嵌入式大作业课题报告

小组成员：2352662 吴秉寰 2250579 刘贯桥

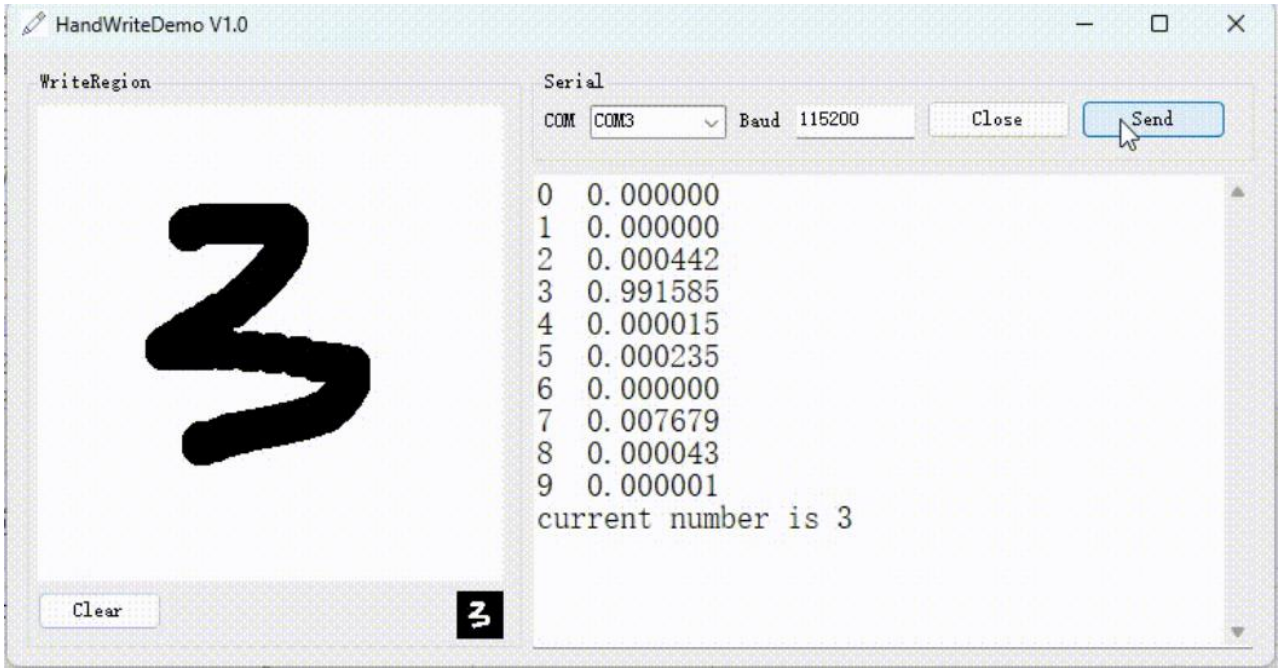
一、 选题：

利用 CUBE AI 在 STM32H747I-DISCO 或其它基于 ST MCU 的开发板上部署手写识别系统（使用板子型号：STM32F767IGT6）

二、主要目标：

在 HandWriteDemo 中写一个数字，通过串口通信的方式从该上位机传输给程序，程序中的神经网络经过计算来判断所写的是什么数字，然后再通过串口通信传输给 Demo，在 Demo 上就会显示神经网络判断的结果，这样就实现了对我们手写的内容的识别系统

示例：



三、核心功能：

1) 串口通信：接收上位机传输的手写数字图像数据，再将神经网络判断的结果传输给上位机
2) AI 模型部署：使用轻量级神经网络，通过 ai_platform 库初始化并运行模型
3) 数据处理：将图像像素值转换为模型输入格式，执行推理后解析输出概率，识别数字
4) 外设驱动：配置串口、时钟、CRC 等外设，确保系统稳定运行

四、主体代码及注释解析：

头文件部分：

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include <string.h>
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "ai_platform.h"
#include "network.h"
#include "network_data.h"

/* USER CODE END Includes */
```

包含必要的系统头文件和 AI 相关库、神经网络模型定义、神经网络权重数据等

函数原型说明:

```
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CRC_Init(void);
static void MX_USART1_UART_Init(void);
```

分别为：配置系统时钟、初始化 GPIO 引脚、初始化 CRC 外设、初始化 USART1 串口

全局变量定义（为神经网络运行准备数据结构）:

```
/* USER CODE BEGIN 0 */
ai_handle network;
float aiInData[AI_NETWORK_IN_1_SIZE];
float aiOutData[AI_NETWORK_OUT_1_SIZE];
ai_u8 activations[AI_NETWORK_DATA_ACTIVATIONS_SIZE];

ai_buffer * ai_input;
ai_buffer * ai_output;
```

aiInData：存储神经网络输入数据的数组

aiOutData：存储神经网络输出结果的数组

activations：存储神经网络中间层激活值的数组

ai_input、ai_output：ai_buffer 类型指针，后续会关联到 aiInData、aiOutData，作为模型输入输出的“缓冲区描述符”

函数原型声明（神经网络核心流程函数 ）：

```
static void AI_Init(void);
static void AI_Run(float *pIn, float *pOut);
void PictureCharArrayToFloat(uint8_t *srcBuf,float *dstBuf,int len);
```

AI_Init：负责神经网络的初始化工作

AI_Run：执行神经网络推理的核心函数，接收预处理后的输入数据，调用模型计算，寻找概率最大的数字结果

PictureCharArrayToFloat：数据转换函数，把从串口接收的图像数据转换为 float 类型

串口接收相关变量：

```
#define UART_BUFF_LEN 1024
#define ONE_FRAME_LEN 1+784+2
uint16_t uart_rx_length = 0;
uint8_t uart_rx_byte = 0;
uint8_t uart_rx_buffer[UART_BUFF_LEN];
volatile uint8_t goRunning = 0;
```

配置串口接收缓冲区，用于接收手写数字图像数据，这里的 1024 是串口接收缓冲区最大长度，1+784+2 是单帧数据长度（起始符+784 像素+校验）

主函数初始化：

```
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_CRC_Init();
MX_USART1_UART_Init();
/* USER CODE BEGIN 2 */
AI_Init();
memset(uart_rx_buffer,0,784);
HAL_UART_Receive_IT(&huart1, (uint8_t *)&uart_rx_byte, 1);
      Uart_send("AI is running!\r\n");
```

系统初始化：时钟、外设、AI 模型；等待串口接收图像数据，触发中断后填充缓冲区

主函数中 while 循环：

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

        char str[10];
        if(goRunning>0)
        {
                if(uart_rx_length == ONE_FRAME_LEN)
                {
                        PictureCharArrayToFloat(uart_rx_buffer+1,aiInData,28*28);
                        AI_Run(aiInData, aiOutData);

                }
                memset(uart_rx_buffer,0,784);
                goRunning = 0;
                uart_rx_length = 0;
        }
}
/* USER CODE END 3 */
```

功能：接收到完整一帧数据后，将图像数据转换为模型输入格式，调用 AI_Run 进行推理。Memset 函数是用来重置接收状态的

串口通信函数：

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *UartHandle)
{
        if(goRunning ==0)
        {
                if (uart_rx_length < UART_BUFF_LEN)
                {
                        uart_rx_buffer[uart_rx_length] = uart_rx_byte;
                        uart_rx_length++;

                        if (uart_rx_byte == '\n')
                        {
                                goRunning = 1;
                        }
                }
                else
                {
                        //rt_kprintf("rx len over");
                        uart_rx_length = 0;
                }
        }
        HAL_UART_Receive_IT(&huart1, (uint8_t *)&uart_rx_byte, 1);
}
```

这是串口接收中断回调函数，当未处于推理状态时(goRunning==0)接收数据、填充接收缓冲区，当检测到换行符，标志数据接收完成；若缓冲区溢出则重置；HAL_UART_Receive_IT作用为重新使能串口接收中断。

```
void Uart_send(char * str)
{
        HAL_UART_Transmit(&huart1, (uint8_t *)str, strlen(str),0xffff);
}
```

这是串口发送函数

数据处理函数：

```
void PictureCharArrayToFloat(uint8_t *srcBuf,float *dstBuf,int len)
{
        for(int i=0;i<len;i++)
        {
                dstBuf[i] = srcBuf[i];//==1?0:1;
        }
}
```

负责将图像像素值转换为 float 类型

AI 模型初始化函数：

```
static void AI_Init(void)
{
  ai_error err;

  /* Create a local array with the addresses of the activations buffers */
  const ai_handle act_addr[] = { activations };
  /* Create an instance of the model */
  err = ai_network_create_and_init(&network, act_addr, NULL);
  if (err.type != AI_ERROR_NONE) {
    printf("ai_network_create error - type=%d code=%d\r\n", err.type, err.code);
    Error_Handler();
  }
  ai_input = ai_network_inputs_get(network, NULL);
  ai_output = ai_network_outputs_get(network, NULL);
}
```

首先定义激活值缓冲区地址数组；然后创建并初始化神经网络模型实例，若初始化失败则报错；最后是获取模型的输入输出句柄

AI 推理函数（部分）：

```
static void AI_Run(float *pIn, float *pOut)
{
        char logStr[100];
        int count = 0;
        float max = 0;
  ai_i32 batch;
  ai_error err;

  /* Update IO handlers with the data payload */
  ai_input[0].data = AI_HANDLE_PTR(pIn);
  ai_output[0].data = AI_HANDLE_PTR(pOut);

  batch = ai_network_run(network, ai_input, ai_output);
  if (batch != 1) {
    err = ai_network_get_error(network);
    printf("AI ai_network_run error - type=%d code=%d\r\n", err.type, err.code);
    Error_Handler();
  }
```

首先绑定输入输出数据缓冲区，然后运行模型推理，这里的 batch==1 表示单张图像

AI 推理函数（剩余部分）：

```
  for (uint32_t i = 0; i < AI_NETWORK_OUT_1_SIZE; i++) {

          sprintf(logStr,"%d  %8.6f\r\n",i,aiOutData[i]);
          Uart_send(logStr);

          if(max<aiOutData[i])
          {
                  count = i;
                  max= aiOutData[i];
          }

  }
        sprintf(logStr,"current number is %d\r\n",count);
        Uart_send(logStr);
}
```

这里的 for 循环中串口会打印每个数字可能的概率，在 if 语句内更新最大概率的数字，最后调用函数使串口打印识别结果

五、程序 Debug 过程中遇到的问题及解决办法：

- ✓ Include 中报错

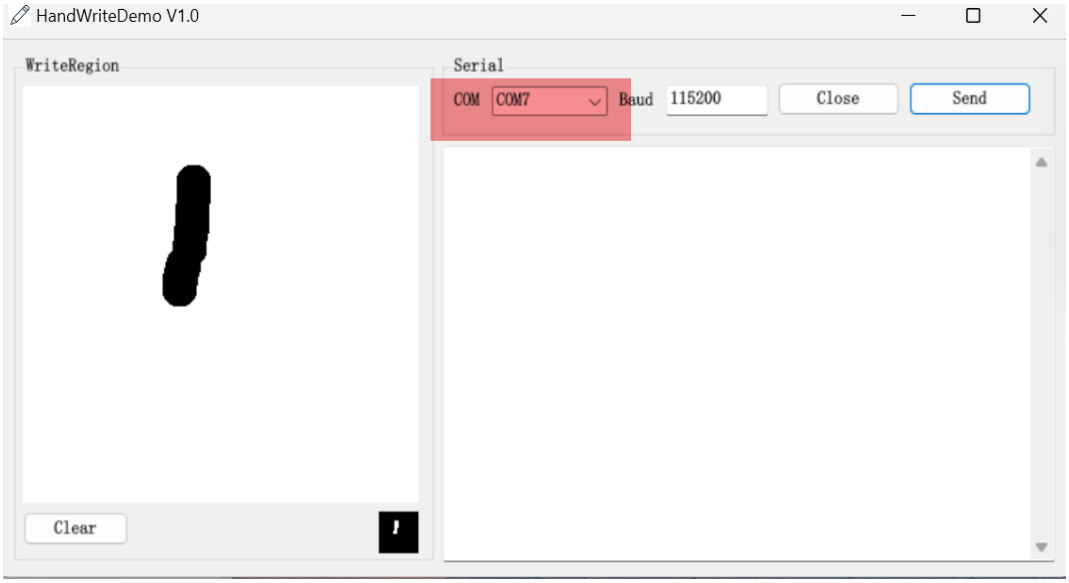
```
25 #include "ai_platform.h"
26 #include "network.h"
27 #include "network_data.h"
```

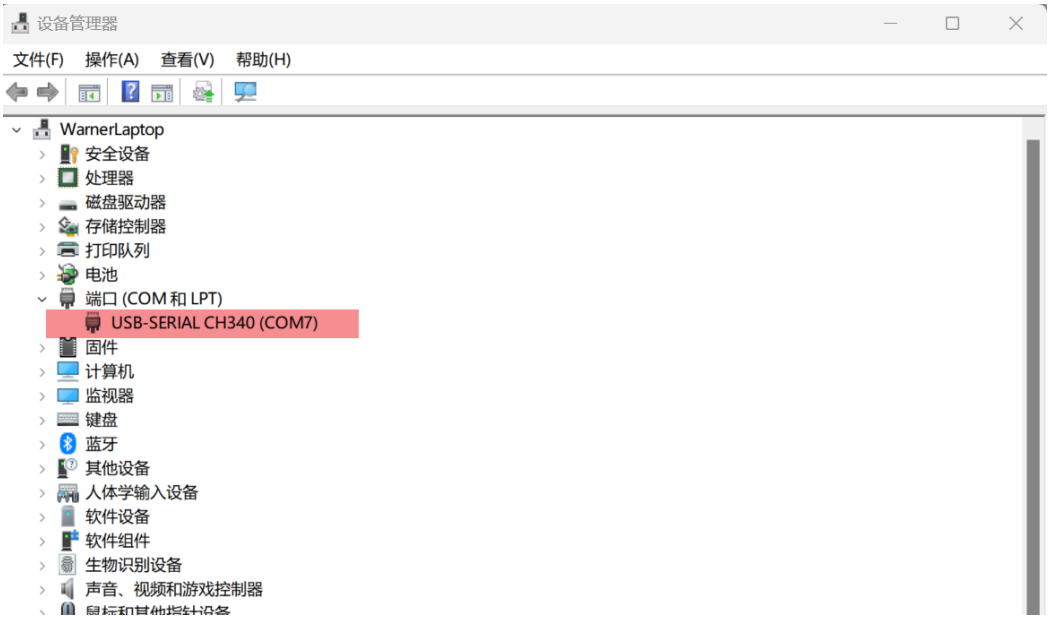
推测 bug 可能是重命名 network 导致的，现在回去重新命名 network 为 network（不要重命名为自己的名字）

```
25 #include "ai_platform.h"
26 #include "network.h"
27 #include "network_data.h"
```

问题解决

- ✓ 上位机软件下载后无应答





解决办法：自己写一个 usart 的示例，理解串口的逻辑，然后再去解决代码的 bug

当时写的示例代码运行报错:

```
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'D:\Applications\Keil_v5
Build target 'Hand_writing_recognition_wbh_lgq'
assembling startup_stm32f767xx.s...
compiling network_data_params.c...
compiling network_data.c...
compiling network.c...
compiling main.c...
../Core/Src/main.c(130): warning:  #223-D: function "memset" declared implicitly
    memset(uart_rx_buffer,0,784);
../Core/Src/main.c(144): warning:  #177-D: variable "str"  was declared but never
        char str[10];
../Core/Src/main.c(325): warning:  #223-D: function "strlen" declared implicitly
      HAL_UART_Transmit(&huart1, (uint8_t *)str, strlen(str),0xffff);
../Core/Src/main.c: 3 warnings, 0 errors
linking...
Program Size: Code=25516 RO-data=39676 RW-data=1636 ZI-data=14844
FromELF: creating hex file...
"Hand_writing_recognition_wbh_lgq\Hand_writing_recognition_wbh_lgq.axf" - 0 Error
Build Time Elapsed:  00:00:03
Load "Hand_writing_recognition_wbh_lgq\\Hand_writing_recognition_wbh_lgq.axf"
Erase Done.
Programming Done
```

在包含了<string.h>头文件后解决

Debug 经验：勾选 microLIB，才能使用重新定义的 printf

✓ 拒绝访问端口

解决办法：关掉串口助手之后再打开上位机

✓ 修复串口

1. 从板子开始：

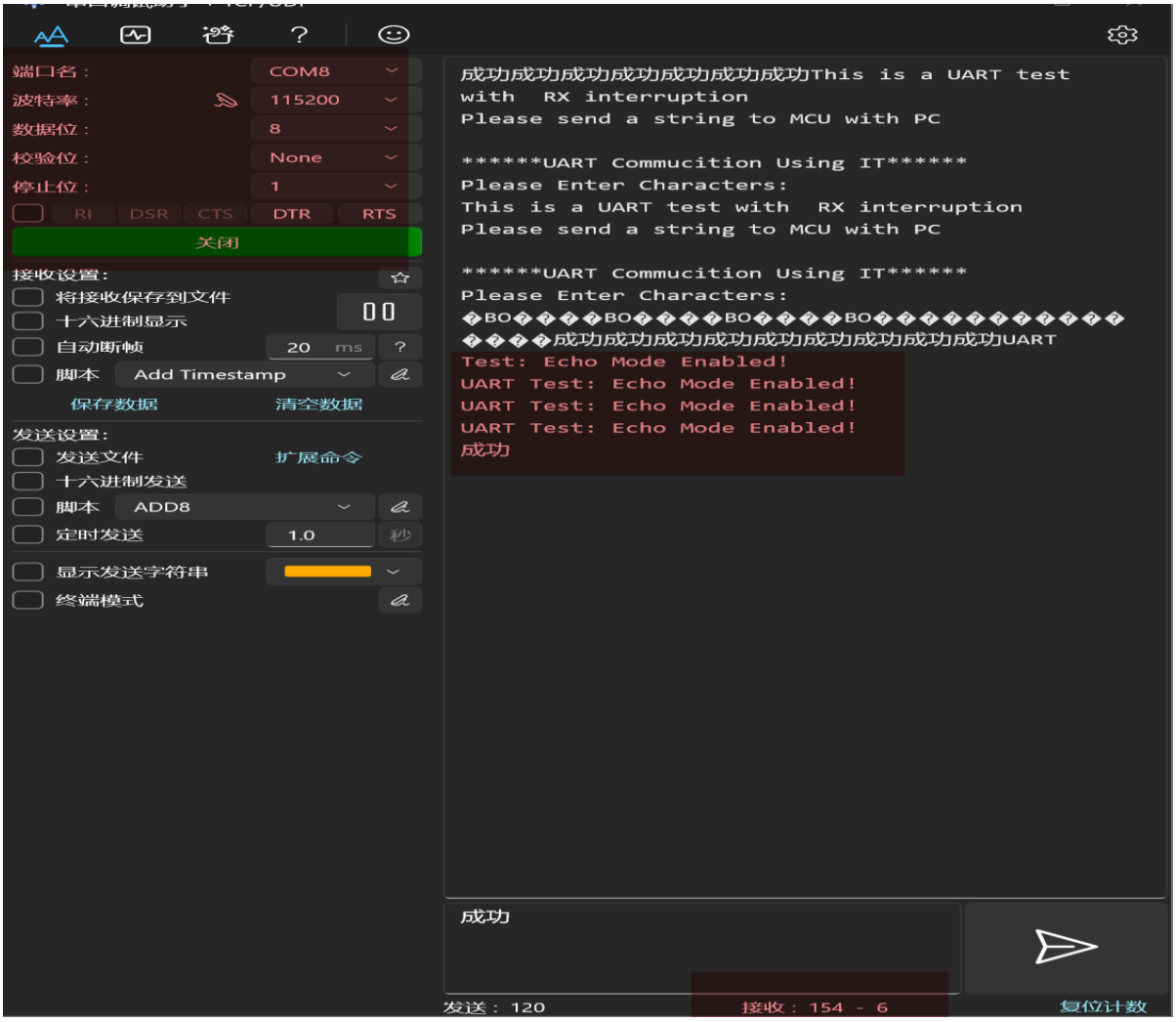
写好的 nano_uart_it 但是串口调试助手不显示

解决方法：

波特率要设置正确，如果高了则收不到，低了会有乱码

```
void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
```

2. 从开源代码学习 config 并基于 stm32f7 系列板子进行参数修改等适配操作



进行适配时的代码:

```
void HAL_UART_MspInit(UART_HandleTypeDef *huart)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    if(huart->Instance == USART1)
    {
        /* 使能时钟 */
        __HAL_RCC_GPIOA_CLK_ENABLE();
        __HAL_RCC_USART1_CLK_ENABLE();

        /* 配置 PA9(TX)、PA10(RX) */
        GPIO_InitStruct.Pin = GPIO_PIN_9 | GPIO_PIN_10;
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

        /* 配置中断 */
```

```
        HAL_NVIC_SetPriority(USART1_IRQn, 3, 3);
        HAL_NVIC_EnableIRQ(USART1_IRQn);
    }
}

void USART1_IRQHandler(void)
{
    HAL_UART_IRQHandler(&huart1); // 调用 HAL 处理函数
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart == &huart1)
    {
        // 回发接收到的字节
        HAL_UART_Transmit(&huart1, &uart_rx_byte, 1, HAL_MAX_DELAY);

        // 重新开启下一次接收
        HAL_UART_Receive_IT(&huart1, &uart_rx_byte, 1);
    }
```

六、Debug 的一些心得体会：

- 动手改 bug 之前要先明白原理，仔细阅读每个细节
- 开始改 bug 时要先缩小范围，在这个程序中知道 AI 部分代码应该是没问题的，那就把精力主要放在串口上
- 改 bug 时一次只改一个，发现不对赶紧改回来；同时可以用正常的示例和错误的示范对照着改
- 将每步操作、顺序、结果记录下来，通过这个办法解决了 network.h 的问题
- 对于硬件问题有时可能是连接处松动了，可以先试试重连