Fall 12-20-2016

Real-time Online Chinese Character Recognition

Wenlong Zhang
San Jose State University

Real-time Online Chinese Character Recognition

Wenlong Zhang

Computer Science Department

San Jose State University

San Jose, CA 95192

408-924-1000

December, 2016

The Designated Project Committee Approves the Project Titled

Real-time Online Chinese Character Recognition

by

Wenlong Zhang

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2016

Dr. Robert Chun	Department of Computer Science
Dr. Katerina Potika	Department of Computer Science
Ezekiel Calubaquib	Software Engineer in Cohesity

Abstract

In this project, I am going to build a web application for handwritten Chinese characters recognition in real time, which means this system will determine a Chinese character while user is drawing/writing. While utilizing different tools to build the app, the techniques I use to build the recognition system including data preparation, preprocessing, features extraction, and classifying. To increase the accuracy, two different types of neural networks will be used in the system: multi-layer neural network and convolutional neural network.

Contents

1 INTRODUCTION

The 21st century is often called “the century of information.” As innovation and knowledge increases, the information era continues to advance. In China, more and more people are beginning to use computers in different aspects of their lives, including simple daily tasks, work, leisure and entertainment. Given the diverse range of education in the Chinese speaking community, the first problem is how to facilitate entering Chinese characters using a standard alphanumeric keyboard, which directly affects the application and development of processing Chinese character information on the computer.

"Pinyin is the official romanization system for Standard Chinese. It includes four diacritics denoting tones. Pinyin without tone marks is used to spell Chinese names and words in languages written with the Latin alphabet, and also in certain computer input methods to enter Chinese characters." [1] Everyone struggles when entering Chinese characters on a computer; people who utilize the Pinyin input method, often misspell the intended word, at least on the first try. Even though other input methods that deconstruct characters into a series of strokes can be used, this requires the user to be knowledgeable about the structure of Chinese characters. In light of this situation, the “handwriting style” input method, in which a user writes characters directly onto a tablet, has become more and more popular. As long as the user writes the Chinese characters on an electronic “writing pad” (as

they would on normal paper), the computer will identify characters by capturing locus points of the handwriting.

In terms of discipline, Chinese character recognition can be categorized as pattern recognition and image processing, but also involves artificial intelligence, formal language and automata, statistical decision theory, computer science and other disciplines. Therefore, it is a comprehensive technical science. Due to the large number of Chinese characters, complex character structure, and word similarity, Chinese character recognition is more difficult than character recognition of other languages. With the advances in pattern recognition and computer science technology in recent years, Chinese character recognition has made a great deal of progress.

Collecting handwriting input can be done in two different ways: online and offline. “Online handwriting involves the text as it is written on a special digitizer or PDA, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. This kind of data is known as digital ink and can be regarded as a digital representation of handwriting.” [2] The basic unit of Chinese character recognition is a single stroke. The combination of strokes, the stroke location, and the relationship between strokes are used to identify characters. “Offline handwriting involves the text in an image which are usable within computer and text-processing applications. The data obtained by this form is regarded as a static representation of handwriting” [2] Because of the arbitrariness of human handwriting, the difficulty of handwriting Chinese character recognition is much higher than printed

Chinese character recognition. “Both online and offline recognition methods can be roughly divided into two categories: structural and un-structural.” [3]

The basic principle of handwritten Chinese character recognition is processing pattern matching between the input characters and each standard character. The basic process involves calculating the degree of similarity and then returning the standard character with the greatest degree of similarity. However, human handwriting is arbitrary, and cursive writing combining multiple strokes into one often occurs. Also, the text will sometimes overlap with background. Thus, the preprocessing stage is needed to recognize handwritten characters. In this stage, the main goal is to achieve the normalization of the Chinese character’s image, which normalizes the character’s size and corrects the distortion of the handwritten input. The program then proceeds to the character recognition stage, which will complete the identification of the Chinese character. This involves feature extraction and classification design. Lastly, the final phase is post-processing. The program will make use of Chinese character structures, the semantics, the meaning, and other contextual information to process error correction and finalize it. Even though characters are processed using three stages, the error rate of the character recognition is still high for even slightly illegible characters.

In the early stage of character recognition research, some scientists underestimated the problems and difficulties that may actually occur. At the time, many people thought the shapes of the ten Arabic digitals, zero to nine, were very

simple and correctly identifying them would not be difficult. However, in practice, although there are only ten digits and their glyphs are not complicated, it is really hard to ensure a 100% correct recognition rate because so many different writing styles exist. If even a highly educated person cannot recognize some especially lazy handwritten numbers, then there are surely some that a computer cannot comprehend.

Online handwritten Chinese character recognition also suffers from similar problems. In the beginning of the study of automatic character recognition, some people thought that online handwritten Chinese character recognition should be easier than the printed Chinese character recognition. This is because the latter's recognition target is two-dimensional graphics and overlapping strokes are not easily separated. When doing online handwriting recognition, the user is writing characters on a board. The strokes of Chinese characters will be separated and imported into the computer one by one, which forms a one-dimensional string of strokes. As long as all types of strokes and mutual relations are correctly determined, that single word can be correctly identified as well. In theory, this perspective is correct, but it's very difficult to achieve in practice due to the volume of Chinese characters, the complex shape, and the orders of strokes and cursive writing when people are writing Chinese characters. Therefore, these affect recognition rates greatly. Let's discuss them separately below.

a. Variation in the number of strokes

The forms of strokes of Chinese characters also have specifications. For example, the character "厂" has two strokes: "一" and "丿", which cannot be finished in one stroke. Another example, the character "美" should be divided two parts. One is "羊" and the other is "大". However, some people will consider it as "竺" and "大". These are all common sense, but the educational level and writing habits of people are in endless variety. As a result, to require everyone to conform to a standard stroke form will not be easy.

b. Stroke trend

Writing direction of strokes of Chinese characters are mostly from left to right or top to bottom. But there are some exceptions, such as the first stroke on the top of "手", "受", and "系" should be from right to left. But some people tend to write from left to right and it becomes horizontal "一" so that stroke string will change thus resulting in incorrect recognition. There're easier problems that occur, like the word "天". Its first stroke is similar as the previous example. If the first stroke is written as "一", it will become "天". There are quite a lot of these kinds of words in Chinese characters, such as "禾" and "未", "壬" and "王" and so on.

c. Stroke order

The order of strokes is an important factor affecting the performance of online handwriting Chinese character recognition. For example, take the character "女". Some people will write "く" first, and others will write "一" first. With different order, a word can have two different stroke strings. Some words are composed of a

couple of parts. If people write in different orders, the change in stroke strings is even larger. For instance, the word “送” has two parts: “辶” and “关”. Some people write “辶” first; on the contrary for other people, “关” will be written first. Another example of the word “因”. The standard writing will be “冂” first, then inside, “大”. Finally, close the box at the bottom with “一”. However, some people will just write a big box, “口”, first, and then construct the inside, “大”. These principles seem simple, but the writing habits of millions of people are not uniform. Obviously, this stroke sequence problem will inevitably impact online handwriting Chinese character recognition.

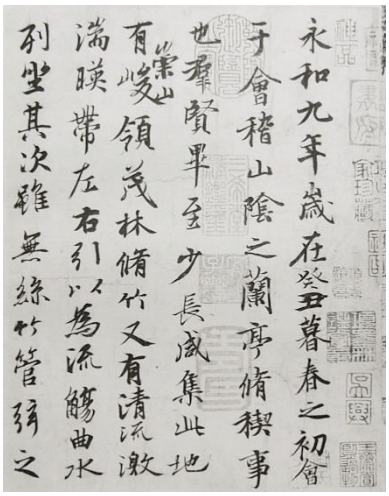
d. Cursive writing

Cursive writing is an important factor that affects online handwritten Chinese character recognition. Perhaps this is the most difficult factor to overcome. In order to speed up writing, people often write cursively. The higher the education level, the more obvious the cursive writing problem becomes. This is also a reason why raising the recognition rate of the majority of online handwritten Chinese character recognition systems is so difficult, even more than solving the stroke order problem.

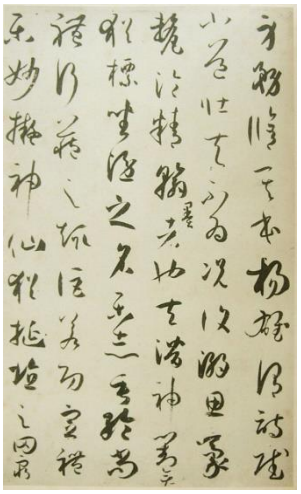
There are three types of handwriting Chinese calligraphies: regular script, semi-cursive script, and cursive script, shown in Figure 1. In Regular script, vertical and horizontal strokes are straight and bodies are upright. On the other hand, cursive script is very scratchy and some words are almost finished within one

stroke. There was a great calligrapher in Jin Dynasty and almost no one can recognize some of his writing. The semi-cursive script is between last two scripts. The strokes are not completely straight, with larger arcs, and are often written cursively. For instance, the character "口" originally has three strokes, which will be written like "𠃌". This does not only change the number of strokes, but also the type of stroke is not the same. These situations are numerous and are the main factor of performance.

Regular Script



Semi-cursive Script



Cursive Script

FIGURE 1: DIFFERENT STYLES OF SCRIPTS

There are two ways to solve the above problem. The first way is restricting the way users write. We could require people to write in regular, proper script. Strokes have to be straight and smooth. Stroke order has to follow the standard. No cursive writing is allowed. According to these requirements, writing characters in this style is usually called "restricted handwriting." If people write characters

according to these provisions, the recognition rate can be very high. Unfortunately, it is hard to change people's writing habits and styles. Making such excessive demands is not only difficult to do, but also does not conform to the principle of user-friendly interface. Therefore, another way to solve the above problem — the appropriate method to solve the problem—is to make an effort to improve the character recognition software.

2 EXISTING SOLUTIONS

2.1 PREPARATION

Handwritten Chinese character recognition is divided into online and offline. Online is usually based on strokes and the data processed is a series of point coordinates. On the other hand, offline is usually based on images that are typically a bitmap containing pixel values. Both include three stages: preprocessing, feature extraction, and classification. However, the actual methods that will be used in each stage are different and these particular methods can also be used in multiple different stages. Several common methods will be described below.

Preprocessing plays an important role in handwritten character recognition. Most of the time, handwriting is captured in an unconstrained environment. Writing styles and habits are not standardized. As a result, these handwritten characters that are collected will have these qualities: font sizes and shapes are different, strokes have different degrees of distortion, and characters are missing strokes. If these characters in their initial state are directly used for character recognition, it will cause large errors. Therefore, the primary purpose of preprocessing is to reduce similar transformation, which ultimately improves the recognition rate.

Image binarization is a process that can usually be used in offline handwriting recognition. An image is actually a two-dimensional matrix and the values in the matrix correspond to the pixel in that position. Binarization will

change all values of the matrix to either zero or one. Zero represents black and one represents white. Reversing this is acceptable, and depends on the actual circumstances. So, each pixel of such image can be represented as a binary number. This can greatly reduce the storage space needed for processing the character image. Now, fast bit operations can be used to process these images, saving a lot of computing time.

"Normalization is considered to be the most important pre-processing factor for character recognition. Normally, the character image is linearly mapped onto a standard plane by interpolation/extrapolation. The size and position of character is controlled such that the x/y dimensions of normalized plane are filled." [4] Based on the relationship between the horizontal and vertical coordinates, normalization can be one dimension or two dimensions. One dimensional normalization means all points in the same row or same column still remain in the same row or column after normalization. According to the characteristics of the mapping function, normalization methods are generally divided into linear normalization, moment normalization, and nonlinear normalization. According to the mutual influence between the horizontal coordinates and vertical coordinates, it produces a pseudo two dimensional normalization.

2.2 BACKPROPAGATION NEURAL NETWORK

Backpropagation (BP) is a very efficient algorithm, which was proposed by Paul Werbos in 1986. [5] It is designed to arrange neurons in layers: one input layer, one or more hidden layers, and one output layer. "Each node from input layer is connected to a node from hidden layer and every node from hidden layer is connected to a node in output layer. There is usually some weight associated with

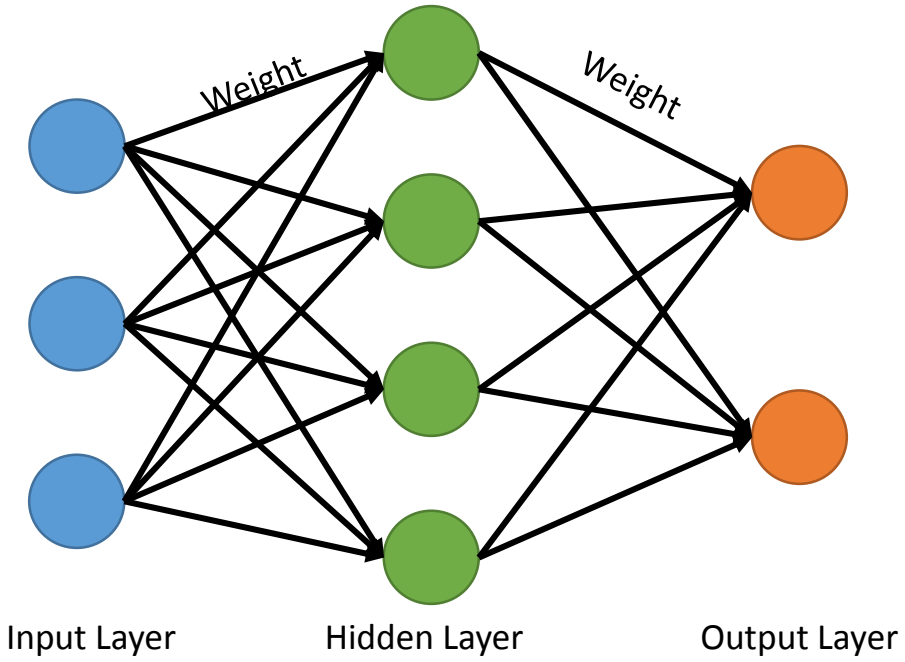


FIGURE 2: NEURAL NETWORK

every connection." [5] Figure 2 is a simple neural network. The neurons in each layer only accept the input from other neurons in the previous layer. The output layer is an outcome of the input layer after one goes through all hidden layers. Because of this design, the backpropagation neural network model solves problems that other basic models cannot solve. It will change the I/O problem of a set of sample data into a nonlinear optimization problem. By using a gradient descent

method and an iterative algorithm to complete learning and memorization, this problem is solved. The increase of neurons in hidden layers can increase the adjustable coefficient, allowing problems to be solved more accurately.

The backpropagation algorithm trains the multilayer neural network, and each time the training example will be calculated twice during delivery. First, feed forward computation, which begins at the input layer, is followed by passing all the layers. After the whole neural network is traversed, an output is produced. At this time, the error is calculated between the actual output and the desired output. Second, backpropagation computation, which is in the opposite direction, starts from the output layer back to the input layer. During this process, the error is used to adjust the weights layer by layer until the final result is approached.

Thus, "since this method requires computation of the gradient of the error function at each iteration step, we must guarantee the continuity and differentiability of the error function. Obviously we have to use a kind of activation function other than the step function used in perceptrons, because the composite function produced by interconnected perceptrons is discontinuous, and therefore the error function too." [5] The most popular activation function is sigmoid:

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

This equation will create a nice S curve between 0 and 1. The derivative of sigmoid is

$$\frac{d\varphi}{dz}(z) = \varphi(z)(1 - \varphi(z))$$

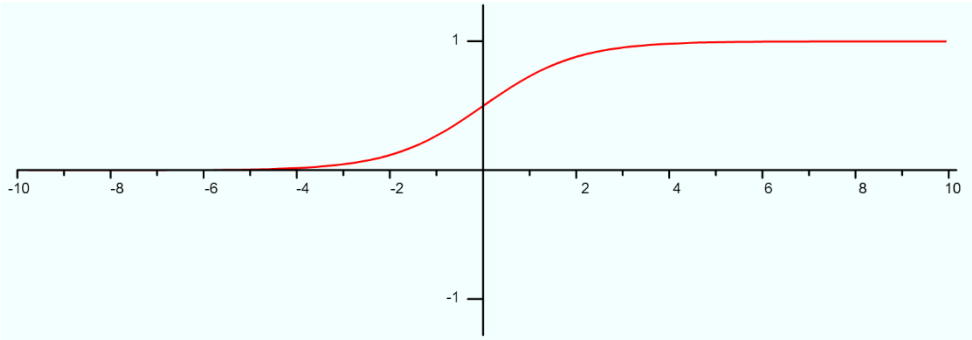


FIGURE 3: SIGMOID EQUATION GRAPHING

"The way to control the Neural Network is by setting and adjusting the weights between nodes. Initial weights are usually set at some random numbers and then they are adjusted during Neural Network training." [6] The weights are usually set to a smaller number in order to avoid saturation or anomalies in the network if the weights reach the maximum.

According to the requirements of BP algorithm, it is divided into the following steps. First, take one data from the training data set as an input vector for the network. Second, calculate the output vector through the algorithm with that input vector. Third, calculate the difference of error between the output vector and the target vector. Fourth, go over the algorithm in reverse from the output layer to first hidden layer and adjust each weight based on the result in order to reduce the error. Last, repeat steps one to four for each data in the training data set until we get the minimum error.

The first two steps in the algorithm will be repeated during recognition even after the neural network is completely trained. Training will stop when the error between actual output and expected output is low enough, which is usually determined by the programmers. At this time, all weights are fixed and won't change any more.

2.3 Convolutional Neural Network

Convolution neural network (CNN) is one kind of artificial neural network, which has become a hotspot in the field of image recognition. Its weights sharing network structure makes it more similar to the biological neural network, which reduces the complexity of the network model and the number of weights. The advantage of such effect will be more obvious when the inputs are multidimensional images. These images can be directly the input for the network, which avoids the complex feature extraction and data reconstruction in the traditional recognition algorithm. CNN is a multilayer perceptron that specially designed for two-dimensional shape recognition.

The difference between CNN and ordinary neural network is that CNN consists of convolutional layers and pooling layers. In convolutional layer, one neuron is only connected with some neighboring neurons rather than fully connected in normal neural network. So, it usually contains several feature maps that compose of a number of rectangular array of neurons. The neurons in the same feature map share weights, where the weights form a filter or kernel. The kernel is

initialized in the form of random fractional matrix and will get reasonable weights during the training process. The direct benefits of shared weights (kernels) is to reduce the connection between each layers in the network while reducing the risk of over fitting. Pooling usually has two forms: mean pooling and max pooling. Obviously, mean pooling will take the mean value of an area and max pooling will take the largest number of an area. Pooling can be seen as a special convolution process. Convolution and pooling greatly simplify the complexity of the model and reduce the number of parameters in the model.

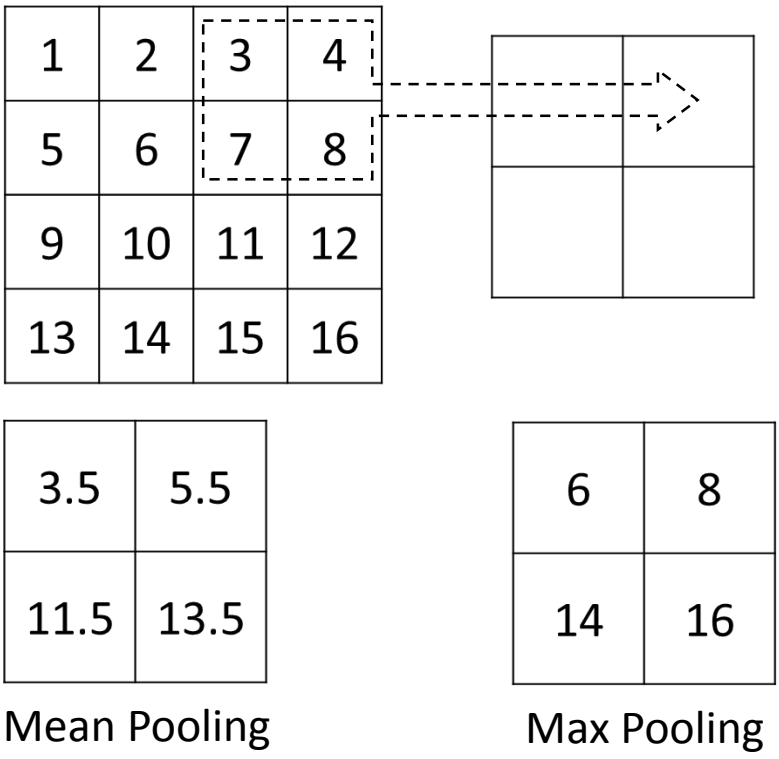


FIGURE 4: POOLING LAYER

Convolutional neural network usually uses a number of pairs of convolutional layers and pooling layers, which forms a stack structure as a feature extractor. They continually reduce the size of feature map, but the quantity tends to be increased. A classifier is right after the feature extractor, which is usually composed of a multi-layers perceptron that one layer is fully connected to next layer. Figure 5 is a typically convolutional neural network structure.

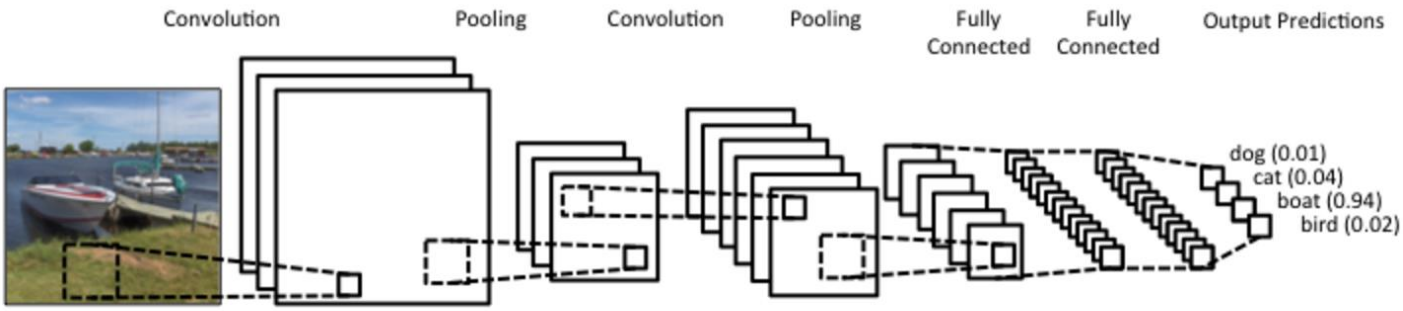


FIGURE 5: ARCHITECTURE OF CONVOLUTIONAL NEURAL NETWORK *[25]*

2.4 QUICKSTROKE

Writing Chinese characters with the use of the keyboard is not only difficult but inconvenient. Simplified Chinese character symbols can be classified by up to 4400 ideograms, which is non-trivial compared to the 26-character English alphabet and 10-character Arabic numeral. “An ideal input device for ideographic text would use on-line handwritten input. Common touch-sensitive input devices like TouchPads, tablets, or PDAs are all capable of capturing such on-line handwriting data in the form of pen or finger trajectories.” [7] Many researchers and development teams have gained ground toward improving writing and recognizing Chinese characters. QuickStroke which is an incremental recognition system for

"printed and partially cursive Chinese characters" [7] is not only fast, but also more accurate than other similar systems. An article by Nada P. Matic, John C. Platt, and Tony Wang titled "QuickStroke: An Incremental On-Line Chinese Handwriting Recognition System" explains the methodology and results of why QuickStroke is an improved system for recognizing Chinese ideograms.

QuickStroke first does a coarse classification by initiating the pre-classifier. This pre-classifier shrinks the set of character candidates to a smaller group. This would make recognizing characters faster and efficient, similar to parsing only what is needed. QuickStroke has 33 groups which come from the 4400 GB classes of simplified Chinese characters. The pre-classifier uses these groups to recognize the first 3 strokes given to it. "We choose an initial set of classes for each group based on the similarity of these first three strokes. We then train a first prototype of the pre-classifier on this limited set of classes and use bootstrapping to label all available training data into groups." [7] Figure 6 shows the user interface of QuickStroke, which recognizes a character within a few strokes. In addition, many variations exist when it comes to writing, such as when people write characters in different order. In order to accommodate this, QuickStroke allows the 33 groups to

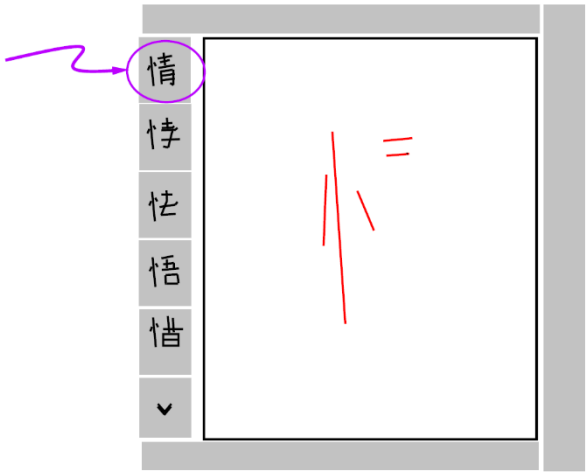


FIGURE 6: THE USER INTERFACE OF QUICKSTROKE *[7]*

overlap which allows each character to be in more than one group. “For example, the first three strokes of a particular character can be written using different stroke order by different writers, which in turn leads to one variant of a class belonging to one group and another variant belonging to an alternate group.” [7]

QuickStroke then does a detailed classification by using pairs of neural networks for each of the 33 groups. The result is that each detail classifier is a vector of n probabilities based on the user’s input. These two processes are also efficient as the group’s identity is static and do not change. Thus, more inputs from the users simplify the algorithm by not calling the pre-classifier again.

From this article, the result from the test using QuickStroke included twenty writers to be used for training and validation. Partial character tests will be three or more strokes and complete character tests will be done with whole characters. Partial character accuracy reading for the 4400 GB classes of simplified Chinese characters for top one characters resulted in a 97.3% accuracy. Complete character accuracy reading came to around 96.3% accuracy. These show that QuickStroke is precise for both partial and complete character inputs. “Our testing results indicate that, on average, only half of the total number of input strokes need to be entered in order for the system to recognize the character (i.e., 6 strokes out of an average of 12). Thus, users of QuickStroke can enter characters much faster than alternative input methods.” [7]

3 PROPOSED SOLUTION

3.1 DATASET

The dataset I'm going to utilize is "a pair of online and offline Chinese handwriting databases built by the Institute of Automation of Chinese Academy of Sciences (CASIA). The handwritten samples were produced by 1,020 writers using Anoto pen on papers and include both isolated characters and handwritten texts (continuous scripts)." [8] They provide six datasets of online and offline data: half for isolated characters and half for handwriting texts. However, "the total number of Chinese characters is very large, e.g., the standard set GB18030-2000 contains 27,533 characters, which are not yet exhausted. We estimate that the number of characters used by most people day to day is about 5,000, which is almost the maximum that ordinary educated people can recognize." [8] Some characters that

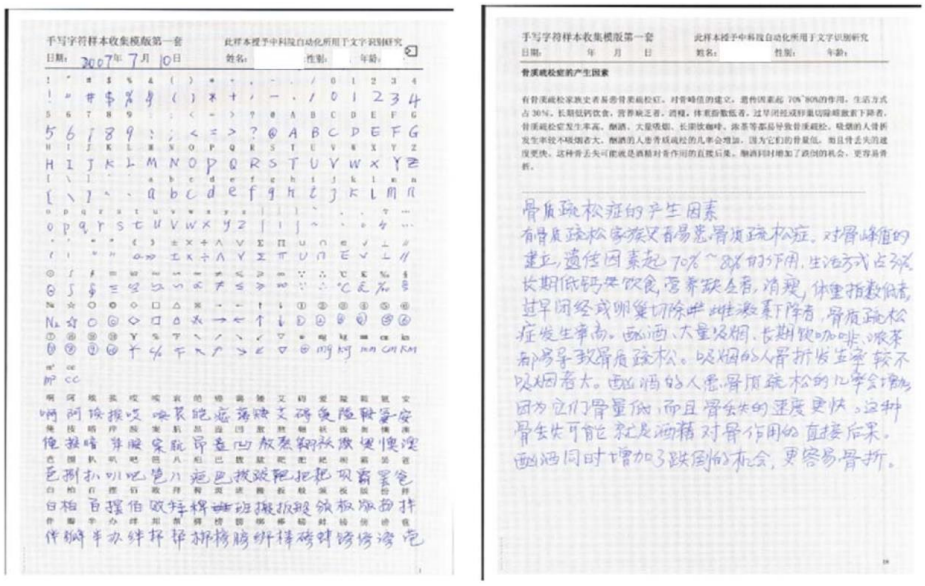


FIGURE 7: SCANNED PAGES OF ISOLATED CHARACTERS AND HANDWRITING

are not used often were filtered out. The resulting dataset contains 3755 level 1 and 3008 level 2 Chinese characters, 52 English letters, 10 Arabic digits, and some other frequently used symbols.

3.2 IDEA

I will build three types of neural networks:

1. Strokes: use coordinates to determine what stroke it is
2. Direction: use strokes' coordinates to determine the writing direction
3. Image: use the whole image to determine character

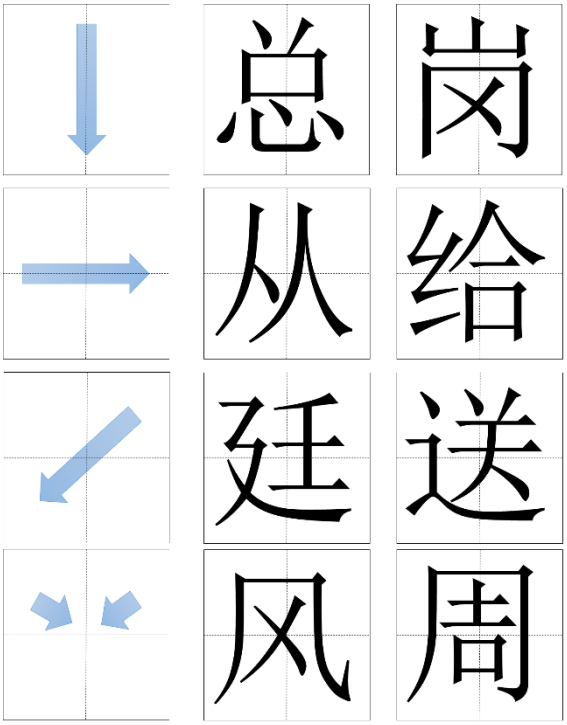


FIGURE 8: WRITING DIRECTION

First, classify all Chinese characters into four classes depending on writing direction.

1. From top to bottom.
2. From left to right.
3. From upper right to lower left.
4. From outside to inside.

Each class will have its own neural network which is trained within each data set. If the result is not accurate enough, I will try to create more classes based on Chinese character components within each writing direction class.

The workflow will be similar to the following: first, "pen" down, users start to write a character and "pen" up, users finish writing. At this time, the character can be either partial or full. Second, the program will determine the writing direction according to the input coordinates. Third, pass the input to the specific writing direction neural network to recognize character. At the same time, an image of the input will be used in the image neural network.

3.3 TOOLS

3.3.1 Python

I am going to write programs for preprocessing, classification, and neural networks in Python. Python is a very powerful programming language and contains tons of extensions and libraries. For example, "NumPy is an extension to the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays." [9] I can use it to do some complex calculations such as multiplication of multi-dimensional arrays and the sigmoid function. Therefore, I can build a simple neural network with NumPy. However, if I don't want to construct the neural network by myself, there are still many libraries for it (e.g. scikit-neuralnetwork, PyBrain, Lasagne, Blocks, TensorFlow, Keras, Deepy).

3.3.2 XAMPP

"XAMPP is a free and open source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. XAMPP stands for Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P)…… XAMPP also provides support for creating and manipulating databases in MariaDB and SQLite among others." [10] This will be used for building the web application which is the user interface.

3.3.3 jSignature

"jSignature is a JavaScript widget (a jQuery plugin) that simplifies creation of a signature capture field in a browser window, allowing a user to draw a signature using mouse, pen, or finger." [11] The reason why I chose this tool is because of its features – "[it] captures the signature as vector outlines of the strokes. Although jSignature can export great bitmap (PNG) too, extraction of highly scalable stroke movement coordinates (aka vector image) of the signature allows much greater flexibility of signature rendering." [11] I can use these stroke movement coordinates to detect users' writing directions and type of strokes. Then, I can construct them as an input for the neural network to test.

3.3.4 Keras

Keras is a third-party neural network library that is extremely simplified and highly modularized. It gives full play to the operation of GPU and CPU based on the development of Python and Theano (a numerical computation library for python.

The purpose of its development is to quickly have neural network experiments. “We can easily construct both sequence-based networks (where the inputs flow linearly through the network) and graph-based networks (where inputs can “skip” certain layers, only to be concatenated later). This makes implementing more complex network architectures much easier.” [12]

4 IMPLEMENTATION

4.1 DATASET PREPARATION

In this research, the online characters dataset (OLHWDB1.1) from CASIA will utilized. These data will be stored in .pot format files. Table 1 shows the structure of each character in a file. After decoded, each file will generate 3755 common Chinese characters.

TABLE 1: FORMAT OF ONLINE ISOLATED CHARACTER DATA FILE (*.POT)

http://www.nlpr.ia.ac.cn/databases/handwriting/Online_database.html

Item	Type	Length	Instance	Comment
Sample size	unsigned short	2B		Number of bytes for one sample (byte count to next sample)
Tag code (GB)	DWORD	4B	"啊"=0x0000b0a1 Stored as 0xa1b00000	Only two bytes (GB2132 or GBK) are meaningful
Stroke number	unsigned short	2B		Number of strokes in a sample
Strokes (concatenated). Each stroke is a point sequence from pen-down to lift				
Coordinates (x, y) (concatenated)	short	2B+2B		All values less than 32768
Stroke end (-1, 0)	signed short	2B+2B		
Character end tag				
Character end (-1,-1)	signed short	2B+2B		

Below is a snippet of code to read data from files. The only data we need it's the coordinates of stroke points, which will be processed into features. In order to

quickly grab these features as input for neural network training, they will be stored into a database.

```
my_path = 'OLHWDB1.1trn_pot'

# Iterate files
for file_name in os.listdir(my_path):
    data_file = open(my_path +  '/' +  file_name, 'rb')
    total_bytes = os.path.getsize(my_path +  '/' +  file_name)

    current_bytes = 0
    word_count = 0

    # Iterate characters
    while current_bytes != total_bytes:
        data_length, = struct.unpack('H', data_file.read(2))
        tag_code, = struct.unpack('I', data_file.read(4))
        stroke_number, = struct.unpack('H', data_file.read(2))
        current_bytes += data_length

        all_x_coor = []
        all_y_coor = []

        # Iterate strokes
        for stroke_index in range(stroke_number):
            stroke_x_coor = []
            stroke_y_coor = []

            # Iterate coordinates of points
            while 1:
                x, = struct.unpack('h', data_file.read(2))
                y, = struct.unpack('h', data_file.read(2))
                if x == -1:
                    break
                stroke_x_coor.append(x)
                stroke_y_coor.append(y)

            if len(stroke_x_coor) > 1:
                all_x_coor.append(stroke_x_coor)
                all_y_coor.append(stroke_y_coor)

        end_tag1, = struct.unpack('h', data_file.read(2))
        end_tag2, = struct.unpack('h', data_file.read(2))
```

4.2 PREPROCESSING

4.2.1 Introduction

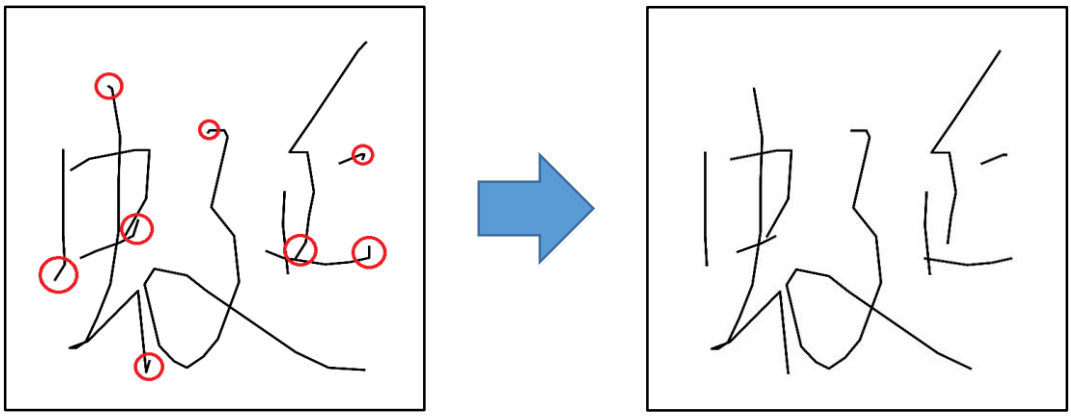
The better and complete image information we get from preprocessing, the more ideal the result obtained in the later processes of feature extraction and character recognition. If the result of preprocessing is not ideal, it not only reduces the system's recognition rate, but also affects the speed of recognition and the overall performance is reduced. In this paper, the involved techniques are: de-hook, normalization, and smooth.

4.2.2 De-hook

"Hooks can occur at the beginning and end of strokes due to inaccuracies in pen-down detection and rapid or erratic motion in placing the stylus on, or lifting it off the tablet. Usually, hooks can be detected by their location, small size and large angular variation." [13] In some papers, authors only compare the number of points with a specific threshold [13]. Stroke will be removed if it's below the threshold. In this reach, to remove hooks, start from the first point, and determine the direction of two points. (Detail of direction will be explained in the feature extraction part.) Continue to next point until the direction is changed. Then calculate the straight distance between the first point and the target point. If the distance is below the threshold, all points before the target one will be removed. Here, the threshold is calculated by the following logic for each stroke:

threshold = 10% * max((max_x_coordinate – min_x_coordinate),
(max_y_coordinate – min_y_coordinate))

Repeat the same thing for hooks at the end of strokes, but going backward instead. The threshold is tested between 5% and 25%. If it's too low, hooks cannot be detected. On the contrary, higher threshold will eliminate wrong segments. So, the experiment shows that 10% - 15% has better results. To choose a lower percentage 10% in this research because trying to keep the segment rather than missing stroke.



Original Character After de-hooked Character

FIGURE 8: HOOK REMOVAL

4.2.3 Normalization

Normalization is a very important part of preprocessing. Since the original images have huge differences in size, they must be normalized in order to have a unified size. This is very helpful to reduce training time of the neural network and improve recognition accuracy. The standard character image is used to unify different original characters to the same height and width. Normalization has two methods: projectile normalization [13] [14] and frame normalization. Projectile

normalization basically projects all points into target size of frame by ratio current size and target size. On the other hand, frame normalization will be utilized here because the former will stretch characters if the ratio of the height and the width between current size and target size are different. I think this may cause losing or changing the characteristics of input character so I rather keep the original ratio. Back to frame normalization, that is, the outside border of a character is enlarged or reduced in proportion linearly to the required size. First of all, the height and the width of the original character will be compared with the height and the width of the standard character, through which we'll get a transformation coefficient, and then change all points in original character by a ratio of this transformation

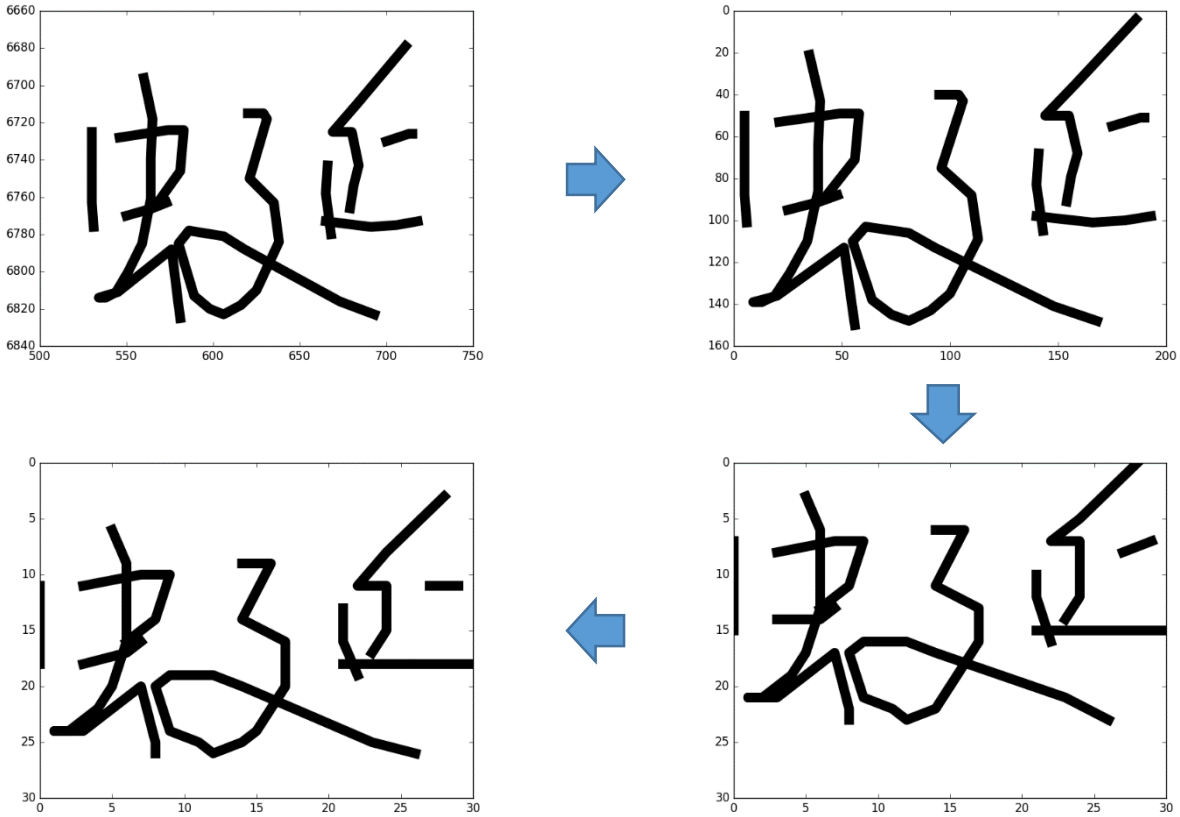


FIGURE 9: NORMALIZATION

coefficient. However, the dataset does not start with a 0 x coordinate and a 0 y coordinate. To solve this problem, simply find the minimum x coordinate and y coordinate of a whole character and then set all points relative to them. After the character is normalized, its position may move either toward the x-axis or the y-axis, so it can be centered. Firstly, calculate the spans over the x-axis and y-axis, which is the distance between the maximum coordinate and minimum coordinate of each axis. If the span of one axis is larger than the other one, all coordinates of that axis will add half of its span.

```
if x_span > y_span
        for all y coordinates + y_span / 2
else
        for all x coordinates + x_span / 2
```

4.2.4 Smooth Using Bezier Curve

The main purpose of smoothing is to reduce the noise in the character. Noise is unavoidable in image processing. Due to the existence of noise, the image will be unclear, the feature will not be obvious, and it seriously affects the performance of feature extraction and recognition. Therefore, efficient noise processing in image processing is essential. The types of noise are very complex and their impact on each image is also different. Selecting the appropriate noise reduction methods according to the nature of the noise is needed in order to achieve desired results. Here, a Bezier curve is proposed.

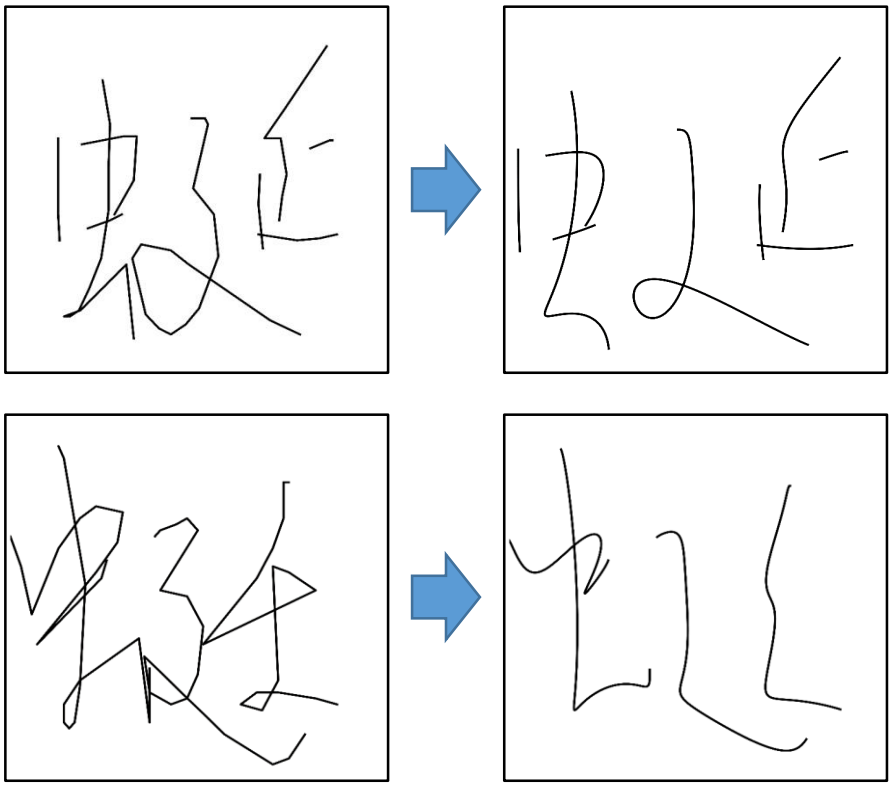
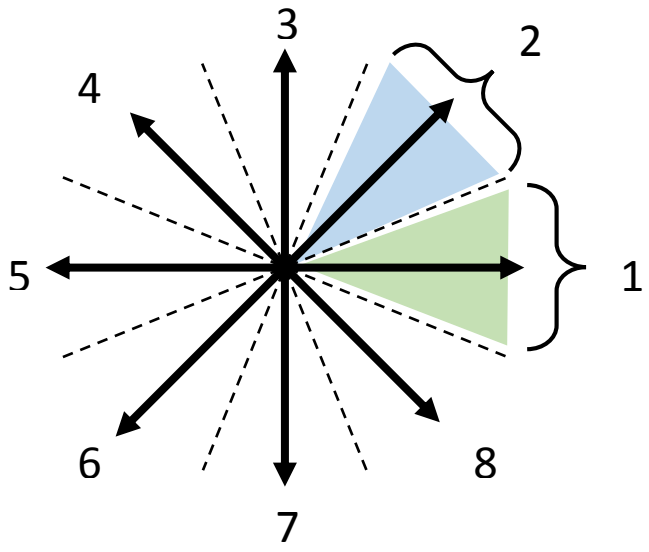


FIGURE 10: SMOOTH

4.3 FEATURE EXTRACTION

4.3.1 Directions

Define eight directions for two adjacent points shown as below. Along with the order of points in each stroke, generate a list of directional code. Discard the current direction if it is the same as previous one. Limit a maximum of 20 directional codes per list and also a maximum of 20 strokes per character. Therefore, a 20x20 array will be generated at the end. Fill with zeros if the array is not full.



Degree	Label
0° ≤ x ≤ 22.5° or 337.5° < x ≤ 360°	1
22.5° < x ≤ 67.5°	2
67.5° < x ≤ 112.5°	3
112.5° < x ≤ 157.5°	4
157.5° < x ≤ 202.5°	5
202.5° < x ≤ 247.5°	6
247.5° < x ≤ 292.5°	7
292.5° < x ≤ 337.5°	8

FIGURE 11: 8 DIRECTIONS

4.3.2 Zones

Standard Chinese characters are written within a square box so that each word occupies the same space. Chinese characters include two types of structures: single characters and composite characters. Single characters cannot be divided,

such as "大" and "中". On the other hand, composite ones are constructed by basic components. More than 90% of Chinese characters are composite. There are many combination structures. Here are some common ones:

- Half surrounded structure: 同, 过, 句
- Full surrounded structure: 回, 国, 围
- Top-bottom structure: 吉, 尖, 笑
- Left-right structure: 咯, 叫, 说
- Top-middle-bottom structure: 赢, 奚, 亵
- Left-middle-right structure: 糊, 脚, 谢
- Triangle ("品" shape) structure: 品, 森, 淼

Here, the frame of a character will be divided into 9 different regions so that it will cover most of the structures. Based on the order of strokes, the program will detect which region will be reached first. Therefore, an array with size of nine is

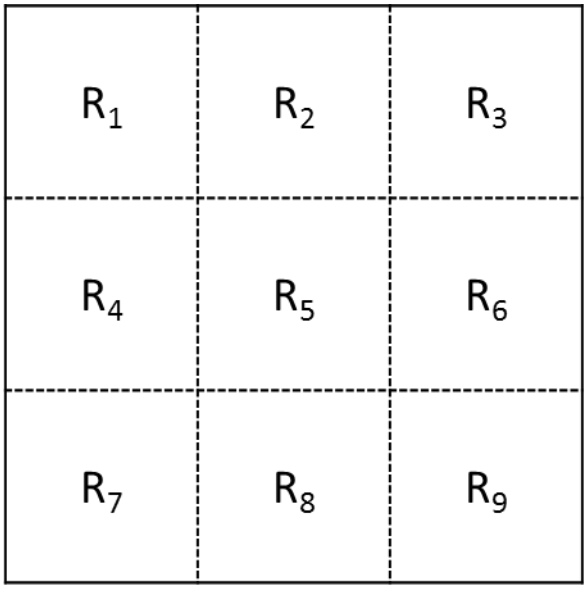


FIGURE 12: 9 ZONES

generated as a feature input for the neural network. Fill with zero if not all zones have been passed over.

4.3.3 Binary Image

This part will generate a 30x30 pixel binary image of the character. Simply initialize a 2-dimensional array with zeros and set a specific position to 1 according to the coordinate of character points. Convert them to integers since points in graphics are floating-type numbers. At the end, this binary image will be used in the convolutional neural network.

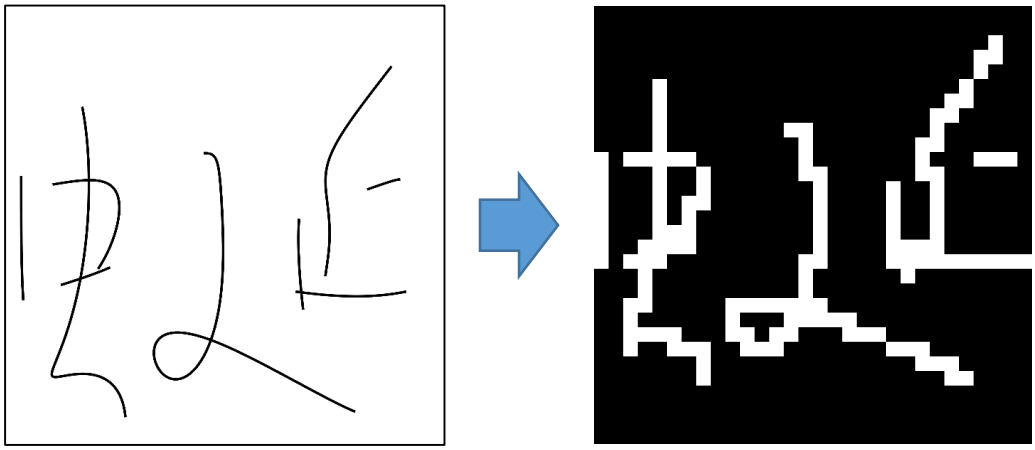


FIGURE 13: BINARY IMAGE

4.4 NEURAL NETWORK

There are two types of neural networks that will be utilized in this research: multi-layer neural network and convolutional neural network.

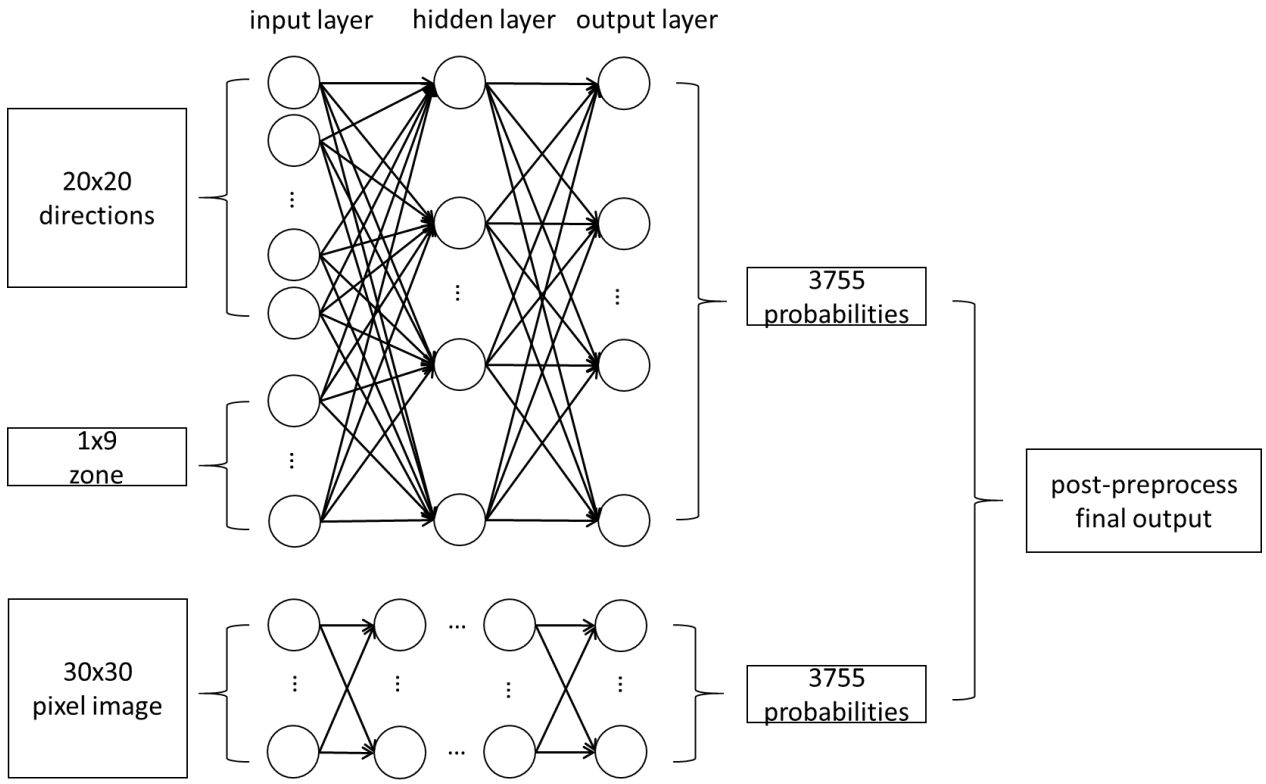


FIGURE 14: NEURAL NETWORK STRUCTURE

4.4.1 Multi-layer Neural Network

The input layer of this neural network contains neurons for features obtained during feature extraction. The direction feature is a 2-dimensional array with size 20x20. On the other hand, the region feature only contains 9 pieces of data that represent 9 zones. Hence, the input layer has a total $409 = 20 \times 20 + 9$ neurons.

The second layer is the hidden layer. Hidden layer is customizable. It can be one or more layers and each layer can contain any numbers of neurons. During the experiment, I'll test different setups to see which way has better results. More layer or less layer? More neurons or less neurons?

The output layer contains 3755 neurons because the dataset only have 3755 unique Chinese characters. Each neuron is the probability of the corresponding character.

Twenty of this type of neural network will be built according to the number of strokes. Since people will write cursively, characters will not have too many strokes. However, if it exceeds 20 strokes, we just consider the first 20.

4.4.2 Convolutional Neural Network

Same as multi-layer neural network, the number of neurons in input layer and output layer of CNN are fixed. The input is the 30x30 pixel image of the characters and output will have 3755 probabilities of each character. Besides, different structures of CNN will be tested in experiment.

4.5 POST-PROCESSING

Both neural networks will produce two lists with the probability for each character. Pick top 10 from these two lists and sort them in descending order of the probabilities. Give a value 10 to the highest probability and 1 to the lowest probability. Then sum all the values for the same character and sort them in descending order. Figure 15 is an example.

FIGURE 15: FIND THE MOST PROBABLE CHARACTER

5 RESULT

An experiment has been done on a total of 240 writers and each of them contains 3755 characters from CASIA dataset. Randomly choose 168 out of 240, which is 70%, for training the neural network. The rest 72 (30%) is for testing purpose. This is based on the regular training process ratio in neural network – 7:3. The results are shown below.

TABLE 2: MULTI-LAYER NEURAL NETWORK RESULT

Structure	**One Hidden Layer**		**Three Hidden Layers**		**Six Hidden Layers**		
	L1: 1000	L1: 2000	L1: 1000 L2: 1000 L3: 1000	L1: 1500 L2: 1500 L3: 1500	L1: 500 L2: 500 L3: 500 L4: 500 L5: 500 L6: 500	L1: 1000 L2: 1000 L3: 1000 L4: 1000 L5: 1000 L6: 1000	L1: 1500 L2: 1500 L3: 1500 L4: 1500 L5: 1500 L6: 1500
Accuracy	51.18%	61.48%	55.54%	54.81%	51.42%	64.82%	59.52%

TABLE 3: CONVOLUTIONAL NEURAL NETWORK RESULT

Structure #	#1	#2	#3
Convolutional Layer	25, 3x3	30, 5x5	60, 7,7
Pooling Layer	2x2	2x2	2x2
Convolutional Layer	50, 3x3	60, 4x4	80, 4x4
Pooling Layer	2x2	2x2	3x3
Fully Connected Layers	L1: 1000 L2: 1000	L1: 1000 L2: 500 L3: 500	L1: 1000 L2: 500 L3: 500
Accuracy	64.58%	69.96%	65.34%

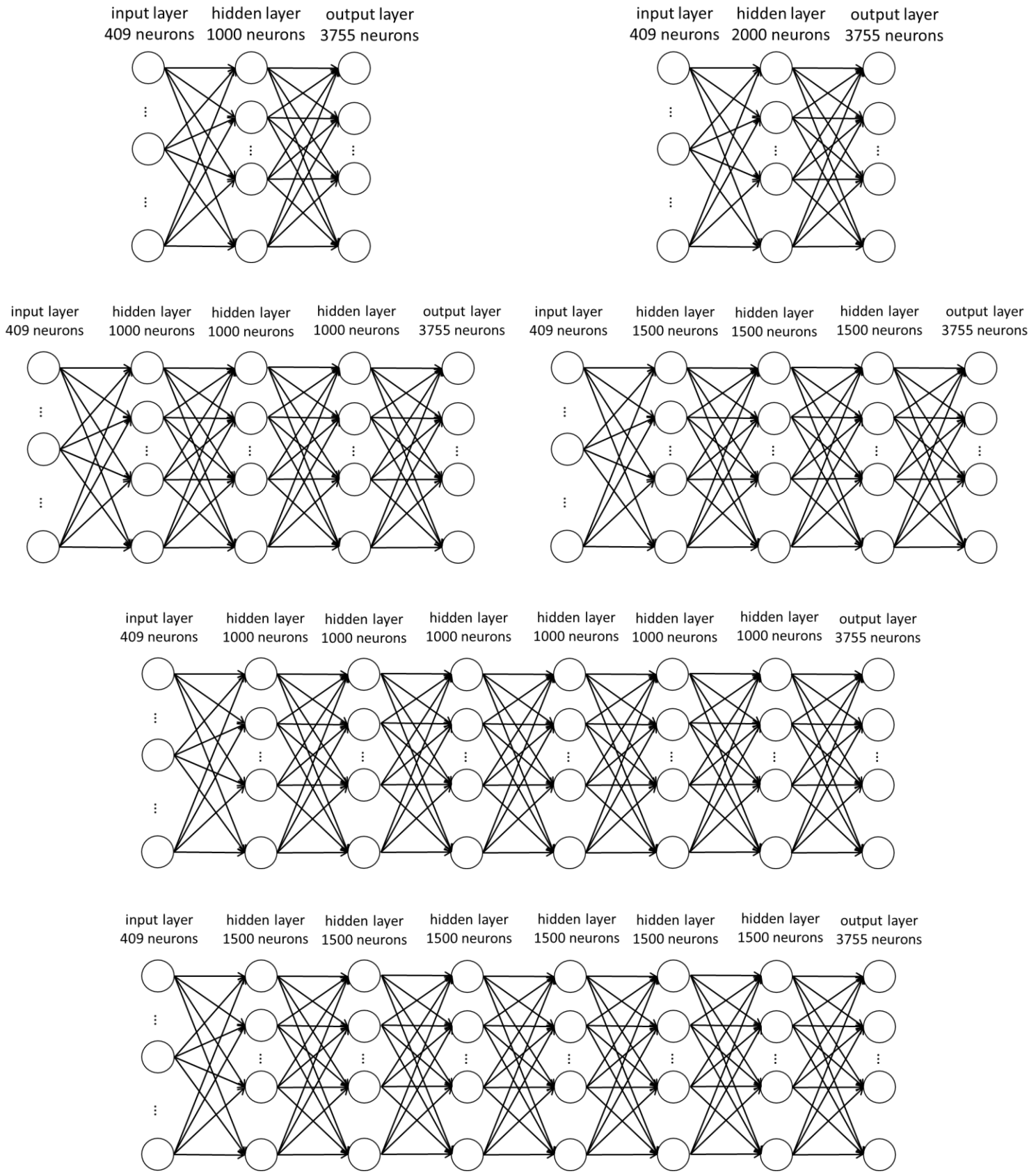


FIGURE 16: DIFFERENT STRUCTURES OF MULTI-LAYER NEURAL

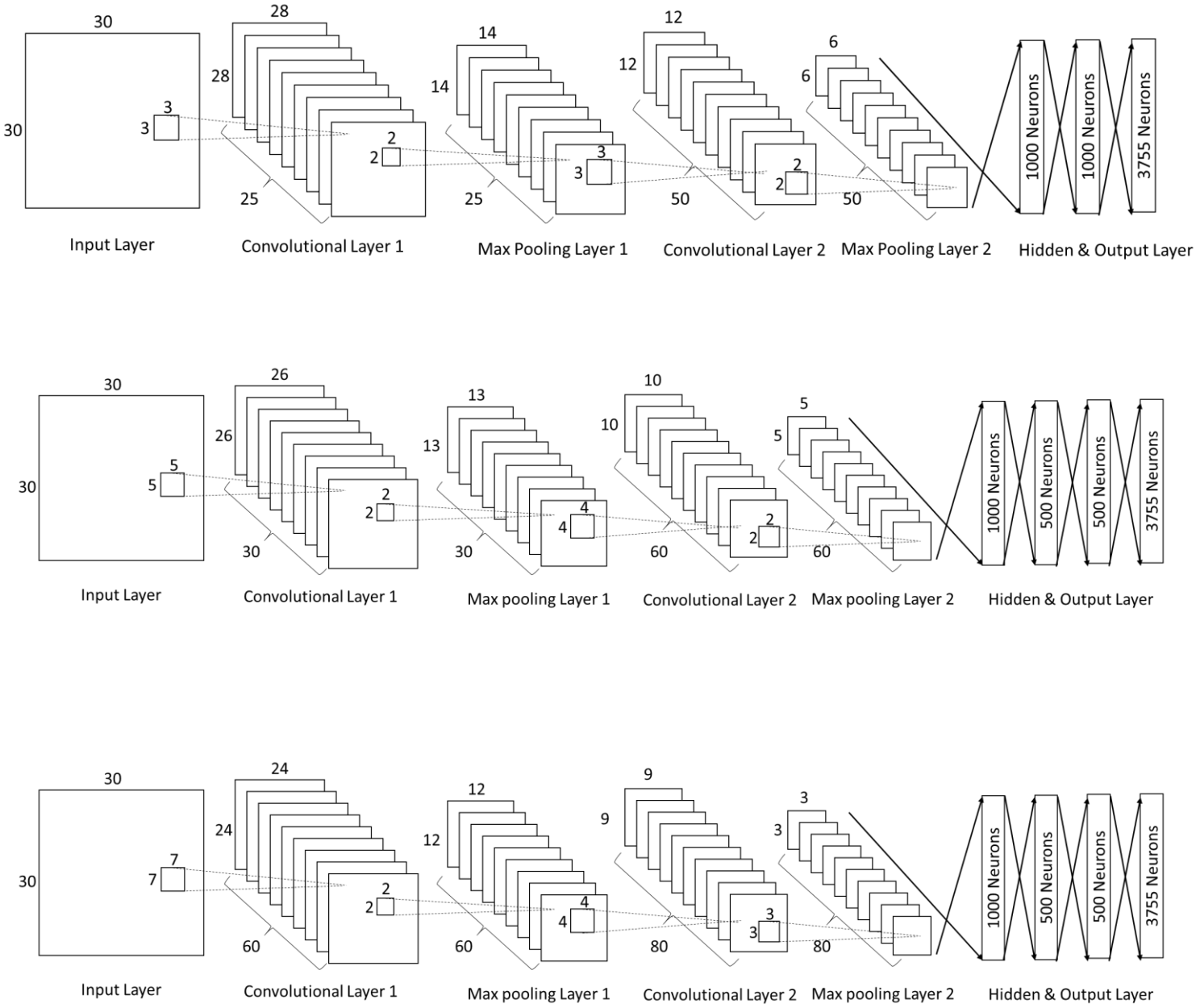


FIGURE 17: DIFFERENT STRUCTURES OF CONVOLUTIONAL NEURAL

As we can see in Table 1 and Table 2, neither increasing the number of hidden layers nor the number of neurons can always improve the performance. So, neural network is like a black box. We cannot determine what kind structures are better for training the model for any recognition. The only way is try different setups and see which can produce better results.

TABLE 4: FINAL RESULT

	QuickStroke	This Research
Top1	97.3%	96.84%
Top2	98.25%	98.31%
Top3	98.47%	98.39%

Comparing to QuickStroke, the accuracy of recognition does not have significant improvement because first one or two strokes are mostly similar so it's very hard to recognize.

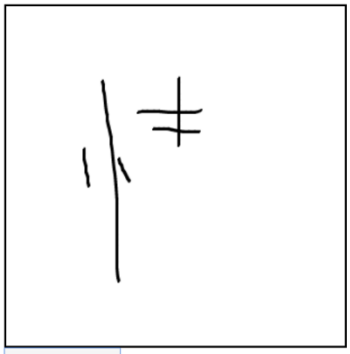
Figure 18 is the actual user interface on a web browser. The top square box is the drawing area for users to write the character. A clear button at the bottom to remove all drawings. Following is the top 10 predicted results, sorted by descending order of probability. The program will automatically predict the input while user is drawing. The run time is a bit slow and it takes about 8 to 10 seconds to display the results. I have tested it and figured out that the program spends too much time to load the trained models.

Clear

疑 籍 簿 篷 颖 簇 窥 镑 梁 蔬

Clear

理 瑶 玛 竭 琵 摄 埋 螺 瑰 嫂



Clear

情 愤 懂 慌 惜 悸 惰 懒 懦 幢

FIGURE 18: USER INTERFACE

6 CONCLUSION AND FUTURE WORK

It is difficult to conquer the problem of online handwritten Chinese character recognition. Through data acquisition or transmission equipment, it is convenient and fast to input Chinese characters into a computer. Online handwritten Chinese character recognition has very broad prospects for application, and has become a popular but difficult research area in the field of machine recognition.

Even though there have been a lot of research advancements on online handwritten Chinese character recognition, the recognition performance still remains unsatisfactory because of the large number of Chinese characters, the huge amount of noise in handwriting, and the amount of similarity between certain characters. The main topics handled in this research can be summarized below:

- Analyzing the research about online handwritten Chinese character recognition. Describe in detail the work at each stage in the process of Chinese character recognition and the processing algorithms or methods used in each stage.
- Feature extraction is the key of whole Chinese character recognition system. In order to improve the performance of character recognition, the feature accuracy plays an important role. Thus, the usage of preprocessing, which contains hooks removal, normalization, and stroke smoothing, can improve the performance of feature extraction.

- Utilize different types of neural networks to improve the performance of recognition: multi-layer neural network and convolutional neural network.
- Utilize different tools to build a real time handwritten Chinese character recognition web application.

The use of current knowledge to solve Chinese character recognition system is still partial and incoherent. Comprehensively utilizing various types of knowledge and researches to apply to online handwritten Chinese character recognition will require further, in-depth research. Meanwhile, improving the performance of each part of online handwritten Chinese character recognition so that the recognizing speed and result accuracy are enhanced is the goal of my current study.

7 REFERENCES

[1] Wikipedia, "Pinyin," [Online]. Available: https://en.wikipedia.org/wiki/Pinyin. [Accessed 10 August 2016].

[2] Wikipedia, "Handwriting Recognition," [Online]. Available: https://en.wikipedia.org/wiki/Handwriting_recognition. [Accessed 24 April 2016].

[3] B. Zhu and M. Nakagawa, "Online Handwritten Chinese/Japanese Character Recognition," 2012.

[4] C.-L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritten digit recognition: investigation of normalization and feature extraction techniques," Central Research Laboratory, Hitachi Ltd., Tokyo, Japan, 2002.

[5] R. Rojas, Neural Networks - A Systematic Introduction, Berlin: Springer-Verlag, 1996.

[6] M. Cilimkovic, "Neural Networks and Back Propagation Algorithm," Institute of Technology Blanchardstown.

[7] N. P. Matic, J. C. Platt and T. Wang, "QuickStroke: An Incremental On-line Chinese Handwriting Recognition System," ICPR, 2002.

[8] C.-L. Liu, F. Yin, D.-H. Wang and Q.-F. Wang, "CASIA Online and Offline Chinese Handwriting Databases," in *2011 International Conference on Document Analysis and Recognition*, Beijing, 2011.

[9] "NumPy," [Online]. Available: https://en.wikipedia.org/wiki/NumPy. [Accessed 2 May 2016].

[10] "XAMPP," [Online]. Available: https://en.wikipedia.org/wiki/XAMPP. [Accessed 1 May 2016].

[11] "jSignature," [Online]. Available: https://willowsystems.github.io/jSignature/#/about/. [Accessed 2 May 2016].

[12] A. Rosebrock, "pyimagesearch," 27 June 2016. [Online]. Available: http://www.pyimagesearch.com/2016/06/27/my-top-9-favorite-python-deep-learning-libraries/. [Accessed 14 November 2016].

[13] S. M.s and S. Idicula, "On-Line Handwritten Character Recognition using Kohonen Networks," in *Nature & Biologically Inspired Computing*, 2009.

[14] D. Singh, S. K. Singh and M. Dutta, "Hand Written Character Recognition Using Twelve Directional Feature Input and Neural Network," *International Journal of Computer Applications,* vol. 1, no. 3, pp. 82-85, 2010.

[15] "Running Python Scripts on Windows with Apache and Xampp," [Online]. Available: http://elvenware.com/charlie/development/web/Python/Xampp.html. [Accessed 1 May 2016].

[16] L. Austin, "Install Python and Django with Xampp on Windows 7," 9 December 2010. [Online]. Available: http://www.leonardaustin.com/blog/technical/install-python-and-django-with-xampp-on-windows-7/. [Accessed 1 May 2016].

[17] "how to run python through cgi in xampp all error solved," 11 April 2015. [Online]. Available: http://learn2programming.blogspot.com/2015/04/how-to-run-python-through-cgi-in-xampp.html. [Accessed 1 May 2016].

[18] "Apache Spark," [Online]. Available: http://spark.apache.org/. [Accessed 2 Map 2016].

[19] M.-G. Wen, K.-C. Fan and C.-C. Han, "Classification of Chinese Characters Using Pseudo Skeleton Features," *Journal Of Information Science And Engineering,* vol. 20, pp. 903-922, 2004.

[20] B. Verma, J. Lu, M. Ghosh and R. Ghosh, "A Feature Extraction Technique for Online Handwriting Recognition," in *Neural Networks*, 2004.

[21] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß and J. Schmidhuber , "PyBrain," *The Journal of Machine Learning Research,* vol. 11, pp. 743-746, 2010.

[22] M. I. Razzak, S. A. Husain, A. A. Mirza and A. Belaid, "FUZZY BASED PREPROCESSING USING FUSION OF ONLINE AND OFFLINE TRAIT FOR ONLINE URDU SCRIPT BASED LANGUAGES CHARACTER RECOGNITION," *International Journal of Innovative,* vol. 8, no. 5, pp. 3149-3161, 2012.

[23] S. Rao and E. Reddy, "Comparative Analysis of Pattern Recognition Methods: An Overview," Indian Journal of Computer Science and Engineering (IJCSE), 2011.

[24] H. J. Kim, J. W. Jung and S. K. Kim, "On-line Chinese character recognition using ART-based stroke classification," *Pattern Recognition Letters,* vol. 17, pp. 1311-1322, 1996.

[25] "UNDERSTANDING CONVOLUTIONAL NEURAL NETWORKS FOR NLP," WILDML, 7 November 2015. [Online]. Available: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. [Accessed 15 November 2016].