml	2020/12/08 11:48	XML ドキュメント	6 KB

図 11　作成結果

Cascade.xml:画像の特徴量をまとめたファイル
Params.xml：情報を取得するためのファイル
stage??.xml:画像をステージごとに分けたそれぞれの特徴量をまとめたファイル

【５】プログラムの作成

初めに、参照ボタンを押すと画像ファイルの絶対パスを取得するプログラムを作成し、ボタン作成には Tkinter という Python のライブラリを使用した。次に、画像識別プログラムを作成し、顔認識をさせた。画像識別には同じく Python のライブラリである OpenCV を使用した。

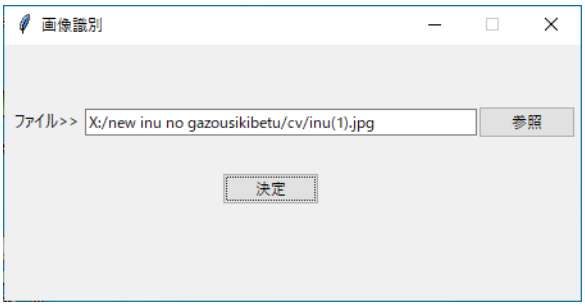

図 12　画像選択画面例

```
# 参照ボタンのイベント
# button1クリック時の処理
def button1_clicked():


    fTyp = [("","*")]
    iDir = os.path.abspath(os.path.dirname(__file__))
    filepath = filedialog.askopenfilename(filetypes = fTyp,initialdir = iDir)
    file1.set(filepath)
```

図 13　参照ボタンの処理例

```
def button2_clicked():

        file2 = file1.get()
        print("%s" %file2)
        img = cv.imread('%s' %file2)
        grayimg = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        custom_cascade = cv.CascadeClassifier('./cascade/cascade.xml')
        custom_rect = custom_cascade.detectMultiScale(grayimg, scaleFactor=1.07, minNeighbors=2, minSize=(1, 1))
        print(custom_rect)
        if len(custom_rect) > 0:
            for rect in custom_rect:
                cv.rectangle(img, tuple(rect[0:2]), tuple(rect[0:2]+rect[2:4]), (0, 0, 255), thickness=3)
        cv.imshow('image', img)
        cv.waitKey(0)
        cv.destroyAllWindows()
```

図 14　決定ボタンの処理例

```python
if __name__ == '__main__':
    # rootの作成
    root = tkinter.Tk()
    root.title('画像識別')
    root.geometry('450x200')
    root.resizable(False, False)


    # Frame1の作成
    frame1 = ttk.Frame(root, width=400,height=200)
    frame1.grid()


    # 参照ボタンの作成
    button1 = ttk.Button(root, text=u'参照',command = button1_clicked)
    button1.place(x=370,y=48)


    # ラベルの作成
    #「ファイル」ラベルの作成
    s = tkinter.StringVar()
    s.set('ファイル>>')
    label1 = ttk.Label(frame1, textvariable=s)
    label1.place(x=6,y=50)


    # 参照ファイルパス表示ラベルの作成
    file1 = tkinter.StringVar()
    file1_entry = ttk.Entry(frame1, textvariable=file1, width=50)
    file1_entry.place(x=63,y=50)


    # frame2の作成
    frame2 = ttk.Frame(root,width=100,height=100)
    frame2.place(x=170,y=100)


    # 決定ボタンの作成
    button2 = ttk.Button(frame2,text='決定',command = button2_clicked)
    button2.pack(side = 'left')
```

図 15　GUI の表示処理例

ここに判別させたい画像を選択させる。それにより下の図が出て結果が表示される。

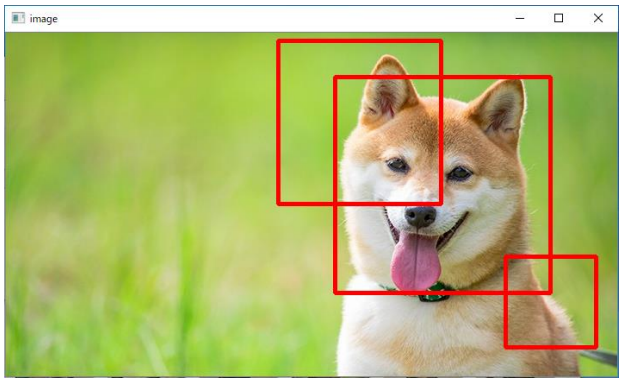

図 16　結果表示

画像の中で一致した場所を四角で囲んでいる。

今回は正解画像と不正解画像の枚数が少なかったので赤の四角が多くなってしまった。

３．反省

最初は、犬と猫を同時に識別する予定だった。しかし、画像収集が思いのほか時間がかかったため犬だけの識別になった。また，コマンドプロンプトのエラーに苦戦した。このエラーを通してファイルの扱いを学ぶことができた。今後はそれを活かして画像識別の精度を上げていきたい。

４．参考文献

OpenCV の勉強③（分類器を作成してみる）
https://qiita.com/takanorimutoh/items/5bd88f3d17239a147581

OpenCV のカスケード分類器を自作して画像認識-パソコン工房
https://www.pc-koubou.jp/magazine/21280

Python でファイル処理の GUI プログラムを作ってみた！-Qiita
https://qiita.com/Tomo666/items/8e5ebf7d8b0b21c8fd3a