在 K64 上运行卷积神经网络

目录

一、简介

普通 mcu 受限于资源，很难做一些复杂的深度学习。不过虽然难，但是仍然可以做。CNN，卷积神经网络，是深度学习的一种，可以用来解决分类任务。实现 CNN 以后，普通 mcu 也能作边缘计算的设备，下面我们介绍在 FRDM-K64 上运行 CNN，来识别手写体数字，数字图片大小为 28x28。28x28 的图片经过 CNN 后，输出一个 1x10 的矩阵。网上给 mcu 写的深度学习库很少，即使有，但是又会出现各种问题。Nnom 框架，从移植到应用都比较轻松，所以我们就使用它了。

二、实验

1 所需工具：FRDM-K64，Python3.7，pip，IAR，TCP232

2 下载深度学习框架源码，https://github.com/majianjia/nnom 这是个纯 C 框架，并不依赖硬件结构，移植很方便

3 移植，我们选择 bubble 这个例程，将 inc，port 和 src 文件夹添加到工程中，如图

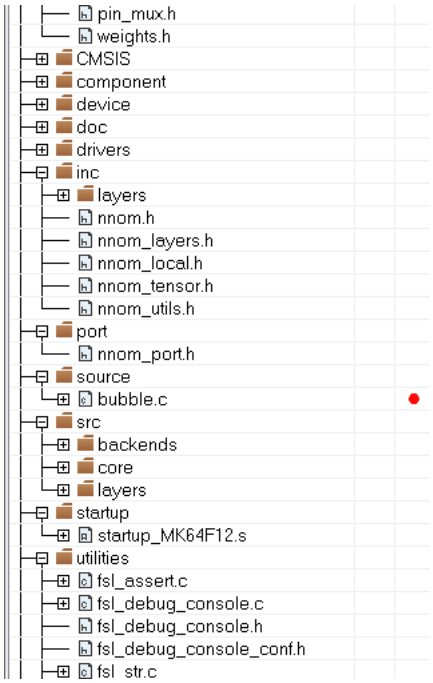

图 1

打开 port.h，将 NNOM_LOG 的串口改成 PRINTF(__VA_ARGS__)，打开 icf 文件，将堆大小改成 0x5000，define symbol __size_heap__ = 0x5000;
这个库用的 malloc，就是从这里分配内存，小了的话，就没办法运行网络。

4 从下载的框架里，找到 mnist-simple/mcu，里面有训练好的权重 weights.h 文件，还有随机生成的手写图片文件，image.h，将这两个文件加入工程。

5 在 bubble.c 文件里，添加头文件

```
#include "nnom_port.h"
#include "nnom.h"
#include "weights.h"
#include "image.h"
```

6 删掉原来代码，添加如下代码

```
nnom_model_t *model;
const char codeLib[] = "@B%8&WM#*oahkbdpqwmZO0QLCJUYXzcvunxrjft/\\|()1{}[]?-_+~<>i!ll;:,\"^`.    ";
/*******************************************************************************
 * Code
 ******************************************************************************/
void print_img(int8_t * buf)
{
    for(int y = 0; y < 28; y++)
    {
        for (int x = 0; x < 28; x++)
        {
```

```
                int index =    69 / 127.0 * (127 - buf[y*28+x]);
                if(index > 69) index =69;
                if(index < 0) index = 0;
                PRINTF("%c",codeLib[index]);
                PRINTF("%c",codeLib[index]);
            }
            PRINTF("\r\n");
        }
}

// Do simple test using image in "image.h" with model created previously.
void mnist(char num)
{
    uint32_t predic_label;
    float prob;
    int32_t index = num;
    PRINTF("\nprediction start.. \r\n");

    // copy data and do prediction
    memcpy(nnom_input_data, (int8_t*)&img[index][0], 784);
    nnom_predict(model, &predic_label, &prob);

    //print original image to console
    print_img((int8_t*)&img[index][0]);

    PRINTF("\r\nTruth label: %d\n", label[index]);
    PRINTF("\r\nPredicted label: %d\n", predic_label);
    PRINTF("\r\nProbability: %d%%\n", (int)(prob*100));
}

int main(void)
{
    uint8_t ch;
    /* Board pin, clock, debug console init */
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();
    /* Print a note to terminal */
    model = nnom_model_create();
    // dummy run
    model_run(model);
    PRINTF("\r\nwhich image to distinguish?    0-9 \r\n");
    for(uint8_t i=0; i<10; i++)
    {
```

```
            print_img((int8_t*)&img[i][0]);
        }
        while(1)
        {
            PRINTF("\r\nwhich image to distinguish?  0-9 \r\n");
            ch = GETCHAR();
            if((ch >'9') || ch < '0')
            {
                continue;
            }
            PRINTF("\r\n");
            mnist(ch-'0');
        }
}
```

编译 weights.h 会报错，原因是少个参数，在 layer[1], layer[4], layer[7],需要加个 dilation(1,1)参数在 stride(1, 1)之后，这样编译就通过了。

7 结果，打开串口软件，一开始会打印出各种手写体数字的图片，然后输入几，就会识别哪张图片。输入的数字，不代表图片数字就是几，图片是随机的。

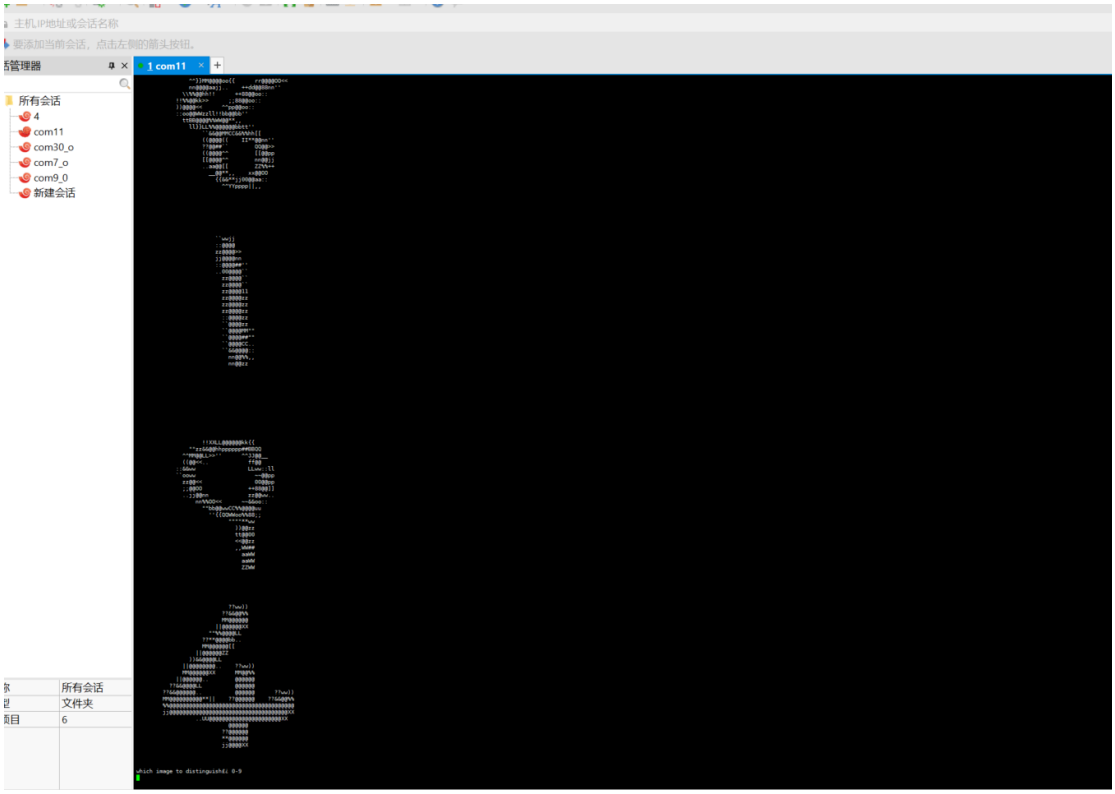

图 2 显示所有手写体数字

串口输入 8 之后，识别的就是这个手写体的‘9’

图 3 识别手写体 9

Truth label 对应的就是 image.h 里的 IMG9_LABLE，Predicted label 则是预测结果

三、训练

通过以上步骤，我们就实现了简单的手写体数字识别，下面要介绍 weights.h 这里的权重，模型是怎么训练出来的？这里的图片数据都是来自于 mnist 数字集，我们怎样才能自己写个数字，让 mcu 去识别？

1 在 nnom-master\examples\mnist-simple 下，有个 mnist_simple.py，你需要运行它，就可以生成 weights.h 和 image.h，运行这个需要安装 tnsorflow，keras 等等，当你运行时候缺什么就用 pip 命令装什么。

这个网络运行大概过程，如图

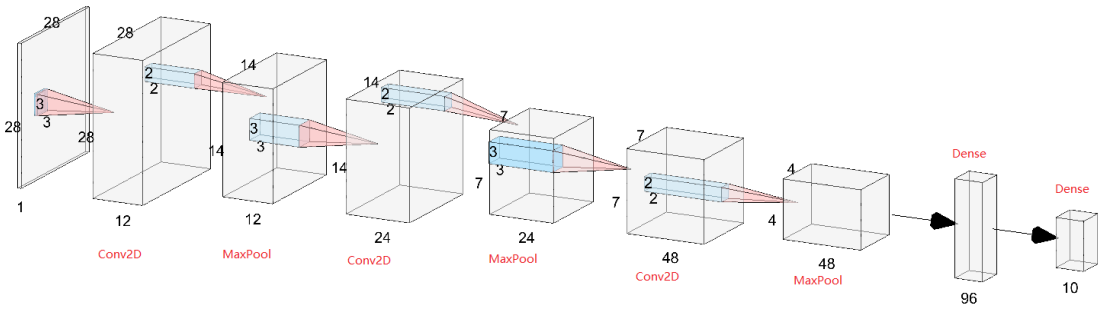

图 4 CNN

Conv2D，卷积操作，MaxPool 池化，dense 激活函数。

卷积操作意义在于提取这个图片的特征，池化有点像压缩数据，这样可以降低运行需要空间。

28x28 输入最后输出一个 1x10 矩阵，代表 0-9 的可能性。

2 我们可以用 win 自带的 paint 程序，将画布调成 28x28，在上面写上数字，保存成 png 格式，我写了一个 4

图 5 手写体‘4’

然后将代码改成这样

```
nnom_model_t *model;
uint8_t temp[28*28]={0};
const char codeLib[] = "@B%8&WM#*oahkbdpqwmZO0QLCJUYXzcvunxrjft/\\|()1{}[]?-_+~<>i!ll;:,\"^`'.    ";
/*******************************************************************************
 * Code
 ******************************************************************************/
void print_img(int8_t * buf)
{
    for(int y = 0; y < 28; y++)
    {
        for (int x = 0; x < 28; x++)
        {
            int index =   69 / 127.0 * (127 - buf[y*28+x]);
            if(index > 69) index =69;
            if(index < 0) index = 0;
            PRINTF("%c",codeLib[index]);
            PRINTF("%c",codeLib[index]);
        }
        PRINTF("\r\n");
    }
```

```
}


void mnist_pic(uint8_t *temp)
{
    float prob;
    uint32_t predic_label;
    PRINTF("\nprediction start.. \r\n");

    // copy data and do prediction
    memcpy(nnom_input_data, (int8_t*)temp, 784);
    nnom_predict(model, &predic_label, &prob);

    //print original image to console
    print_img((int8_t *)temp);
    PRINTF("\r\nPredicted label: %d\n", predic_label);
    PRINTF("\r\nProbability: %d%%\n", (int)(prob*100));
}

int main(void)
{
    /* Board pin, clock, debug console init */
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();
    /* Print a note to terminal */
    model = nnom_model_create();
    // dummy run
    model_run(model);
    while(1)
    {
        PRINTF("\r\n Send picture by serial\r\n");
        DbgConsole_ReadLine(temp,784);
        PRINTF("\r\n Got picture\r\n");
        mnist_pic(temp);
    }
}
```

3 然后使用附件的 pic2mnist.py，用 cmd 运行这个脚本，输入‘python pic2mnist.py 1.png’，1.png 就是要解析的图片，接着会生成一个 content.txt 文件，里面就是这个图片的数据，把这个里面数据，用串口发送给 MCU,注意发送要勾选 Send As Hex，同样的会先显示手写体图片，然后识别图片。

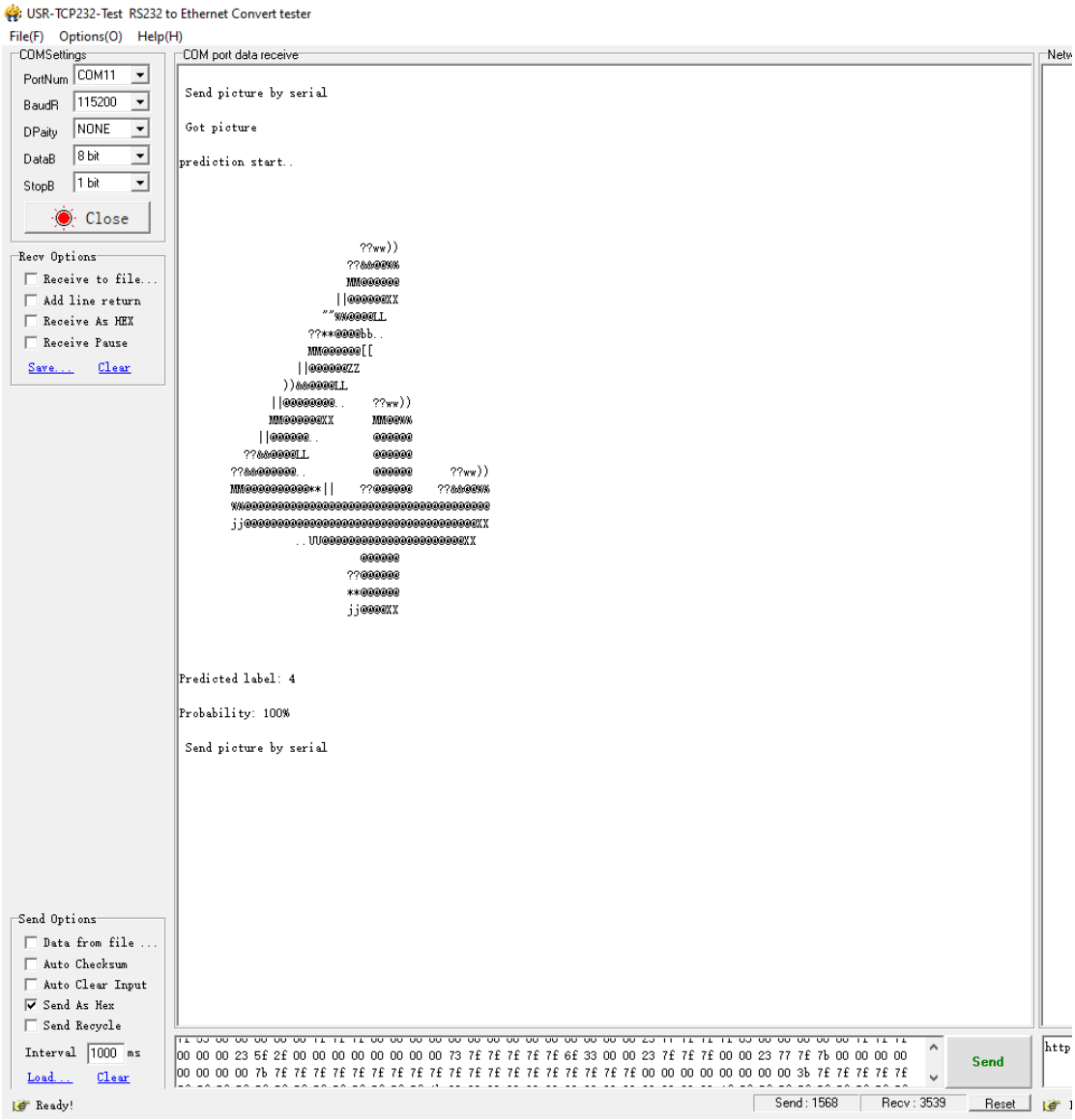

图 6 串口识别手写体

可以看到识别出了‘4’