



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Handwritten Digit and Signal Recognition for Children

EIE4512 2023-24 Summer

2024/7

Group 1

Boshi Xu Yuhang Huang Yiwei Huang



01

PART 01

Part1 Motivation



//

In today's digital era, labor costs are increasingly expensive, and the digitalization of early childhood education has become an inevitable trend. To save parents and teachers from spending time on simple homework correction, we aim to design a program that can check children's answers of simple handwritten calculation.

//

Motivation



02

PART 02

Part2 Goal

02 Goal



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

The goal of our project is to create a simple and easy-to-use handwritten recognition and calculation product.

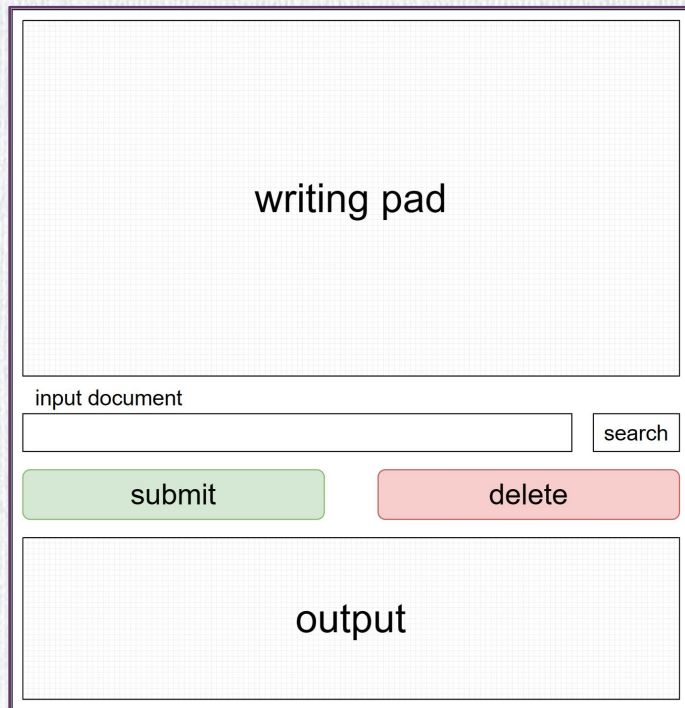
The front-end part includes:

1. Handwriting board
2. Optional file input
3. Text result output

The diagram illustrates the front-end interface of the project. It consists of a large rectangular area at the top labeled "writing pad" with a light gray grid background. Below this is a section for file input, labeled "input document", which includes a text input field and a "search" button. Underneath the input field are two buttons: a green "submit" button and a red "delete" button. At the bottom of the interface is another large rectangular area labeled "output" with a light gray grid background.

Pipeline

Input from Handwritten
Board or document



writing pad

input document

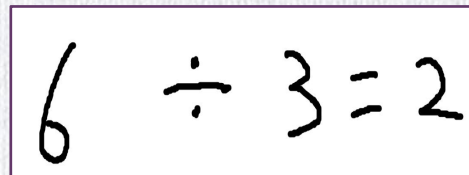
search

submit delete

output

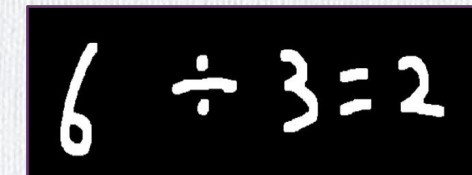
Output to the expected
part of the frontend page

Read the input image

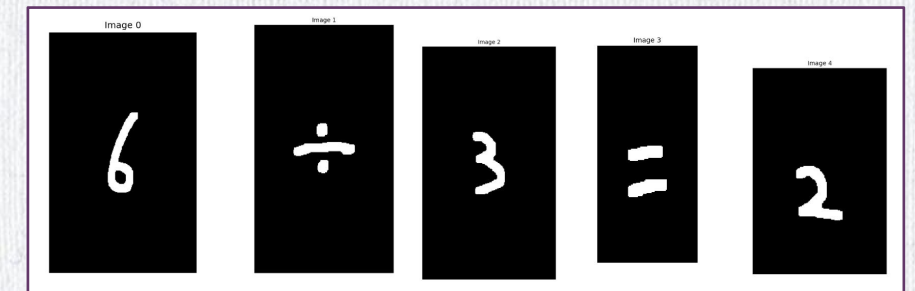


6 ÷ 3 = 2

Image after processing



6 ÷ 3 = 2



Separated Images

CORRECT!
6 ÷ 3 = 2

CNN proceed result



03

PART 03

Part3 Datasets

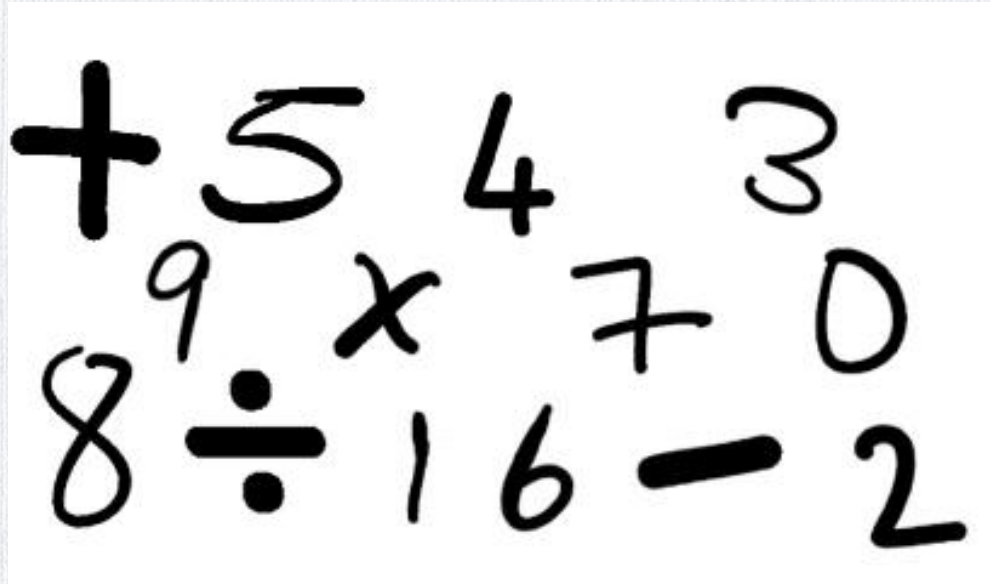
03 Datasets



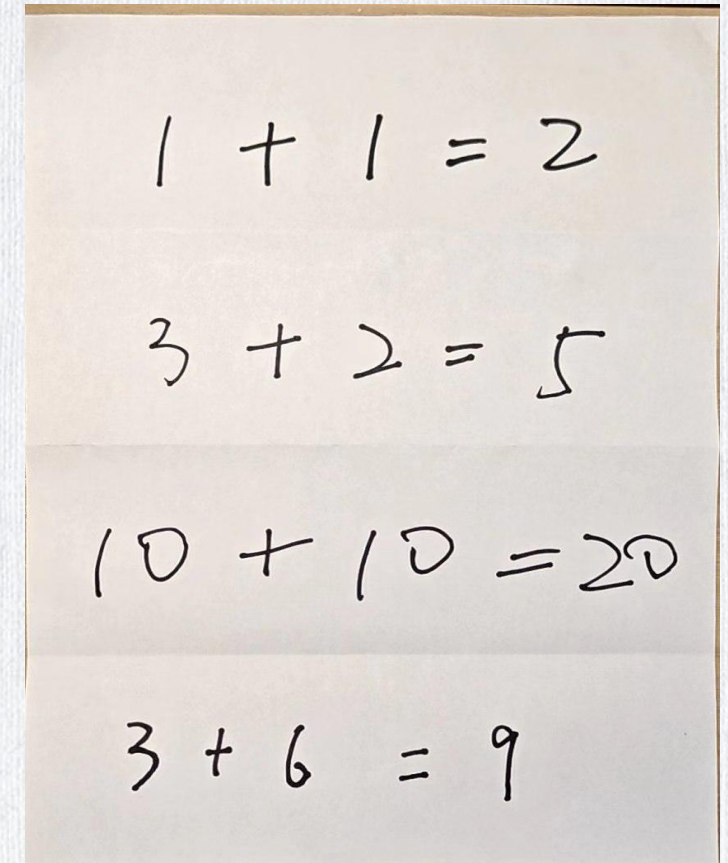
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

The dataset combine with two parts:

1. *Mathematics Symbols Data* collected from the Internet
2. Numbers of children's handwriting pictures collected by ourselves.



Mathematics Symbols Data



Children's handwriting

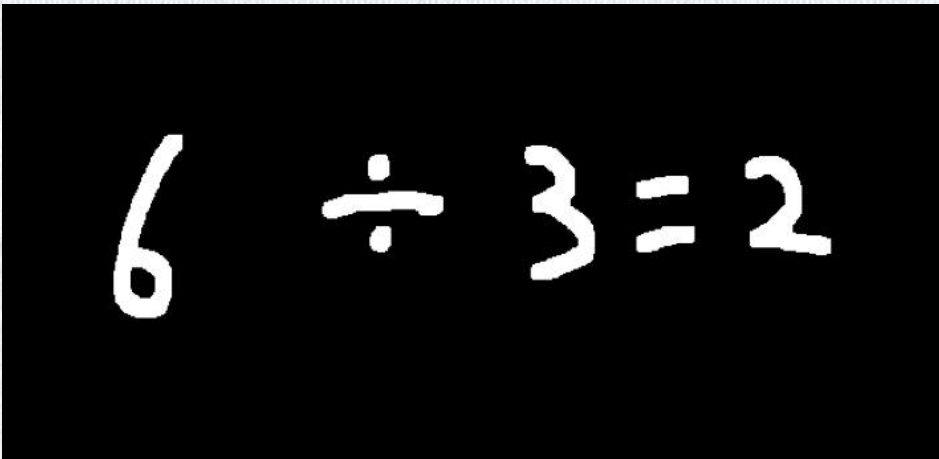


04

PART 04

Part4 Methodology

1. OpenCV Preprocessing

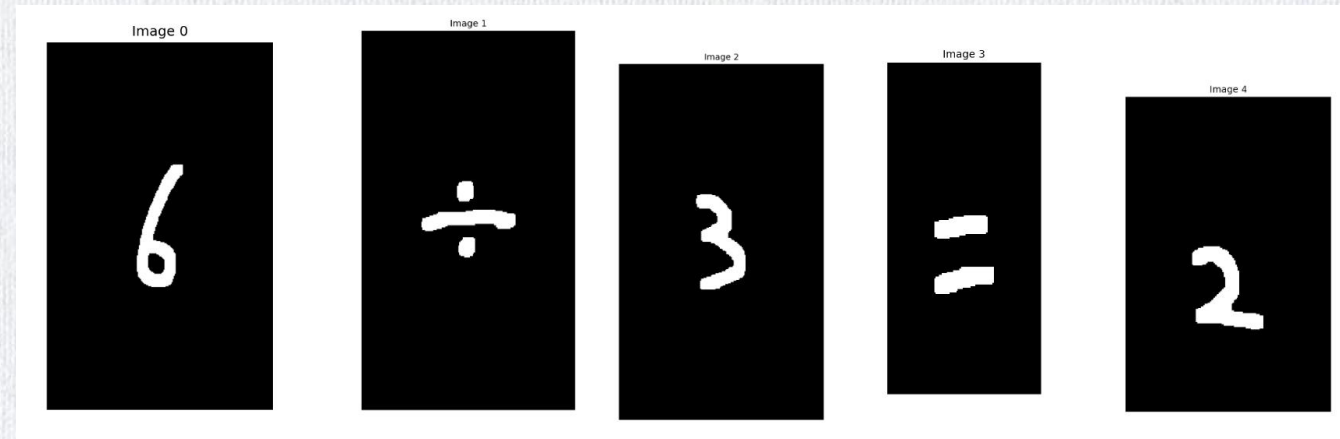


Mathematics Symbols Data

1. Preprocessing of Input Image

The input image is subjected to a series of operations including Gaussian blurring, edge detection, and dilation and erosion to facilitate subsequent processing.

1. OpenCV Splitting



Separated
Images

2. Using contour recognition to split equations into individual numbers and symbols.

Contour recognition is used to obtain correctly sized bounding boxes, which are then used to crop the image to obtain individual images. However, traditional contour recognition treats division signs and equal signs as multiple objects, resulting in multiple bounding boxes. Therefore, we identify division signs and equal signs by detecting the distance between each contour and considering two contours that are too close to each other as one.



2. CNN Training

1. Process the data by collecting them to npy files of NumPy array.

2. Build and train a convolutional neural network (CNN) model named CNN_result.h5 to classify the images.

```
Saved 2840 images to D:\Python_code\Output\add.npy
Saved 2629 images to D:\Python_code\Output\divide.npy
Saved 2775 images to D:\Python_code\Output\eight.npy
Saved 2827 images to D:\Python_code\Output\five.npy
Saved 3180 images to D:\Python_code\Output\four.npy
Saved 3160 images to D:\Python_code\Output\multiply.npy
Saved 3212 images to D:\Python_code\Output\nine.npy
Saved 3631 images to D:\Python_code\Output\one.npy
Saved 3246 images to D:\Python_code\Output\seven.npy
Saved 2984 images to D:\Python_code\Output\six.npy
Saved 3452 images to D:\Python_code\Output\subtract.npy
Saved 2586 images to D:\Python_code\Output\three.npy
Saved 3828 images to D:\Python_code\Output\two.npy
Saved 2399 images to D:\Python_code\Output\zero.npy
```

```
Epoch 1/10
1069/1069 ————— 358s 332ms/step - accuracy: 0.6530 - loss: 1.2670 - val_accuracy: 0.9216 - val_loss: 0.2641
Epoch 2/10
1069/1069 ————— 351s 329ms/step - accuracy: 0.9164 - loss: 0.2700 - val_accuracy: 0.9504 - val_loss: 0.1668
Epoch 3/10
1069/1069 ————— 350s 327ms/step - accuracy: 0.9454 - loss: 0.1700 - val_accuracy: 0.9592 - val_loss: 0.1379
Epoch 4/10
1069/1069 ————— 358s 335ms/step - accuracy: 0.9682 - loss: 0.0984 - val_accuracy: 0.9614 - val_loss: 0.1392
Epoch 5/10
1069/1069 ————— 389s 364ms/step - accuracy: 0.9780 - loss: 0.0706 - val_accuracy: 0.9593 - val_loss: 0.1635
Epoch 6/10
1069/1069 ————— 411s 384ms/step - accuracy: 0.9802 - loss: 0.0590 - val_accuracy: 0.9586 - val_loss: 0.1685
Epoch 7/10
1069/1069 ————— 412s 385ms/step - accuracy: 0.9864 - loss: 0.0409 - val_accuracy: 0.9676 - val_loss: 0.1231
Epoch 8/10
1069/1069 ————— 412s 385ms/step - accuracy: 0.9905 - loss: 0.0316 - val_accuracy: 0.9674 - val_loss: 0.1442
Epoch 9/10
1069/1069 ————— 411s 385ms/step - accuracy: 0.9884 - loss: 0.0356 - val_accuracy: 0.9650 - val_loss: 0.1606
Epoch 10/10
1069/1069 ————— 414s 388ms/step - accuracy: 0.9903 - loss: 0.0321 - val_accuracy: 0.9671 - val_loss: 0.1371
```


2. CNN Recognition

1. Read and preprocess the input images to fit the size of model.
2. Use trained CNN model to recognize the characters in the image.
3. Check the calculations and output the result to the front

```
Processed image saved as processed_a1.png  
1/1 ————— 0s 27ms/step  
Processed image saved as processed_a2.png  
1/1 ————— 0s 23ms/step  
Processed image saved as processed_a3.png  
1/1 ————— 0s 24ms/step  
Processed image saved as processed_a4.png  
1/1 ————— 0s 23ms/step  
Processed image saved as processed_a5.png  
1/1 ————— 0s 23ms/step
```

```
print(recognized_characters)
```

```
['6', '÷', '3', '=', '2']
```

CORRECT!

$6 \div 3 = 2$



2. CNN Recognition

1. Read and preprocess the input images to fit the size of model.
2. Use trained CNN model to recognize the characters in the image.
3. Check the calculations and output the result to the front

```
Processed image saved as processed_a1.png  
1/1 ————— 0s 20ms/step  
Processed image saved as processed_a2.png  
1/1 ————— 0s 19ms/step  
Processed image saved as processed_a3.png  
1/1 ————— 0s 21ms/step  
Processed image saved as processed_a4.png  
1/1 ————— 0s 21ms/step  
Processed image saved as processed_a5.png  
1/1 ————— 0s 22ms/step
```

```
print(recognized_characters)
```

```
['6', '÷', '3', '=', '3']
```

T_T WRONG

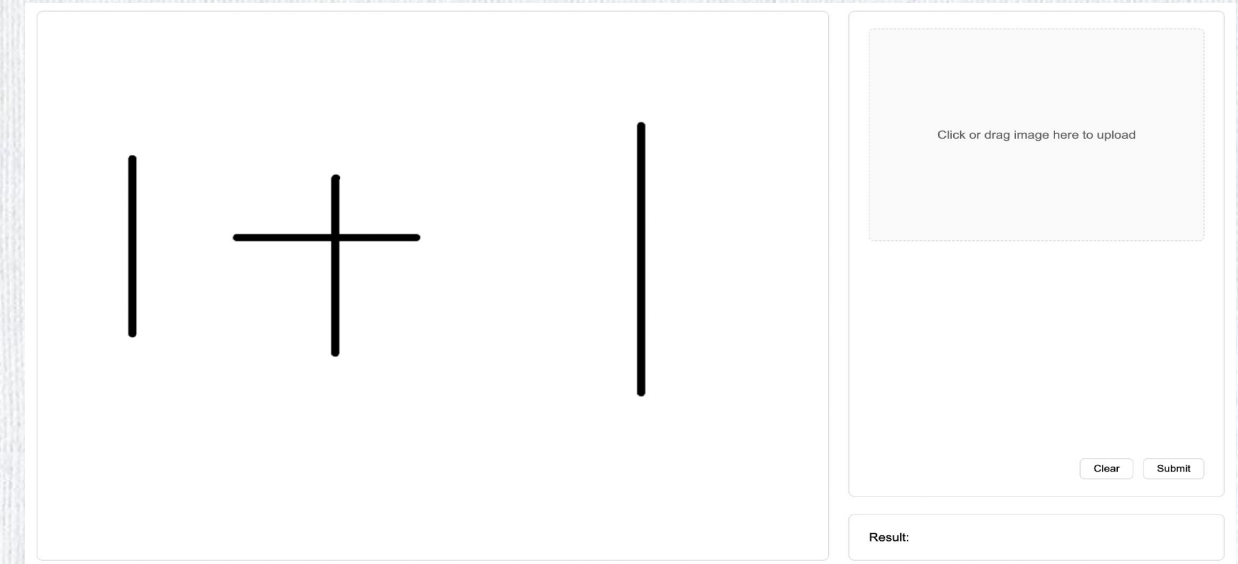
$6 \div 3 = 2$

3. Front End



Diagram illustrating the front-end interface components:

- writing pad**: A large text area for drawing or writing.
- input document**: A text input field.
- search**: A button to search for documents.
- submit**: A green button to submit the input.
- delete**: A red button to delete the input.
- output**: A large text area for displaying the result.



1. Complete the HTTP front-end code using the React framework.
2. Use the canvas context to track mouse movement and complete image drawing.
3. Implement image uploading using an input dom element with type=file.
4. Use a simple Python server and HTTP API to obtain the uploaded image from the front-end, call the algorithm, and return the resulting image.

Video Presentation



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'BACKEND' folder containing various files, including 'main.py', 'CNN_recognition.py', and several image files. The code editor displays the 'main.py' file, which contains a Python script for a web application. The script defines an 'evaluate' function and a 'RequestHandler' class with 'do_GET' and 'do_POST' methods. The 'do_POST' method handles image uploads and calls the 'evaluate' function. The status bar at the bottom indicates the file is 'main.py' at line 31, column 23, with a UTF-8 encoding and LF line endings.

```
10  
11 函数注释 | 行间注释 | 生成单测 | 代码解释 | 调优建议  
12 def evaluate(input):  
13     # TODO: 完成这个函数, 将前端传来的input图片识别并返回output图片  
14     # 下面是样例代码, 会把input图片保存到本地input.png文件中并直接把文件再返回回去  
15     file_path = dir_path + '/input.png'  
16     with open(file_path, 'wb') as file:  
17         file.write(input.read())  
18     subprocess.run(['python', path.join(dir_path, 'pre_process.py')])  
19     subprocess.run(['python', path.join(dir_path, 'CNN_recognition.py')])  
20     file_path = dir_path + '/output.jpg'  
21     with open(file_path, 'rb') as file:  
22         return file.read()  
23  
24 class RequestHandler(BaseHTTPRequestHandler):  
25     函数注释 | 行间注释 | 生成单测 | 代码解释 | 调优建议  
26     def do_GET(self):  
27         file_path = dir_path + '/public' + ('/index.html' if self.path == '/' else self.path)  
28         with open(file_path, 'rb') as file:  
29             self.send_response(200)  
30             self.send_header('Content-Type', guess_type(file_path)[0])  
31             self.end_headers()  
32             self.wfile.write(file.read())  
33     函数注释 | 行间注释 | 生成单测 | 代码解释 | 调优建议  
34     def do_POST(self):  
35         if self.path == '/api/evaluate':  
36             form = FieldStorage(  
37                 fp=self.rfile,  
38                 headers=self.headers,  
39                 environ={'REQUEST_METHOD': 'POST'})  
40             input = form['input'].file  
41             output = evaluate(input)  
42             self.send_response(200)
```




香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Thank you for Watching

Appendix: Reference



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

CNN reference:

<https://cloud.tencent.com/developer/article/2398369>

Dataset 1 of images of numbers & operators to train for maths expression solver:

<https://www.kaggle.com/datasets/amitamola/mathematics-symbols-data/data>

Dataset 2 of images of numbers & operators to train:

Collected by Teammates