在K64上运行卷积神经网络

目录

[一、简介 1](#_Toc61863911)

[二、实验 1](#_Toc61863912)

[三、训练 5](#_Toc61863913)

# 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

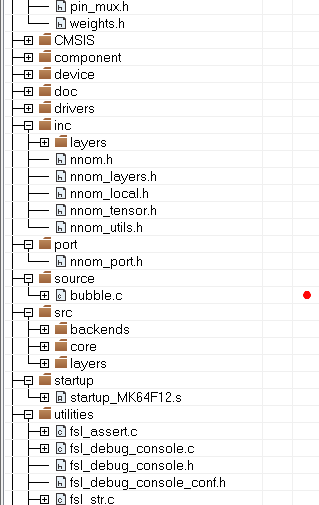


图1

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!lI;:,\"^`'. ";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 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。

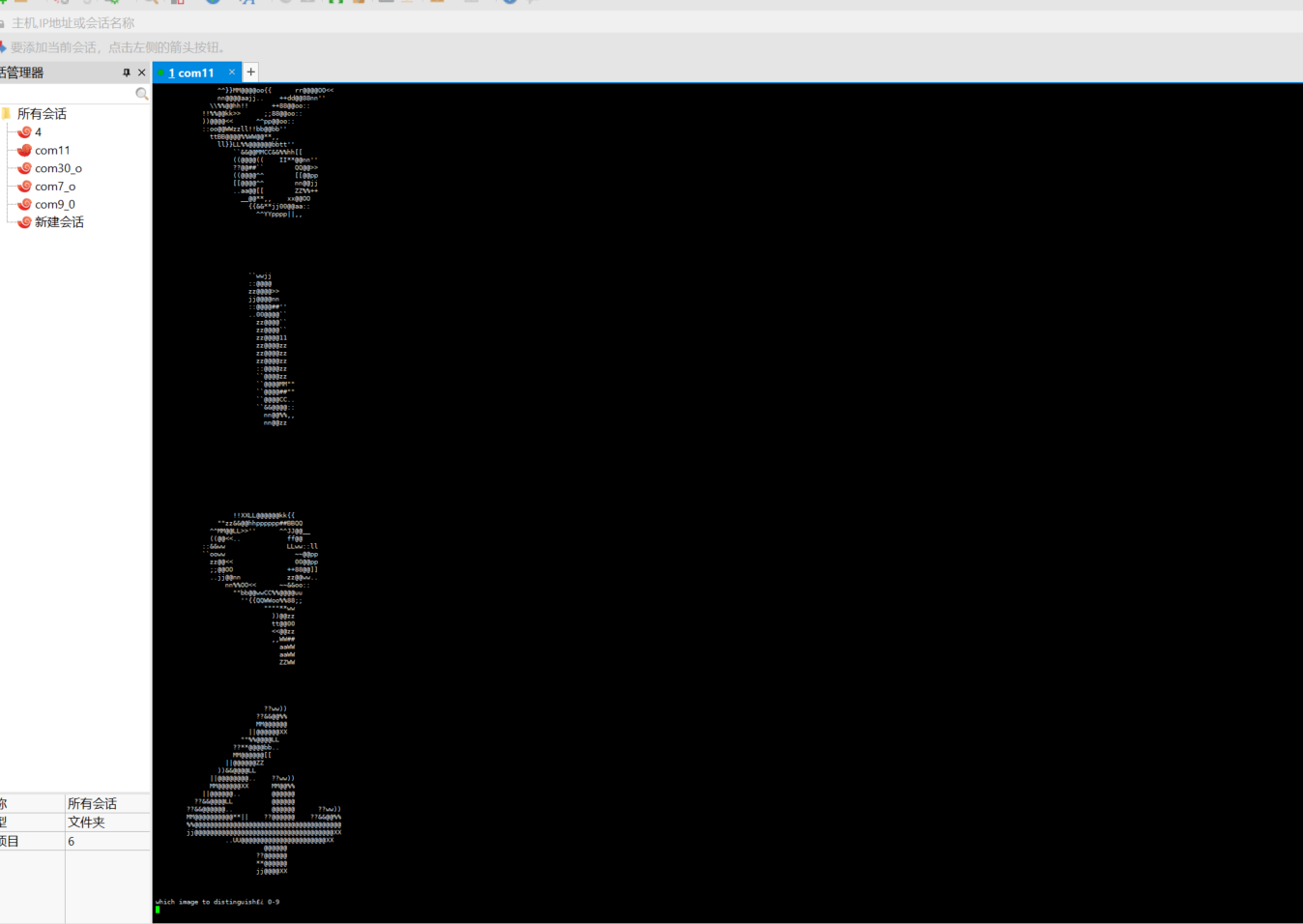


图2 显示所有手写体数字

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tensorflow，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

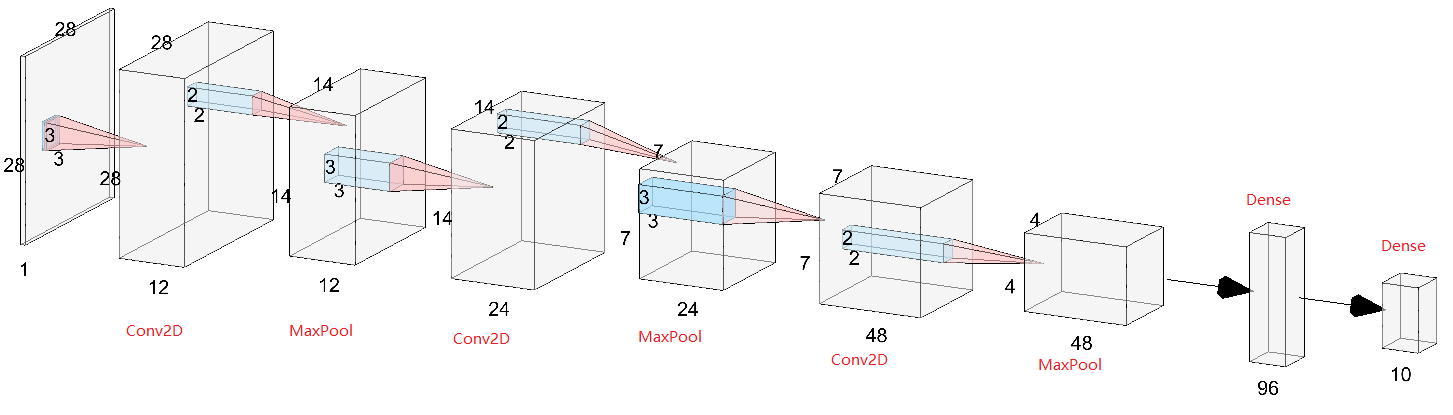


图4 CNN

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!lI;:,\"^`'. ";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 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。

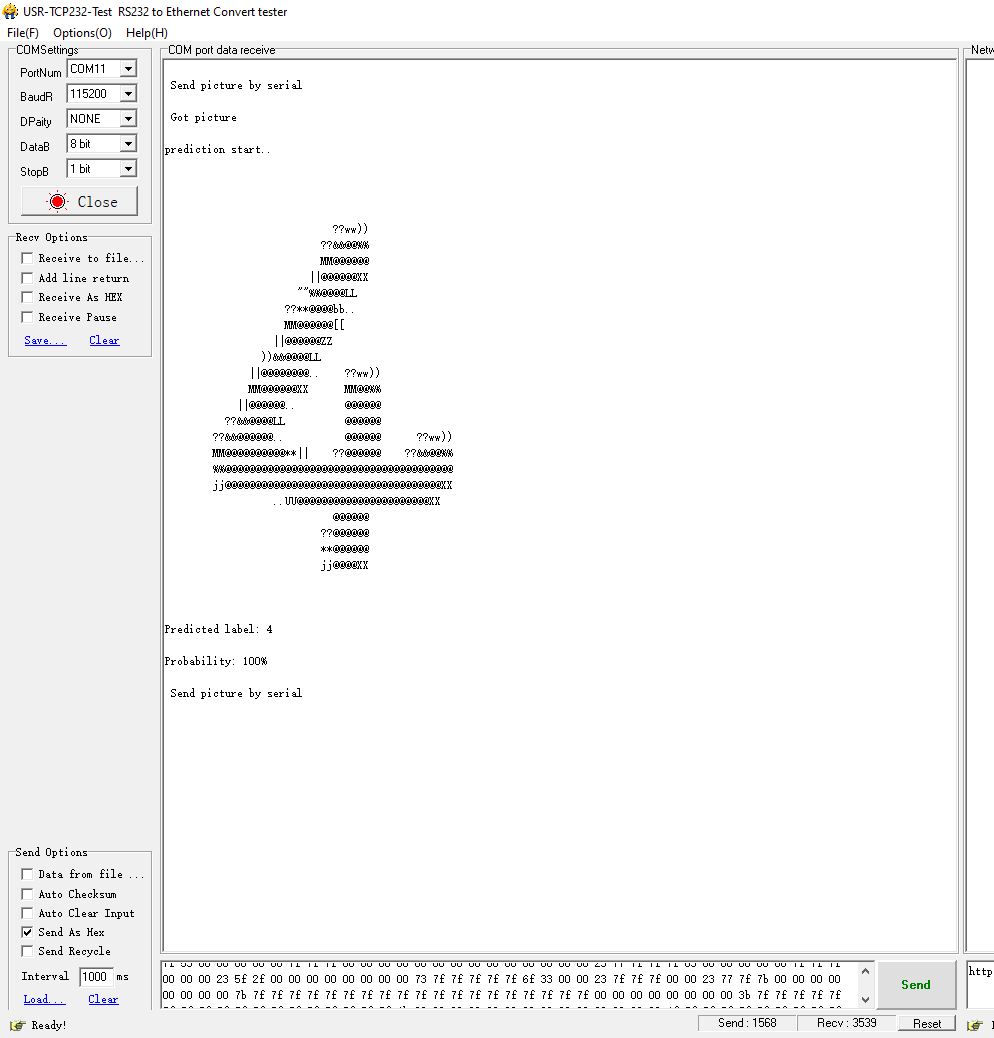


图6 串口识别手写体

可以看到识别出了‘4’