Report of Project: Optical Character Recognition

Jiecheng Song(ID:111783762)
Ruijun Wu(ID:111776774)

December 10, 2017

Abstract

Our Project is about optical character recognition based on Python, a python code project can recognize the characters in a picture. In the project, we did some operations to matrix and processes to image. And we used support vector machine to classify the matrices of pictures and output the characters in the pictures.

1 Introduction

This part is divided into two sections: brief introduction of OCR, project obejectives.

1.1 Brief introduction of OCR

The optical character recognition(OCR) is widely used and applied in our lives. Its main idea is to convert typed, handwritten or printed text on an image to machine-encoded text. There are many practical applications: to extract the numbers on a bank card, to extract information from one's passport documents and to recognize plate number, etc. So its results can be very impressive if it can work perfectly and obtain whatever the fonts are and how complicated the background is. In short, OCR is a complexation of pattern recognition, artificial intelligence and computer vision.

In the beginning, we choose this project just based on our interests. It seems that OCR works very smoothly and efficiently. However, after reading some paper, we realize that we may face a lot of difficulties implementing the algorithms and many functions entirely new for us. For example, it is hard to detect the boundary between the text and the background. Thus, in practice, we just adjusted our goal based on our knowledge and time appropriately, which makes it won't be either too high to achieve nor too easy to accomplish.

1.2 Project objectives

Our main objectives are to detect where the words are and convert them into machine-encoded text properly. As mentioned in first part, we mainly deal with English words and some punctuations frequently used in a monochrome background. To implement our goal, we need to write all algorithms and every procedure in OCR implementation step by step, which requires we need to have a clear mind and logistics about the whole project. Also, we want to improve our skills in programming and learn more knowledge from this project.

Also, we manage to extract the characters with more than one colour and try to deal with a complicated background.

2 Techniques and tools

2.1 Methods

2.1.1 Gray Processing

Due to most of pictures are in RGB mode, the picture we read in would be a 3-D matrix. Each point of the picture includes 3 values corresponding Red, Green and Blue. But it is difficult to analysis with 3-D matrix, so we first gray process the picture with the formula $Gray_{value} = 0.30 * Red_{value} + 0.59 * Green_{value} + 0.11 * Blue_{value}$, then we transfer the matrix into 2-D, which is much easier to analyze.

2.1.2 Binarization

To distinguish the background and characters better, we did binarization to the picture, transfer the matrix into two values, with the background into 0, and the character into 1. Since in our case, there are only two colors in the picture, and always the background is white with value 255, while the colorful characters with value less than 255, so the mean value of total values are less than 255 but greater than the value of characters. As a result, we transfer the value which is over the mean of total values to 0 and below mean to 1.

2.1.3 Line Cut

We cut the picture into lines. In the picture, if the whole line is all white, then the corresponding row in the matrix should be all zeros. So we calculate the sum of each line, if it is 0, then the points in this row are all white. We recorded the row of changing(from 0 to 1 means the start of one line of words, and from 1 to 0 means the end of one line). And we pick out the matrix by line from each start to end. Then we get the each word line.

2.1.4 Column Cut

We cut the picture into characters. In each line of words, if the whole column is all white, then the corresponding column in the matrix is all zeros. So we calculate the sum of each column, if it is 0, then the points in this column are all white. We recorded the column of changing(from 0 to 1 means the start of a character, and from 1 to 0 means the end of a character). And we pick out the the matrix by columns from each start to end. Then we get the each character.

And if the distance between the end of last character with the start of next character is over than 0.4*the height of the line, we recorded it as a space with a 10*10 identity matrix(can be distinguished with other character) and recorded a upper triangular matrix with all non-zero elements are 1 as the sign of the new line.

2.1.5 Character Cut, Noise clear and Resize

We cut the background in each picture of characters, making there is no blank in the picture. If there is less than one point in the neighbour of the point, we thought it is a noise point and will be cleared. Then we resize the matrix into a 10*10 matrix, and treat each character as a point in the space of 10*10 matrix.

2.1.6 Support vector matrix

Support vector machines, also called SVM, is an efficient algorithm with supervised learning model that can be used to analyze data for classification.

With a base of training data, each signed as belonging to one or the other category (two in total), we can use SVM training algorithm to build a model that can classify a new data to one category or the other.

In an SVM algorithm, we consider the data as points in space, with the algorithm a boundary will be found to the separate the two categories of points, and the categories are divided by a clear gap that is as wide as possible, as a result, the prediction will make less error. Then the new data points are mapped into that space and considered to belong to the category which represents the side it mapped.

In SVM algorithm, we usually use kernel function to map the points into a higher dimension space, which helps us to find the boundary and gap more easily.

In our project, we treat each picture of character we get as a point in the space of 10*10 matrix, and use SVM training algorithm to build a model to classify the different character. So we can predict the picture we input with the model.

We train our model with a picture with all capital and lower characters and 4 frequently used punctuations with a space between each pair of characters. So there are 58 groups in our training data, "0" represents the space, "1" to "26" represent the capital characters, "27" to "52" represent the lower characters, and "53" to "56" represent four punctuations ".","," ,"!","?". The last represent the sign of new line. And we choose linear kernel function to transfer our data, also due to there are more than two groups in our training data, we choose "ovo" model, means build models between each pair of groups ($\frac{58*57}{2}$ total), and classify the new data by each model.Then we considered the data belonging to the group which be classified most.

2.2 Problem and Improvement

2.2.1 Mistake in "l","I" and "."

Since the matrix of "l","I" and "." are similar after resizing(almost all element are ones),so if a character was predicted into one of these 3 categories. We need to calculate the rate of height and width of column cut picture to classify them. If the height is less than 4*width, it is not a narrow picture, and it is more likely to be a ".". Due to the "I" usually appear in the first of a sentence or word, if the second character before it is a punctuation, we consider it is "I". The other we consider it as an "l".

Figure 1: "l" after resize

Figure 2: "I" after resize

Figure 3: "." after resize

After resizing, they look similar(all the elements are the same or similar).

Figure 4: “l” before background cut

Figure 5: “I” before background cut

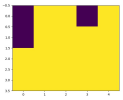


Figure 6: “.” before background cut

We can distinguish the “.” from “l” and “I” by calculating the rate of the height and width.

2.2.2 Mistake in Similarity of Capital and Lower

Some characters with similar capital and lower format may cause mistakes, like “s” and “S”, “c” and “C”, “u” and “U” and so on. So if the picture was predicted into these groups, we need to find if there are some blank in the picture before cutting background, since there should not be a blank in capital characters.

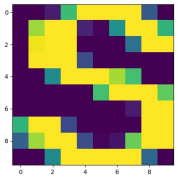


Figure 7: capital “S” after resize

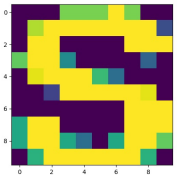


Figure 8: lower “s” after resize

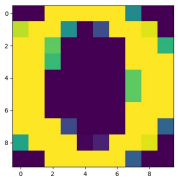


Figure 9: capital “O” after resize

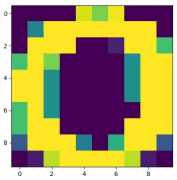


Figure 10: lower “o” after resize

We cannot distinguish the capital and lower format very well with resized matrix.

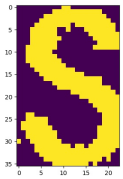


Figure 11: capital “S” before background cut

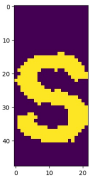


Figure 12: lower “s” before background cut

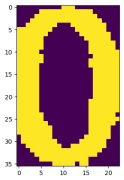


Figure 13: capital “O” before background cut

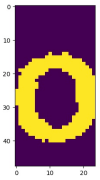


Figure 14: lower “o” before background cut

There is some blank on the top and at the bottom of lower characters but there is not in capital characters, so that we can distinguish them.

2.2.3 Mistake in “l” and “i”

Since the matrix of “l” and “I” are similar after resizing,so if the character was predicted into one of these 2 groups, we test if any line in the middle of the picture is all white. If there is, the character is “i”, if not, it is “l”.

Figure 15: capital “l” after background cut

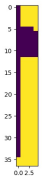


Figure 16: lower “i” after background cut

There is no blank in the middle in “l” but there is blank in “i”, so we can distinguish them.

2.3 Tools

Our project is based on python, we use some operations of matrix, processes of picture and support vector machine algorithm.

Table 1: Packages and Functions

Packages	Functions
Numpy	operations to matrix
Matplotlib	plot figure
Image	read and process image
Os	process files
Cv2	use opencv to process image
Math	some math calculation
Sklearn	machine learning and support vector machine

2.4 New skills we learnt

In our project, we developed our ability in python, such as list process, for loop and functions. Also, we used some packages mentioned before to process image and matrices, as well as some

algorithms in machine learning. After this program, we improved our coding skills a lot.

What's more, we have read some paper about optical character recognition, support vector machine and some other altorithms in machin learning, which deepens our understanding in machine learning methods.

3 Conclusion

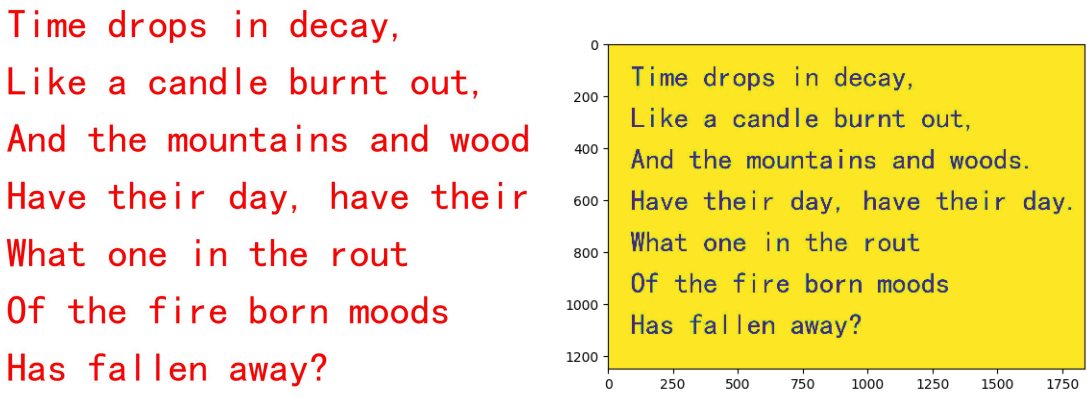
In this part, we mainly show how our code works, what we have done so far and the improvments in our project.

3.1 User Guide

First, excute the code, and you will be asked to input the file name. Then you should input the whole name of the picture you want to recognize, like "time.jpg"(the pictures must be in the same document as where the code's in). Finally, the characters in the picture will be output.

3.2 Performance of our code

First, we convert image figure17(a) to a matrix (figure17(b)) which represnts every point's greay scale value.

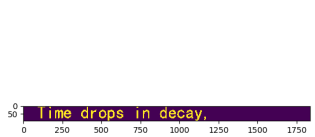


(a) The input image

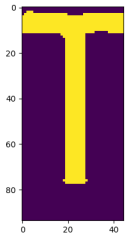
(b) The output matrix

Figure 17: Convert image to gray scale matrix

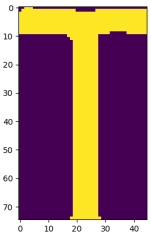
Next, we cut the backgrounds off by rows(figure18(a)) and by columns(figure18(b)). Finally, we can get the single matrix of every character(figure18(c)).We take character "T" as an example.



(a) The matrix after cutting rows



(b) The matrix after cutting columns



(c) The matrix of a character

Figure 18: Three steps to get the matrix of a character

Next, as mentioned in the method we need to resize the matrix into the same size in order to classify them using SVM algorithm. The resized matrix is shown below:

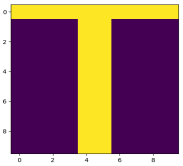


Figure 19: The resized matrix of “T”

After doing all the steps above, every character’s matrix can be recogized and converted to text in computer. And the final reslut is shown below:

```
>>>
 RESTART: C:\2017-2018 fall\AMS595 computation foundation\Project\project.py
Input the file name:time1.jpg
 Time drops in decay!
Like a candle burnt out.
And  the  mountains  and  woods.
Have their day!  have their day.
What  one in  the  rout
Of  the  fire  born  moods
Has fallen away?
```

Figure 20: The output of figure (17(a))

3.3 Improvement of our project and shortages

In this section, it will show another impressive result of a picture with green characters, a conclusion of our improvements and some shortages.

This is another case,when we input an figure(21):

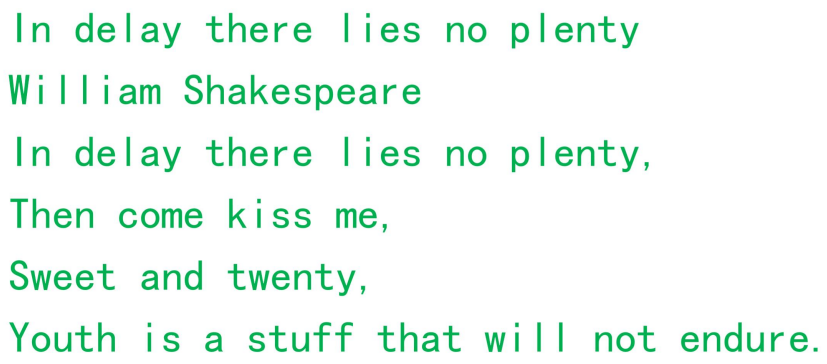


Figure 21: The picture with green characters

The text which are extracted from it are:

```
>>>
 RESTART: C:\2017-2018 fall\AMS595 computation foundation\Project\project.py
Input the file name:shakespeare.jpg
 In delay there lies no plenty
Wiliam Shakespeare
ln delay there lies no plenty.
Then come kiss me.
Sweet and twenty.
Youth is  a  stuff that will not  endure.
```

Figure 22: The picture with green characters

As shown above, we can see our code can recognize them properly. In practice, we keep modifying our code and solve many problems described in details in Chapter2.2. For example, at first, our code can't split blank space and recognize where the new line is. But after inserting a new matrix(figure23) representing the beginning of a new line and some commands, the code works now(method used explained in Chapter2.1.4).

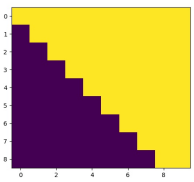


Figure 23: matrix of new line

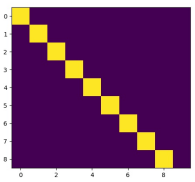


Figure 24: matrix of blank space

Actually, in practice, our results are consistent with the input, with the accuracy of about 95%.

But there are still some issues we can't solve now. The main challenge in our project is how to recognize more fonts of English and so far we can recognize at most two fonts. But the reason of this is that the algorithms we use is not very efficient and it can only classify the specific fonts one time due to the train set we give. And we don't have enough time to implement all of them.

So in the future, we want to implement another high efficiency algorithm such as neural networks to do the classification.

4 Reference

C. Cortes and V. Vapnik, Support-Vector Networks, Machine Learning, 20(3):273-297, September 1995.

Eikvil L. Optical character recognition[J]. citeseer. ist. psu. edu/142042. html, 1993.

Mithe R, Indalkar S, Divekar N. Optical character recognition[J]. International Journal of Recent Technology and Engineering, 2013, 2(1): 72-75.

Suykens J A K, Vandewalle J. Least squares support vector machine classifiers[J]. Neural processing letters, 1999, 9(3): 293-300.

Tong S, Koller D. Support vector machine active learning with applications to text classification[J]. Journal of machine learning research, 2001, 2(Nov): 45-66.