实验三　手写数字识别

RT-AK 教育套件实验手册

WWW.RT-THREAD.ORG

Monday 22nd November, 2021

目录

第 1 章

实验介绍和目的

1.1 实验介绍

本实验是基于 Tensorflow 训练第一个 AI 模型：MNIST 手写数字识别模型。

MNIST 手写数字识别模型的主要任务是：

输入一张手写数字的图像，然后识别图像中手写的是哪个数字。

该模型的目标明确、任务简单，数据集规范、统一，数据量大小适中，在普通的 PC 电脑上都能训练和识别，堪称是深度学习领域的 “Hello World!”，学习嵌入式 AI 的入门必备模型。

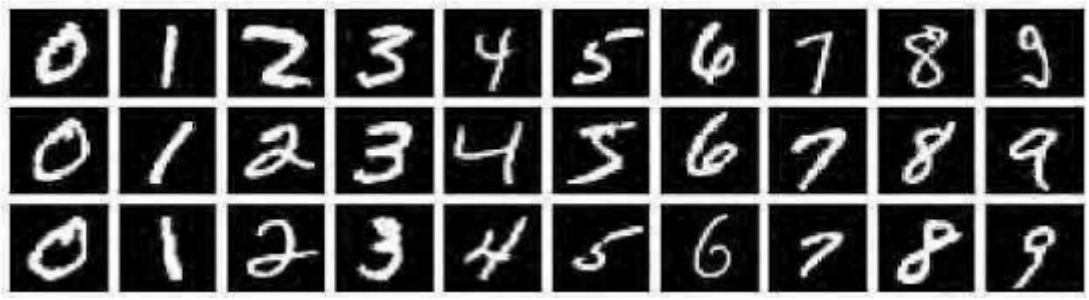

图 1.1: *数据样本示例*

1.2 实验目的

1. 掌握基于 Tensorflow 训练手写字符识别模型
2. 掌握 RT-AK 一行命令部署 AI 模型的使用
3. 完成首次嵌入式 AI 开发：输入一张照片并成功推理一次模型

第 2 章

实验器材

1. 上位机（电脑）
2. EgdeAI 实验板

第 3 章

实验步骤

确保环境安装无问题之后做后续实验，环境安装请参考实验二

3.1 AI 模型训练

0. 打开本实验文件夹，打开 Jupyter 编辑器，打开 lab1_mnist_training.ipynb 文件

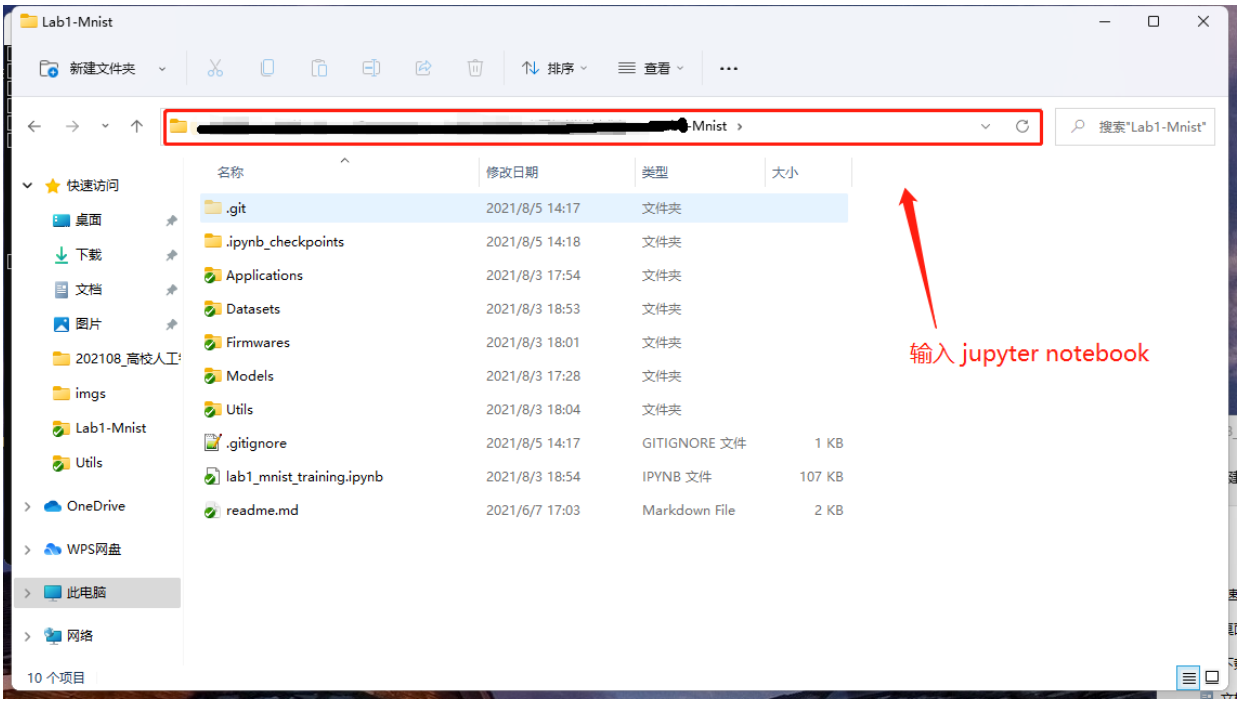

图 3.1: *实验文件夹*

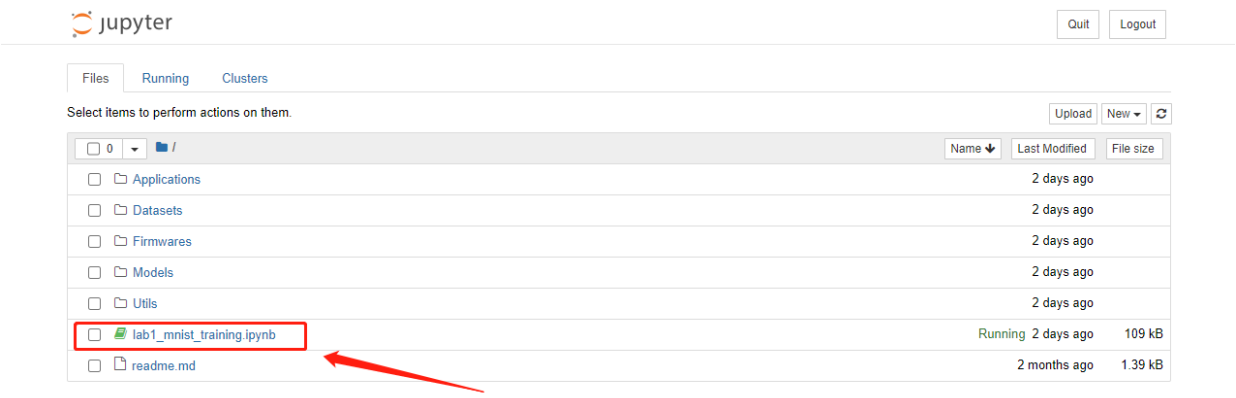

图 3.2: *神经网络训练代码*

以下是模型训练代码的示例版，内容摘自 lab1_mnist_training.ipynb 文件

1. 导入库

```
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras import models, layers, datasets, utils
from pathlib import Path
import matplotlib.pyplot as plt

print(f"The tensorflow version is : {tf.__version__}\n")
print(f"The numpy version is : {np.__version__}")

# 当一个单元格代码写完的时候，点一下上面的Run运行按钮，运行代码
```

```
# 在当前路径下创建 model 文件夹，存放模型
# exist_ok 如果文件夹已经存在，不会报错，如果没有，则会创建
model_path = Path("Models")
model_path.mkdir(exist_ok=True)
```

2. 准备数据集

载入并准备好MNIST 数据集 http://yann.lecun.com/exdb/mnist/。训练集包含 60,000 个示例图像，测试集包含 10,000 个示例图像

此处的数据集文件存放在国外网站，这里已经提前下好到 ./Datasets/training_data/mnist.npz，直接使用即可

其中，x_train 是训练数据集，y_train 是训练数据集的对应标签。x_test，y_test 是测试集的样本和标签。

```
data_path = Path("Datasets")
(x_train, y_train), (x_test, y_test) = datasets.mnist.load_data(data_path.resolve()/
    'train_data/mnist.npz')  # resolve 变成绝对路径
```

```
# Normalize data，归一化，将图像的像素值都处理到[0,1]范围
x_train, x_test = x_train / 255., x_test / 255.
print(f"x_train shape : {x_train.shape}")

# 扩展一个维度，二维卷积需要用的输入数据是 NHWC，分别是number of batch, height,
    width, channels
x_train, x_test = tf.expand_dims(x_train, -1), tf.expand_dims(x_test, -1)
print(f"add dim x_train shape : {x_train.shape}")
```

3. 搭建神经网络模型

两个卷积层 + （Flattern 操作）+ 两个全连接层，最后一层输出预测的 10 个概率值

```
# build network
tf.keras.backend.clear_session()

model = models.Sequential()
# conv1
model.add(layers.Conv2D(input_shape=(28, 28, 1), filters=4,
    kernel_size=(3, 3), activation='relu', name='conv1'))
model.add(layers.MaxPool2D(pool_size=(2,2), name='pool1'))

# conv2
model.add(layers.Conv2D(filters=8, kernel_size=(3, 3),
                        activation='relu', name='conv2'))
model.add(layers.MaxPool2D(pool_size=(2,2), name='pool2'))

# flattern
model.add(layers.Flatten(name='flatten'))

# FC1
model.add(layers.Dense(128, activation='relu', name='FC1'))

# FC2
model.add(layers.Dense(10, activation='softmax', name="FC2"))

model.summary()  # 显示网络结构图
```

```
# 网络结构显示

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 26, 26, 4)         40
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 13, 13, 4)         0
_________________________________________________________________
```

```
conv2 (Conv2D)               (None, 11, 11, 8)         296
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 5, 5, 8)           0
_________________________________________________________________
flatten (Flatten)            (None, 200)               0
_________________________________________________________________
FC1 (Dense)                  (None, 128)               25728
_________________________________________________________________
FC2 (Dense)                  (None, 10)                1290
=================================================================
Total params: 27,354
Trainable params: 27,354
Non-trainable params: 0
_________________________________________________________________
```

4. 训练

   compile() 方法：指定损失、指标和优化器

   要使用 fit() 训练模型，您需要指定损失函数、优化器以及一些要监视的指标（可选）。

   将它们作为 compile() 方法的参数传递给模型.

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy',])
history = model.fit(x_train, y_train, batch_size=128, epochs=10)
```

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=32)
print("test loss, test acc:", results)
```

```
查看每次训练的损失和精度。
```

```
def plot_metric(history, metric):
    train_metrics = history.history[metric]
    epochs =range(1, len(train_metrics) + 1)
    plt.plot(epochs, train_metrics, 'ro--')
    # 相关属性
    plt.title('Training ' + metric)
    plt.xlabel('Epochs')
    plt.ylabel(metric)
    plt.legend(['train_'+metric])
    plt.show()

plot_metric(history, 'loss')
plot_metric(history, 'accuracy')
```

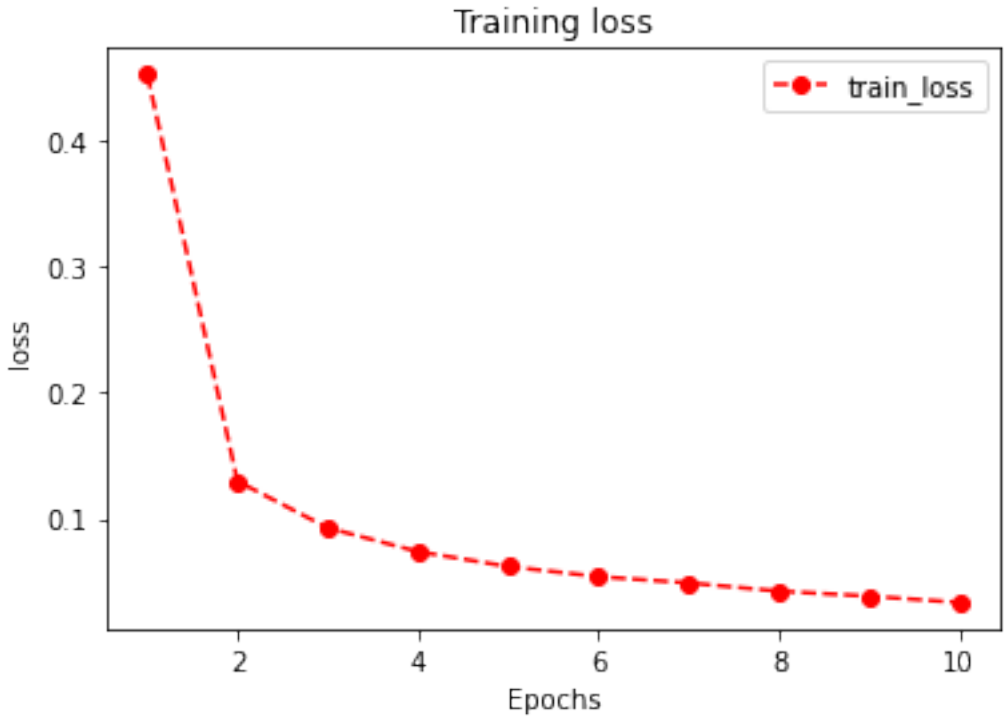

图 3.3: *Training loss*

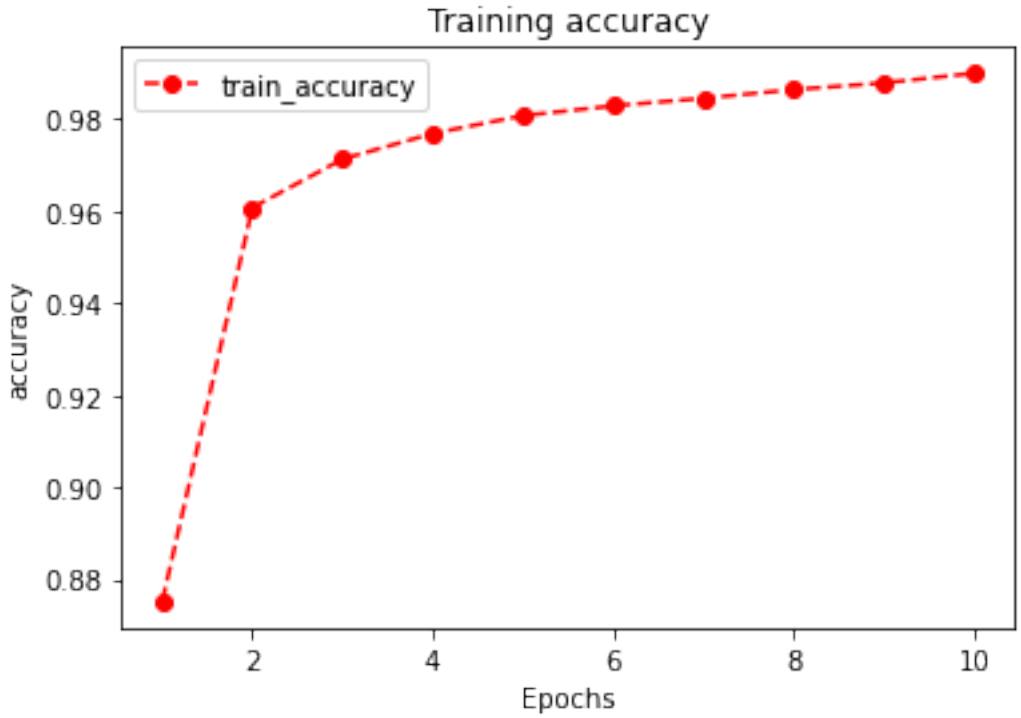

图 3.4: *Training accuracy*

模型预测:

```
验证 model 是否成功训练，用 test 数据集进行推理（使用模型）
```

```
# 获取测试集的索引为10的图片
img = x_test[10]
# 模型推理
y_pred = model.predict(tf.expand_dims(img, axis=0))
# 获取概率最高的元素的索引，np.argmax 返回参数 list 中值最大的元素的索引
p = np.argmax(y_pred)
plt.imshow(img, cmap="gray")
plt.show()

print(f"model inference ouput:\n {y_pred}")
print(f"predict : {p}")
```

5. 模型保存

```
# Path 重载了 / 操作符，因此可以直接这样连接路径
keras_file = model_path/'mnist.h5'
model.save(keras_file, save_format="h5")
```

6. 加载保存的模型并预测

```
# load model and test
model_restore = models.load_model(keras_file)

# 验证模型，没有任何返回则模型加载成功
np.testing.assert_allclose(model.predict(x_test), model_restore.predict(x_test))
```

```
# 找五张图片做测试
for test_image in x_test[:5]:
    y_pred = model_restore.predict(tf.expand_dims(test_image, axis=0))
    p = np.argmax(y)    # 获取概率最高的元素的索引
    plt.imshow(test_image, cmap="gray")
    plt.show()
    print(f"model inference ouput:\n {y_pred}")
    print(f"predict : {p}")
```

7. 模型转成 **RT-AK** 部署所支持的格式

```
# keras 模型转 tflite，后者模型会更小一点，算子支持更多
model = tf.keras.models.load_model(keras_file)
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

tflite_file = model_path/ "mnist.tflite"
tflite_file.write_bytes(tflite_model)
```

当所有的代码执行完成之后，会在当前文件夹下的 Models 得到两个模型文件：

文件	描述
Models/mnist.h5	使用 Tensorflow 中 Keras API 训练所得的模型文件
Models/mnist.tflite	RT-AK 所支持的模型文件格式

其中，两个模型的内部保存的都是网络结构和权重数据。

使用 Netron 查看网络结构（双击 mnist.tflite 文件即可）：

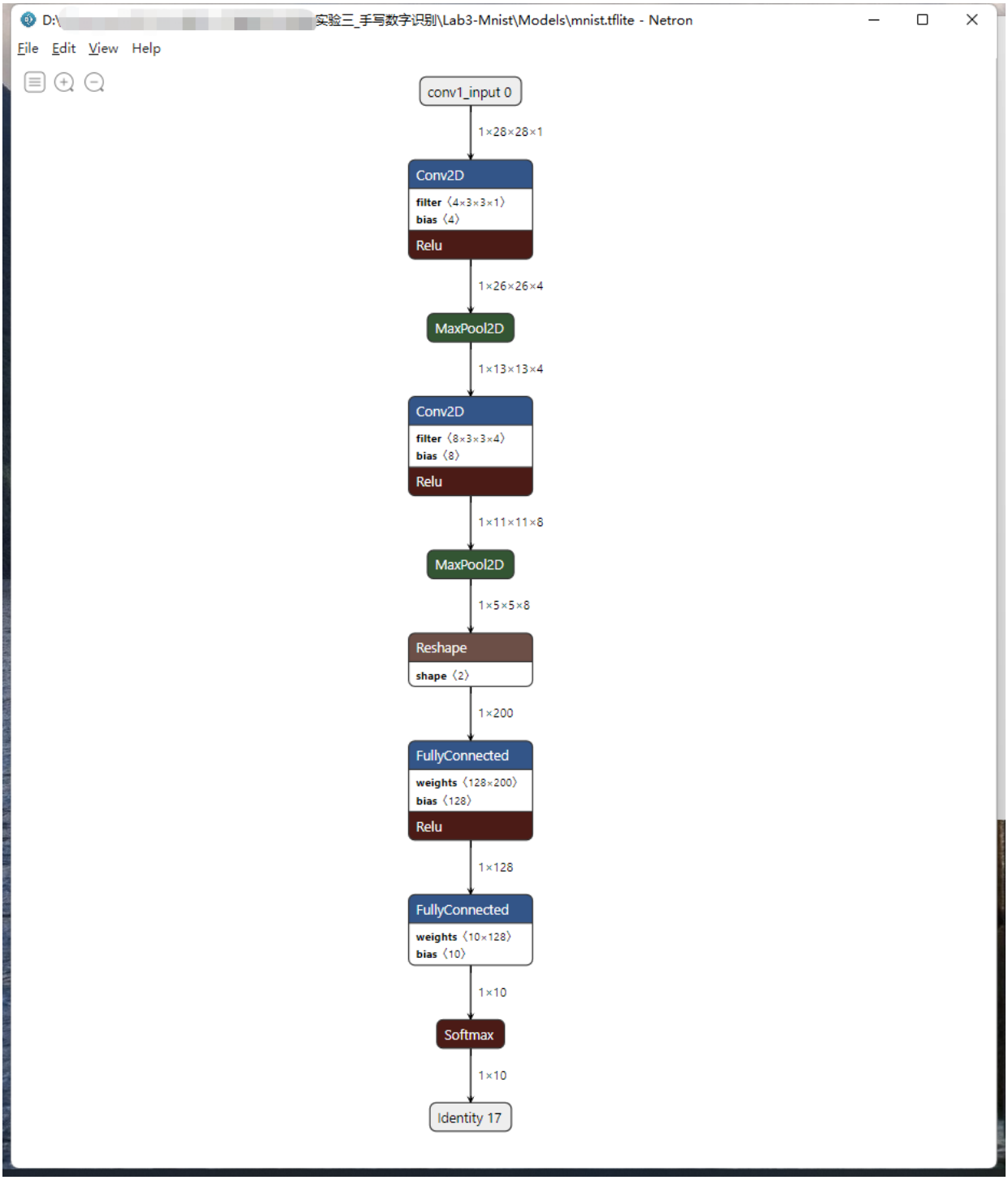

图 3.5: *网络结构*

3.2 模型信息

- 神经网络模型：`Models/mnist.h5` && `Models/mnist.tflite`
- 模型的输入是 NCHW: [number of batch * channe * heigh * width]，1*1*28*28，N=1，C=1，H=28，W=28，数据类型：uint8

  N 代表了一张图片

如果是灰度图，channel 是 1 通道，如果是 RGB 图，channel 是 3 通道

- 模型的输出是 1*10，数据类型：float32

3.3 模型部署

在 RT-AK/rt_ai_tools 路径下打开 Windows 终端

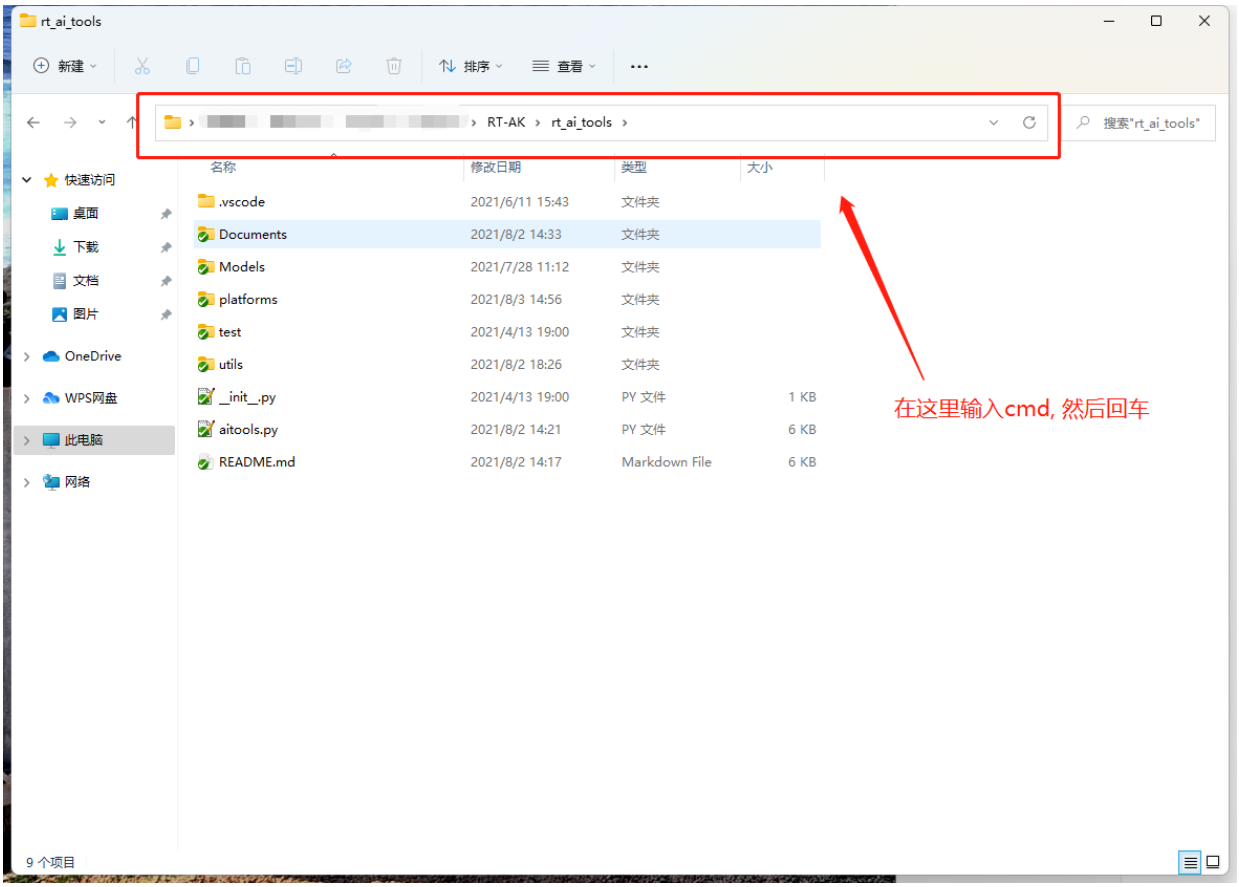

图 3.6: *打开 Windows 终端*

输入以下命令：

```
# 量化为 uint8，使用 KPU 加速，量化数据集为图片
$ python aitools.py --project=<your_project_path> --model=<your_model_path> --
    model_name=mnist --platform=k210 --dataset=<your_val_dataset>
```

示例：

```
# 示例(量化模型，图片数据集)
$ python aitools.py --project="D:\Project\K210_Demo\k210-test" --model="./Models/
    mnist.tflite" --model_name=mnist --platform=k210 --dataset="xxx\Lab1-Mnist\
    Datasets\quantize_data"
```

其中，--project 是你的目标工程路径，--model 是你的模型路径（这里使用的是 RT-AK 自带的模型文件），--model_name 是转化的模型文件名，--platform 是指定插件支持的目标平台为 K210，--dataset

是模型量化所需要用到的数据集。

更多详细的参数信息请看文档: RT-AK\rt_ai_tools\platforms\plugin_k210\README.md。

当部署成功之后，目标工程文件会多出几个文件：

文件	描述
rt_ai_lib/	RT-AK Libs，模型推理库
applications/mnist_kmodel.c	kmodel 的十六进制储存
applications/rt_ai_mnist_model.c	与目标平台相关的信息
applications/rt_ai_mnist_model.h	模型相关信息

同时，在 RT-AK\rt_ai_tools\platforms\plugin_k210 路径下会生成两个文件

文件	描述
mnist.kmodel	k210 所支持的模型格式
convert_report.txt	tflite 模型转成 kmodel 格式的缓存信息

如果不想生成上述两个文件，可以在模型部署的时候命令行参数末尾加上：--clear

注意：

1、RT-AK 部署成功后不会产生应用代码，比如模型推理代码，需要手工编写，详见 “3. 嵌入式 AI 模型应用”

2、在应用开发过程中，请遵守 RT-Thread 的编程规范以及 API 使用标准

3.4 嵌入式 AI 模型应用

使用 RT-AK 将训练好的 tflite 模型成功部署到工程之后，我们就可以开始着手编写应用层代码来使用该模型。本节的所有代码详见文件 Lab3-Mnist\Applications

3.4.1 代码流程

系统内部初始化：

- 系统时钟初始化

RT-AK Lib 模型加载并运行：

- 注册模型（代码自动注册，无需修改）
- 找到注册模型
- 初始化模型，挂载模型信息，准备运行环境

- 运行（推理）模型
- 获取输出结果

Mnist 业务逻辑层：

- 找出输出最大值的索引

3.4.2 核心代码说明

```
// main.c
/* Set CPU clock */
sysctl_clock_enable(SYSCTL_CLOCK_AI);  // 使能KPU时钟（系统时钟初始化）
...

// 注册模型的代码在 rt_ai_mnist_model.c 文件下的第31行，代码自动执行
// 模型的相关信息在 rt_ai_mnist_model.h 文件
/* AI modol inference */
mymodel = rt_ai_find(RT_AI_MNIST_MODEL_NAME);  // 找到注册模型,rt_ai_mnist_model.h
    文件中有模型相关信息声明，命名格式 RT_AI_<model_name>_MODEL_NAME
if (rt_ai_init(mymodel, (rt_ai_buffer_t *)kpu_img) != 0)  // 初始化模型，传入输入数
    据
...
if (rt_ai_run(mymodel, ai_done, NULL) != 0)    // 模型推理一次
...
output = (float *)rt_ai_output(mymodel, 0);  // 获取模型输出结果

/* 对模型输出结果进行处理，该实验是Mnist，输出结果为10个概率值，选出其中最大概率即可
    */
for(int i = 0; i < 10 ; i++)
{
    if(output[i] > scores  && output[i] > 0.2)
    {
        prediction = i;
        scores = output[i];
    }
}
```

第 4 章

编译烧录

4.1 编译

参考 lab2-env 教程中 Studio 使用方法。新建->RT-Thread项目->基于开发板->K210-RT-DRACO 输入工程目录和工程名，新建基于开发板的模板工程。将实验代码复制到 application 文件中替换原文件代码。点击 编译。会在你的工程根目录下生成一个 rtthread.bin 文件，然后参考下面的 烧录方法。其中 rtthread.bin 需要烧写到设备中进行运行。

4.2 烧录

连接好串口，点击 Studio 中的下载图标进行下载，详细可参考lab-env2

或者

使用 K-Flash 工具进行烧写 bin 文件。

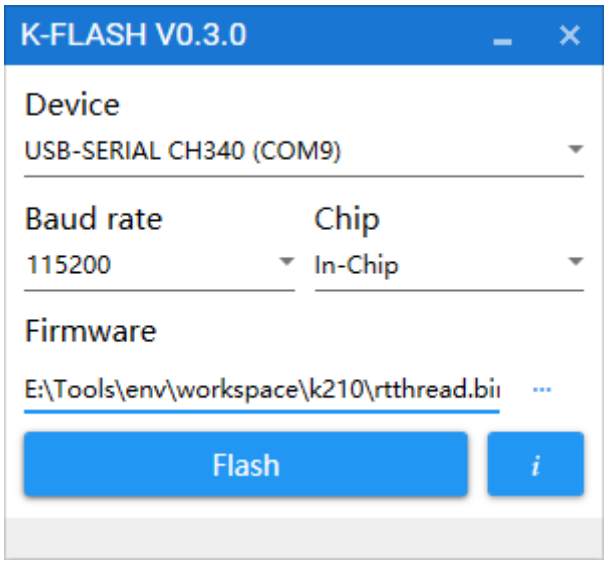

图 4.1: *K-Flash*

第 5 章

实验现象

5.1 实验现象

如果编译 & 烧写无误，若使用 K-Flash 下载，下载完成会自动打开 Windows 终端，自动连接实验板。Studio 下载需在 Studio 中打开串口终端。系统启动后，我们的程序会自动运行，会在终端串口显示预测的结果。在 LCD 上会显示被预测图片和预测类别结果。

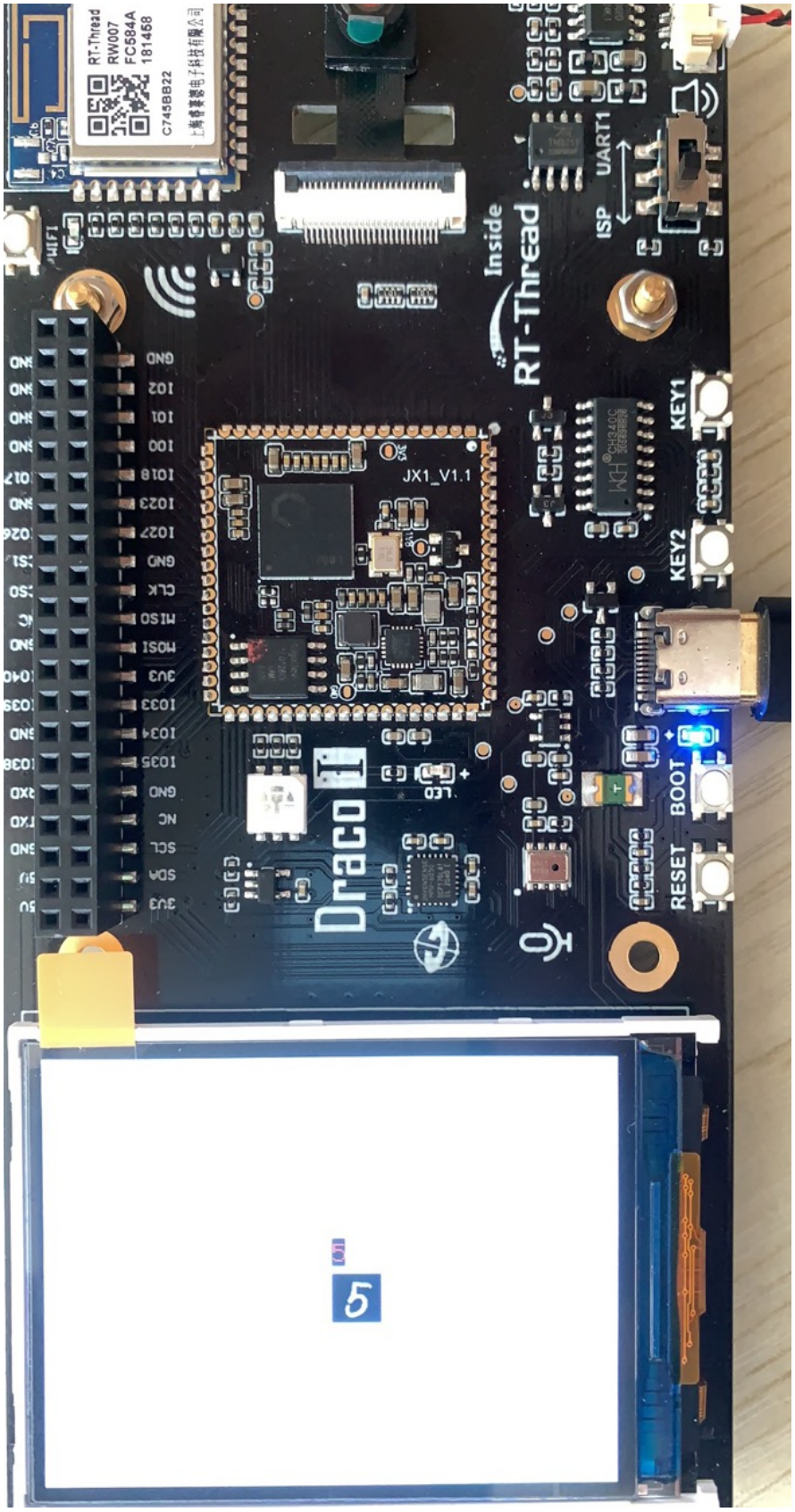
RT-Thread
RW007
FC584A
181458
C745BB22
WIFI
Inside
RT-Thread
UART1
ISP
KEY1
KEY2
RESET
BOOT
JX1_V1.1
Draco
5

图 **5.1:** *实验结果*

5.2 模型在部署阶段使用的量化数据获取说明

获取量化数据集的文件为 `Utils/create_quantize_data.py`

使用方式：

```
cd Utils
python create_quantize_data.py
```

代码文件流程简要说明：

1. 读取 `Datasets/train_data` 路径下的 `.npz` 数据，
2. 如果已经存在量化图片数据集，则返回，否则继续执行
3. 随机抽取 200 图片，保存

如何获取 test_data 中的样本数据

K210 模型推理时的输入数据格式是 CHW (Color Channel-Height-Width)，即 `image[c][h][w]`，所以需要将图片保存成 CHW 格式，并且拉平成一维数据，

使用的是 `utils/save_chw_img.py` ，该文件中有使用示例

> Ps1: 一般读取到的图片数据是：HWC 三维数据，即 `image[h][w][c]`，如果是灰度图，C 为 1，如果是 RGB 图，C 为 3
>
> Ps2：MNIST 数据集源于扫描识别信封上的邮政编码需求，按日常书写方式用笔书写数字后拍照再缩小形成的图片，或直接用绘图软件画数字获得的图片，其特征与训练数据有较大区别，因此识别效果不佳。

如何使用该文件示例：

```
cd Utils
python save_chw_img.py ../Datasets/quantize_data/1.png ../Applications/test_data/
    new_img2_chw.h 28x28
```

代码文件流程简要说明：

1. 读取图片

   ```
   image_raw = cv2.imread(image_path)
   ```

2. 转成灰度图（训练的时候是灰度图训练，和训练的输入格式保持一致）

   ```
   image = cv2.cvtColor(image_raw, cv2.COLOR_BGR2GRAY)
   ```

3. 图片尺寸缩放，这里用的是 Opencv 的 resize 函数

   ```
   image = cv2.resize(image, shape)
   ```

4. 保存为文件

```
with open(dataset_h, "w+") as f:
    print(f"#ifndef _{dataset_h.stem.upper()}_H_\n#define _{dataset_h.stem.upper
        ()}_H_\n", file=f)
    print(f"// {image_path} {shape}, HW", file=f)
    print(f"const static uint8_t {dataset_h.stem.upper()}[] __attribute__((
        aligned(128))) = {'{'}", file=f)
    # print(", ".join([str(i) for i in image.flatten()]), file=f)
    print(", ".join(map(lambda i: str(i), image.flatten())), file=f)
    print("};\n\n#endif", file=f)
```

详细的代码请看文件

需要做两个操作：

1. 图片格式转换：RGB888 –> RGB565

RGB565用unsigned short 16位字节存储

R7	R6	R5	R4	R3	G7	G6	G5	G4	G3	G2	B7	B6	B5	B4	B3

2. 图片尺寸缩放（本实验未使用）

原因：

1. k210 LCD 显示的图片格式：RGB565，一般图片保存格式：RGB888
2. k210 LCD 显示的图片尺寸是 320*240，输入超过该尺寸的需要做尺寸缩放，否则无法显示全部图片。但是本实验所用的图片大小均小于 320*240，所以不用缩放，原尺寸显示即可

步骤：

1. 代码文件：`Utils/png2rgb565.py`
2. 使用：

```
cd Utils
# python png2rgb565.py 图片路径 保存文件路径
python png2rgb565.py ../Datasets/quantize_data/1.png ../Applications/test_data/
    new_img2_show.h
```

3. 在 `课后拓展/代码/main.c` 中修改相应的显示图片的变量即可，示例代码中该变量为 `lcd_draw_picture(0, 0, 28, 28, (rt_ai_uint16_t *)IMG9);` 中的 `IMG9`
4. 以上是 python 版本，下面是 c 语言版本的，来自网络资料：https://www.xuebuyuan.com/935987.html

```
#define RGB888_RED      0x00ff0000
#define RGB888_GREEN    0x0000ff00
#define RGB888_BLUE     0x000000ff
```

```
#define RGB565_RED      0xf800
#define RGB565_GREEN    0x07e0
#define RGB565_BLUE     0x001f

unsigned short RGB888ToRGB565(unsigned int n888Color)
{
 unsigned short n565Color = 0;

 // 获取RGB单色，并截取高位
 unsigned char cRed   = (n888Color & RGB888_RED)   >> 19;
 unsigned char cGreen = (n888Color & RGB888_GREEN) >> 10;
 unsigned char cBlue  = (n888Color & RGB888_BLUE)  >> 3;

 // 连接
 n565Color = (cRed << 11) + (cGreen << 5) + (cBlue << 0);
 return n565Color;
}

unsigned int RGB565ToRGB888(unsigned short n565Color)
{
 unsigned int n888Color = 0;

 // 获取RGB单色，并填充低位
 unsigned char cRed   = (n565Color & RGB565_RED)    >> 8;
 unsigned char cGreen = (n565Color & RGB565_GREEN)  >> 3;
 unsigned char cBlue  = (n565Color & RGB565_BLUE)   << 3;

 // 连接
 n888Color = (cRed << 16) + (cGreen << 8) + (cBlue << 0);
 return n888Color;
}
```

第 6 章

API 使用说明

6.1 嵌入式 AI 开发 API 文档

```
rt_ai_t rt_ai_find(const char *name);
```

Paramaters	Description
name	注册的模型名
Return	–
rt_ai_t	已注册模型句柄
NULL	未发现模型

描述: 查找已注册模型

```
rt_err_t rt_ai_init(rt_ai_t ai, rt_aibuffer_t* work_buf);
```

Paramaters	Description
ai	rt_ai_t 句柄
work_buf	运行时计算所用内存
Return	–
0	初始化成功
非 0	初始化失败

描述: 初始化模型句柄, 挂载模型信息, 准备运行环境.

```
rt_err_t rt_ai_run(rt_ai_t ai, void (*callback)(void * arg), void *arg);
```

Paramaters	Description
ai	rt_ai_t 模型句柄

Paramaters	Description
callback	运行完成回调函数
arg	运行完成回调函数参数
Return	–
0	成功
非 0	失败

描述: 模型推理计算

```
rt_aibuffer_t rt_ai_output(rt_ai_t aihandle,rt_uint32_t index);
```

Paramaters	Description
ai	rt_ai_t 模型句柄
index	结果索引
Return	–
NOT NULL	结果存放地址
NULL	获取结果失败

描述: 获取模型运行的结果, 结果获取后.

`rt_ai_libs/readme.md` 文件中有详细说明

6.2 LCD API 说明手册

```
/**
 * @fn void lcd_init(void);
 * @brief LCD初始化
*/
void lcd_init(void);
/**
 * @fn void lcd_clear(uint16_t color);
 * @brief 清屏
 * @param color 清屏时屏幕填充色
*/
void lcd_clear(uint16_t color);
/**
 * @fn void lcd_set_direction(lcd_dir_t dir);
 * @brief 设置LCD显示方向
 * @param dir 显示方向参数
*/
void lcd_set_direction(lcd_dir_t dir);
```

```
/**
 * @fn void lcd_set_area(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2);
 * @brief 设置LCD显示区域
 * @param x1 左上角横坐标
 * @param y1 左上角纵坐标
 * @param x2 右下角横坐标
 * @param y2 右下角纵坐标
*/
void lcd_set_area(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2);
/**
 * @fn void lcd_draw_point(uint16_t x, uint16_t y, uint16_t color);
 * @brief 画点
 * @param x 横坐标
 * @param y 纵坐标
 * @param color 颜色
*/
void lcd_draw_point(uint16_t x, uint16_t y, uint16_t color);

/**
 * @fn void lcd_draw_string(uint16_t x, uint16_t y, char *str, uint16_t color);
 * @brief 显示字符串
 * @param x 显示位置横坐标
 * @param y 显示位置纵坐标
 * @param str 字符串
 * @param color 字符串颜色
*/
void lcd_draw_string(uint16_t x, uint16_t y, char *str, uint16_t color);
/**
 * @fn void lcd_draw_picture(uint16_t x1, uint16_t y1, uint16_t width, uint16_t
     height, uint32_t *ptr);
 * @brief 显示图片
 * @param x1 左上角横坐标
 * @param y1 左上角纵坐标
 * @param width 图片宽
 * @param height 图片高
 * @param ptr 图片地址
*/
void lcd_draw_picture(uint16_t x1, uint16_t y1, uint16_t width, uint16_t height,
    uint32_t *ptr);
/**
 * @fn void lcd_draw_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,
     uint16_t width, uint16_t color);
 * @brief 画矩形
 * @param x1 左上角横坐标
 * @param y1 左上角纵坐标
 * @param x2 右下角横坐标
 * @param y2 右下角纵坐标
 * @param width 线条宽度(当前无此功能)
 * @param color 线条颜色
```

```
*/
```