Fibocom 广和通
完 美 无 线 体 验

初学者入门手写数字识别案例

文档版本：V1.0
更新时间：2025 年 10 月 15 日

适用型号

序列	文档版本	适用型号	更新说明
1	V1.0	SC171 开发套件第三代	NA

目录

1 引言

本指南以手写数字识别为例，详细演示如何在 SC171 开发套件 V3 上通过 Fibocom AI Stack 实现高效的深度学习模型推理。通过结合专为边缘设备优化的 Fibo AI Stack 推理框架，本指南提供了一套完整的解决方案，覆盖从模型搭建、模型训练、模型推理到结果的全流程，旨在为边缘端 AI 部署提供标准化范例。

2 数字识别模型搭建

以下显示黑色背景的代码均为在电脑端的 LeNet.py

2.1 数据准备

- 从 Keras 库中引入手写数字数据集 MNIST，他是一个包含 60000 个训练样本和 10000 个测试样本的数据集。使用 load_data()函数将 MNIST 数据集加载到程序中，并将数据集分为训练集和测试集，其中 train_images、train_labels 为训练集，test_images、test_labels 为测试集。
- 由于神经网络只接受数值型数据，所以需要将手写数字图像转换为数据张量格式。也就是将每张图像转换为 28×28 的矩阵，并进行归一化处理。

```
from keras.utils import to_categorical
from keras import models, layers
from keras.optimizers import RMSprop
from keras.datasets import mnist
import tensorflow as tf
# 加载数据集
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# 数据处理
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

2.2 搭建神经网络

模型采用了 Sequential 模型

- C1 层是卷积层，包含 6 个特征图，是由 6 个 3x3 的卷积核对输入图像卷积得到。
- S2 层是一个下采样层，包含 6 个特征图，是由 C1 层的特征图经过 2x2 的窗口进行平均池化
- C3 层是卷积层，包含 16 个特征图，是由 16 个 3x3 的卷积核对 S2 进行卷积得到
- S4 是一个下采样层，包含 16 个特征图，是由 C3 层的特征图经过 2x2 的窗口进行平均池化

深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

- C5 是卷积层包含 120 个特征图，是由 120 个 3x3 的卷积核对 S4 进行卷积得到。
- F6 是包含 84 个神经元的全连接层，采用 relu 激活函数
- 最后添加一个分类层，使用 softmax 激活。输出 0-9 是个数字，所以单元数为 10

模型搭建完成后，进行编译模型。使用交叉熵作为损失函数，RMSprop 优化器进行训练，在训练过程中监测模型的精度。

```
# 搭建LeNet网络
def LeNet():
    network = models.Sequential()
    network.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
    network.add(layers.AveragePooling2D((2, 2)))
    network.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
    network.add(layers.AveragePooling2D((2, 2)))
    network.add(layers.Conv2D(filters=120, kernel_size=(3, 3), activation='relu'))
    network.add(layers.Flatten())
    network.add(layers.Dense(84, activation='relu'))
    network.add(layers.Dense(10, activation='softmax'))
    return network
network = LeNet()
network.compile(optimizer=RMSprop(lr=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

2.3 训练网络模型

使用 fit()方法对构建好的神经网络进行训练，epochs 表示训练多少个回合，batch_size 表示每次训练给多大的数据。

```
# 训练网络，用fit函数，epochs表示训练多少个回合， batch_size表示每次训练给多大的数据
network.fit(train_images, train_labels, epochs=20, batch_size=128, verbose=2)
```

2.4 模型测试

使用 evaluate()方法对模型进行测试，并返回测试误差和测试准确率。

```
# 在测试集上测试一下模型的性能
test_loss, test_accuracy = network.evaluate(test_images, test_labels)
print("test_loss:", test_loss, "test_accuracy:", test_accuracy)
```

下面为模型测试的结果

```
Epoch 16/20
469/469 - 6s - loss: 0.0077 - accuracy: 0.9976 - 6s/epoch - 13ms/step
Epoch 17/20
469/469 - 6s - loss: 0.0076 - accuracy: 0.9976 - 6s/epoch - 13ms/step
Epoch 18/20
469/469 - 6s - loss: 0.0070 - accuracy: 0.9976 - 6s/epoch - 13ms/step
Epoch 19/20
469/469 - 6s - loss: 0.0061 - accuracy: 0.9982 - 6s/epoch - 13ms/step
Epoch 20/20
469/469 - 7s - loss: 0.0051 - accuracy: 0.9984 - 7s/epoch - 14ms/step
313/313 [==============================] - 1s 2ms/step - loss: 0.0474 - accuracy: 0.9890
test_loss: 0.04737924039363861 test_accuracy: 0.9890000224113464
```


深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

2.5 模型保存

Keras 支持保存 HDF5 文件，也就是.h5 文件，这个文件包含模型的体系结构，权重值和 compile()信息。

因为 Fibo AI Stack 模型转化工具支持 onnx-dlc、tflite-dlc 与 TensorFlow-dlc,当前不支持直接将.h5 文件转化成 dlc，所以说需要先将.h5 文件转化成 tflite，然后在用 Fibo AI Stack 模型转化工具将 tflie 转化成 dlc。

```
# 模型保存为.h5格式
network.save('MyModel.h5')
# 模型转换为.tflite格式
keras_file = 'MyModel.h5'
tf.keras.models.save_model(network, keras_file)
model = tf.keras.models.load_model(keras_file)
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open('MyModel.tflite', 'wb') as f:
    f.write(tflite_model)
```

在电脑端运行 LeNet.py 后，工程文件夹中会出现 MyModel.h5 和 MyModel.tflite 两个文件。

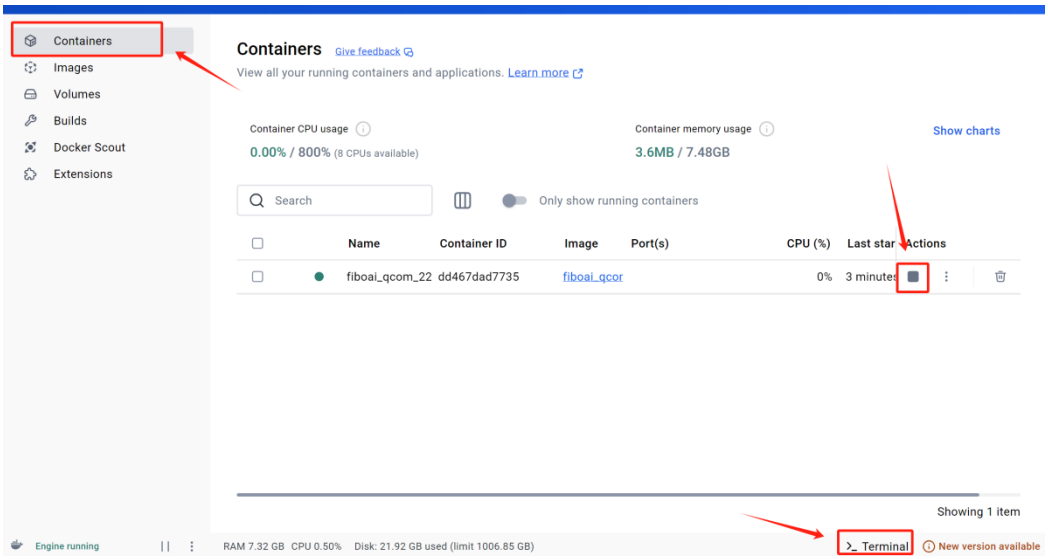
MyModel.h5	2023/10/20 9:49	H5 文件	902 KB
MyModel.tflite	2023/10/20 9:49	TFLITE 文件	434 KB

3 Fibo AI Stack 模型转化

3.1 模型导入

打开 Docker Desktop，点击运行虚拟机并打开终端。（具体环境搭建操作可在 AI 端侧部署开发中的课程查看,课程名称为“Fibo AI Stack 模型转化指南”，这里以 Docker 环境为例）

深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn



在终端输入命令,查看正在运行的虚拟机，如图所示即可：docker ps

输入以下命令，将电脑中的模型文件传输到虚拟机中：
docker cp <电脑中的模型文件路径> <虚拟机名称>:<虚拟机中的文件路径>
如：docker cp D:\Project\MyModel.tflite fiboai_qcom_226_env:/home/project/
如图所示，传输成功

```
PS C:\Users\FIBOCOM> docker cp D:\Project\MyModel.tflite fiboai_qcom_226_env:/home/project/
Successfully copied 445kB to fiboai_qcom_226_env:/home/project/
PS C:\Users\FIBOCOM>
```

3.2 模型转化

输入命令，进入虚拟机终端中：docker exec -it <虚拟机名称> bash
如：docker exec -it fiboai_qcom_226_env bash

```
PS C:\Users\FIBOCOM> docker exec -it fiboai_qcom_226_env bash
/opt/2.26.0.240828
[INFO] AISW SDK environment set
[INFO] QNN_SDK_ROOT: /opt/2.26.0.240828
[INFO] SNPE_ROOT: /opt/2.26.0.240828
(snpe_env) root@64d0060ee0cd:/#
```

使用 Fibo AI SDK 中的“snpe-tflite-to-dlc”工具，将 tflite 格式模型转化为 DLC 格式模型，具体方法：

snpe-tflite-to-dlc –-input_network /路径/model.tflite –-input_dim input_name

深圳市广和通无线股份有限公司 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

“1,299,299,3” –-output_path /路径/model.dlc

如：

snpe-tflite-to-dlc --input_network /home/project/MyModel.tflite --input_dim "serving_default_conv2d_input:0" "1,28,28,1" --output_path /home/project/MyModel.dlc

注释：

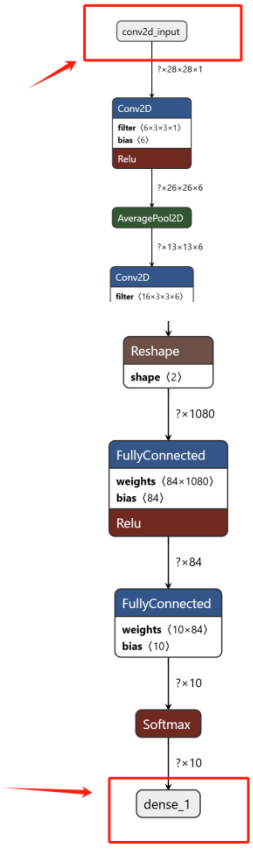
--input_network 参数表示：需要转换的模型框架路径

--output_path 参数表示：转换模型文件输出路径

--input_dim 参数表示：需转化模型的输入名称和输入数据格式，对于输入名称和输入数据格式不清楚的用户，可以在 https://netron.app/中导入自己的模型，即可查看到输入名称和输入数据格式

红色字体部分需自行填入

下面进行示例



如图所示，输入命令，转化成功

```
(snpe_env) root@64d0060ee0cd:/home/project# snpe-tflite-to-dlc   --input_network /home/project/MyModel.tflit
e   --input_dim "serving_default_conv2d_input:0" "1,28,28,1"   --output_path /home/project/MyModel.dlc
2025-04-07 08:16:46,513 - 235 - INFO - INFO_INITIALIZATION_SUCCESS:
2025-04-07 08:16:46,518 - 235 - INFO - INFO_CONVERSION_SUCCESS: Conversion completed successfully
2025-04-07 08:16:46,526 - 235 - INFO - INFO_WRITE_SUCCESS:
(snpe_env) root@64d0060ee0cd:/home/project#
```

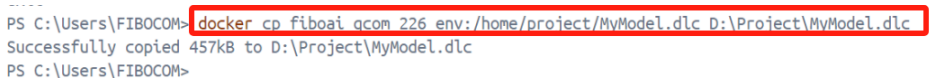

深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

3.3 模型导出

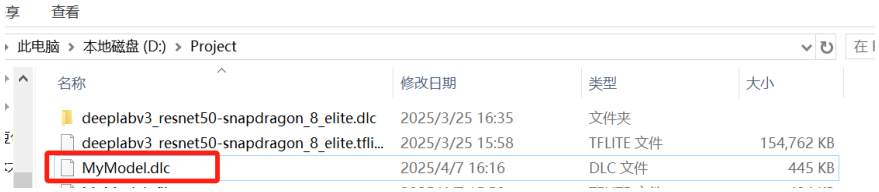
输入命令退出虚拟机终端：exit

输入命令将文件导出虚拟机：docker cp <虚拟机名称>:<虚拟机中的模型路径> <电脑中的模型文件路径>

如 ： docker cp fiboai_qcom_226_env:/home/project/MyModel.dlc D:\Project\MyModel.dlc



如图所示，文件导出成功



4 数字识别模型部署

4.1 准备

1. 将转换后的 dlc 文件以及所需的其他文件放入 SC171 开发套件 V3 开发板中，**注意**：要将本案例附件中的 sample.py 与 api_infer.py 放置在**同一路径**下。

2. 进入开发板的 Ubuntu 的 UI 界面，打开开发板终端，调用脚本配置环境参数：

cd /home/fibo/Fibo_AI_Stack/fiboaisdk_ubuntu_aarch64

. ./scripts/env_qualcomm.sh 68

3. 导入所需的库，在终端输入以下命令：

pip3 install numpy opencv-python

4. 需要准备的参数：

--dlc_path：模型文件路径（.dlc 文件）

--img_path：输入图片路径

--Label_file:标签文件路径

--模型相关数据:输入张量名称、输出张量名称

4.2 Python sample 代码

以下显示黑色背景的代码均为在开发板端上的 sample.py

深圳市广和通无线股份有限公司 总机: +86 755-26733555 传真: +86 755-26520841
邮箱: market@fibocom.com 官网: www.fibocom.com.cn

1. 调用库，定义文件路径
2. 读取标签文件函数

```python
import cv2
import numpy as np
from api_infer import *

dlc_path = "/home/fibo/Fibo_AI_Stack/number/MyModel.dlc"
img_path = "/home/fibo/Fibo_AI_Stack/number/image1.jpg"
output_dir = "/home/fibo/Fibo_AI_Stack/number/"
label_file = "/home/fibo/Fibo_AI_Stack/number/label.txt"

# 读取标签文件，返回列表
def read_label_list():
    with open(label_file, 'r', encoding="utf8") as f:
        data = f.read().splitlines()
    return data
```

调用库

定义文件路径

3. 更换推理设备（如 Runtime.CPU 或 Runtime.GPU）。
4. 根据使用的模型需求来更改对输入内容的前处理和后处理的代码部分。
5. 根据使用模型，修改输入输出张量名

```python
def run_snpe():
    # 创建 SNPE 运行时环境
    snpe_ort = SnpeContext(dlc_path, [], Runtime.GPU, PerfProfile.BALANCED, LogLevel.INFO)
    assert snpe_ort.Initialize() == 0  # 初始化 SNPE 环境

    # 图像预处理
    img1 = cv2.imread(img_path)  # 读取图片
    img = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)  # 转换为灰度图
    img_inverted = 255 - img  # 反色处理
    _, bit_img = cv2.threshold(img_inverted, 127, 255, cv2.THRESH_BINARY)  # 二值化处理
    bit_img_resized = cv2.resize(bit_img, (28, 28), interpolation=cv2.INTER_AREA)  # 调整大小为 28x28
    _, bit_img_final = cv2.threshold(bit_img_resized, 20, 255, cv2.THRESH_BINARY)  # 再次二值化处理
    # 保存预处理后的图片
    cv2.imwrite(f"{output_dir}/preprocessed_image.jpg", bit_img_final)

    # 准备模型输入数据
    input_feed = {"serving_default_conv2d_input:0": bit_img_final}

    # 执行模型推理
    outputs = snpe_ort.Execute(["StatefulPartitionedCall:0"], input_feed)
    print("模型输出结果:", outputs)

    # 后处理部分
    # 打印输出结果
    if outputs is not None:
        for k, v in outputs.items():
            print(f"输出名称: {k}, 输出数据: {v}")
            if k == "StatefulPartitionedCall:0":  # 根据输出名称匹配
                output_data = v  # 获取输出数据
                w = np.argmax(output_data)  # 找到最大值的位置
                label_list = read_label_list()  # 读取标签列表
                print("预测结果:", label_list[w])  # 打印预测标签

    # 释放资源
    assert snpe_ort.Release() == 0

if __name__ == "__main__":
    run_snpe()
```

推理设备

输入内容前处理部分

模型输入张量名

前处理结果

模型输出张量名

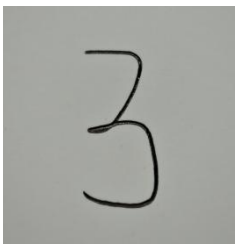
输出内容后处理部分

深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com | 传真: +86 755-26520841 官网: www.fibocom.com.cn

4.3 模型的运行推理

调用 sample 执行文件进行推理，在开发板终端输入以下命令：
python3 <Python sample 文件地址>
如：
python3 /home/fibo/Fibo_AI_Stack/fiboaisdk_ubuntu_aarch64/sample/sample.py

输入内容，如图所示



输出结果，执行成功，结果如图显示：3

```
[2025-03-19 04:39:03.490] [multi-sink] [---I---] [thread 21433]:  [registry] [snpe] Set cpu_float32 runtime s
[2025-03-19 04:39:03.995] [multi-sink] [---I---] [thread 21433]:  [registry] [snpe] Set gpu_float32_16_hybrid
cceeded
[2025-03-19 04:39:03.995] [multi-sink] [---I---] [thread 21433]:  [registry] [snpe] Set cpu_float32 runtime s
[2025-03-19 04:39:03.995] [multi-sink] [---I---] [thread 21433]:  InferAPI init success
[2025-03-19 04:39:04.023] [multi-sink] [---I---] [thread 21433]:  [registry] [snpe] [multiple] SnpeInferPriv:
ecute time: 2957 us
[2025-03-19 04:39:04.023] [multi-sink] [---I---] [thread 21433]:  [registry] [snpe] [multiple] SnpeInferPriv:
ole time: 3037 us
模型输出结果: {'StatefulPartitionedCall:0': [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
输出名称: StatefulPartitionedCall:0, 输出数据: [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
预测结果: 3
fibo@qcs6490-odk:~/Fibo_AI_Stack/fiboaisdk_ubuntu_aarch64$
```

5 Q&A

5.1 无法找到文件

可能原因：
1. 模型、标签等文件路径配置错误。
2. 模型文件未正确下载或解压。
3. 无权限访问文件。

解决办法：
1. 修改模型的存放路径。
2. 确保模型文件已正确下载并解压到指定路径。
3. 确保运行的 Python 文件路径正确。
4. 检查文件权限，确保当前用户有权限访问模型文件。

深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn

5.2 环境配置失败

可能原因：

1. 环境变量未正确设置。
2. 缺少必要的 Python 包。

解决办法：

1. 重新执行环境配置脚本。
2.检查环境变量是否已正确设置，确保路径中没有错误。
3.安装必要的 Python 包。

5.3 推理结果不准确

可能原因：

1. 输入图像分辨率不符合模型要求。
2. 模型未正确加载或配置。

解决办法：

1. 确保输入图像格式正确使用，可以输出输入数据来检查格式是否符合模型要求：print(input_feed)
2. 检查模型，输入输出 Tensor 名称等信息正确。
3. 重新加载模型并运行推理，确保模型配置无误。
4. 检查输入图像的预处理步骤，确保图像数据格式符合模型要求。

深圳市广和通无线股份有限公司 | 总机: +86 755-26733555 邮箱: market@fibocom.com 传真: +86 755-26520841 官网: www.fibocom.com.cn