PC Based Automatic Foot Massager

Although the author and publisher have made every effort to ensure that the information in this writing was correct at press time, the author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

paid link

In today's project, we will build an automatic foot massager that will accommodate different foot sizes. We apply deep learning to determine foot's pressure points. Basically, it is a regression problem. We build neural network models by performing regression with CNNs and Keras.

A collection of foot images are saved in a directory and will be used as training data. For simplicity, this project only deals with right foot. The pressure points only cover lung, neck, and throat. Of course, this program can be expanded so that it also covers other pressure points.

As you know, the models are built using the training data with correct input and output values. For the output values, we determine the x,y positions of the pressure points (lung, neck, and throat) for each of the training images. The data is saved in a csv file (*points.csv*)

```
Lung_X,Lung_Y,Neck_X,Neck_Y,Throat_X,Throat_Y
52,151,126,61,111,53
42,127,125,54,108,44
49,121,125,61,108,52
43,130,124,57,100,49
45,119,129,60,110,48
42,155,112,58,88,56
53,130,114,45,103,37
45,152,133,55,114,50
50,129,127,51,116,42
54,134,123,53,108,50
46,133,124,55,110,54
46,115,122,53,107,39
42,136,119,53,99,56
46,119,122,49,109,40
42,131,130,42,107,38
46,143,117,54,104,50
54,128,130,49,113,39
37,116,116,40,103,44
46,119,121,50,107,52
50,127,125,48,112,55
54,128,120,42,102,44
46,124,114,50,99,47
47,112,125,49,104,46
```

Here are the scripts:

makemodel.py

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from PIL import Image
import numpy as np
```

```
import pandas as pd
import os

img_dir = 'images/right'
img_data = []
point_data = []
img_w = 150
img_h = 400

lung_x = []
lung_y = []
neck_x = []
neck_y = []
throat_x = []
throat_y = []


def get_file_list(directory):
    for dir_path, dir_names, file_names in os.walk(directory):
        for file_name in file_names:
            yield open(os.path.join(dir_path,file_name),'r',encoding='utf-8')


def get_image_data(f_list):
    temp = []
    for f_path in f_list:
        img = Image.open(f_path.name)
        img = img.convert('L')
        img = img.resize((img_w, img_h), Image.ANTIALIAS)       #resize
        temp.append(np.array(img))

    #turn into 1 (non-white) and 0(white)
    for i in range(len(temp)):
        for j in range(len(temp[i])):
            for k in range(len(temp[i][j])):
                if temp[i][j][k] == 255:
                    temp[i][j][k] = 0
                else:
                    temp[i][j][k] = 1
    return temp


def get_all_points():
    dataframe = pd.read_csv('Points.csv', sep=',')
    dataset = dataframe.values

    temp = []
    points = dataset[:,:]
    for p in points:
        temp.append(p)
    return temp


def get_area_points(idx):
    temp = []
    for p in point_data:
        temp.append([p[idx]])
    temp = np.array(temp)
    return temp


def build_model(point_pos,model_name):
    model = Sequential()
    model.add(Conv2D(32, kernel_size=3, activation='relu', input_shape=(img_w,img_h,1)))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(64, kernel_size=3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(64, kernel_size=3, activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
```

```
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(loss='logcosh', optimizer='adam')
    model.fit(img_data, point_pos, epochs=100, batch_size=5)
    model.save(model_name)


f_list = get_file_list(img_dir)

img_data = get_image_data(f_list)
img_data = np.array(img_data).reshape(len(img_data), img_w, img_h, 1)   #to vectors

point_data = get_all_points()
point_data = np.array(point_data)

lung_x = get_area_points(0)
lung_y = get_area_points(1)
neck_x = get_area_points(2)
neck_y = get_area_points(3)
throat_x = get_area_points(4)
throat_y = get_area_points(5)

build_model(lung_x,'lung_x.h5')
build_model(lung_y,'lung_y.h5')
build_model(neck_x,'neck_x.h5')
build_model(neck_y,'neck_y.h5')
build_model(throat_x,'throat_x.h5')
build_model(throat_y,'throat_y.h5')
```

Explanation

The training images are saved in the following directory:

```
img_dir = 'images/right'
```

The variable *img_data* stores the pixel values of the training images. The function *get_image_data()* does the job. In the real world, lots of training images are required to build a sophisticated model. For testing purpose, this project only use 85 images. To reduce noise, the images are converted into black and white.

```
img_data = []
```

The variable *point_data* stores the pressure points that are saved in the file *points.csv*. The function *get_all_points()* does the job.

```
point_data = []
```

Using the function *get_area_points()*, the pressure points for the different foot's area () are stored in the following lists:

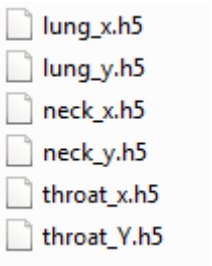
```
lung_x = []
lung_y = []
neck_x = []
neck_y = []
throat_x = []
throat_y = []
```

The function *build_model()* is the core of the program. Three conv2D layers are stacked together to extract the visual features of the images. As we are only interested to the numerical values from the output layer, there is no activation function in that layer. We use *logcosh* as the loss function. It performs much better compared to *mean squared error*.

```
model = Sequential()
model.add(Conv2D(32, kernel_size=3, activation='relu', input_shape=(img_w,img_h,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='logcosh', optimizer='adam')
model.fit(img_data, point_pos, epochs=100, batch_size=5)
model.save(model_name)
```

A number of models for the x and y positions of each of the pressure points are built using the above function.



predictor.py

```
from keras.models import load_model
from PIL import Image
import numpy as np
import cv2


img_data = []
img_w = 150
img_h = 400

lung_x = 0
lung_y = 0
neck_x = 0
neck_y = 0
neck_x = 0
throat_x = 0
throat_y = 0

file_input = 'user/input.png'
file_output = 'user/output.png'

def get_user_data(input_name):
    temp = []
    img = Image.open(input_name)
    img = img.convert('L')
    img = img.resize((img_w, img_h), Image.ANTIALIAS)
    temp = np.array(img)

    for i in range(len(temp)):
        for j in range(len(temp[i])):
            if temp[i][j] == 255:
                temp[i][j] = 0
            else:
                temp[i][j] = 1
    return temp


def predict_point(model_name):
    temp = 0
    model = load_model(model_name)
```

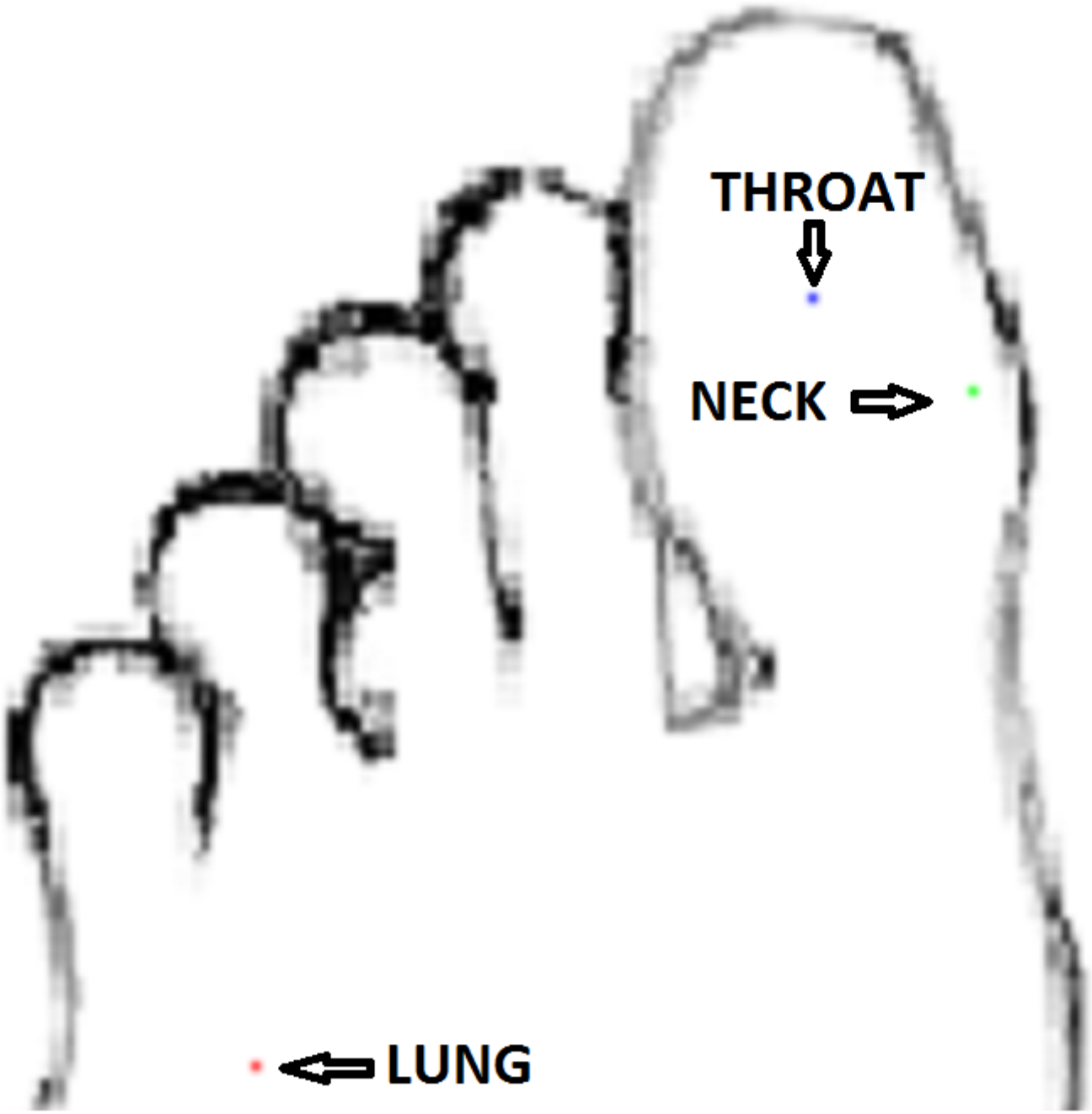
```
    temp = model.predict(img_data.reshape(-1, img_w, img_h, 1))
    return temp

#change to stepper motors' movements
def build_image(input_name,output_name):
    img = cv2.imread(input_name,1)
    img[int(lung_y),int(lung_x)] = [0,0,255]
    img[int(neck_y),int(neck_x)] = [0,255,0]
    img[int(throat_y),int(throat_x)] = [255,0,0]
    cv2.imwrite(output_name,img)


img_data = get_user_data(file_input)

lung_x = predict_point('lung_x.h5')
print(lung_x)
lung_y = predict_point('lung_y.h5')
print(lung_y)
neck_x = predict_point('neck_x.h5')
print(neck_x)
neck_y = predict_point('neck_y.h5')
print(neck_y)
throat_x = predict_point('throat_x.h5')
print(throat_x)
throat_y = predict_point('throat_y.h5')
print(throat_y)

build_image(file_input,file_output)
```

Explanation

The script *predictor.py* determines the user's pressure points based on his/her own foot's image and the previously built models. For simplicity, we do not perform image capturing. The image is saved as *input.png* . Store the image file in the directory named *user*. The output is saved as *output.png*. The function *build_image* creates the ouput image. Of course, the user data can also be converted into stepper motor's movements in order to build a real foot massager.

Here is the output:

```
[[40.528812]]
[[136.36086]]
[[125.51769]]
[[56.260376]]
[[106.0697]]
[[45.856754]]
```



As you can see, the output is not perfect. The network does not learn that well. However, it is something achievable. A better output can be produced using a deeper network, wider layers and more, quality training images.