微控制器平台低代码神经网络推理技术研究

张岩，宋岩

（恩智浦（北京）有限公司，北京 100022）

摘要：基于微控制器平台的深度学习技术（TinyML）的兴起，意味着在广大嵌入式平台上部署人工智能的重大革新必将带来 AIoT 的发展。微控制器种类繁多，不同的芯片，其开发方式和基础 API 也各不相同，与 PC 端相比，模型的推理技术开发工作难度不小，需要具备一定的嵌入式开发经验。本文尝试将高级语言 Python 与微控制器结合，为入门开发者提供一套简单、可靠的开源低代码开发套件（名为“OpenART”），可以大大提高在微控制器上部署深度学习的开发效率、降低学习成本。

关键词：OpenART；TinyML；i. MX RT；低代码推理技术

中图分类号：TP872　　**文献标识码**：A

Research on Low-code Neural Network Inference Technology of Microcontroller Platform

Zhang Yan, Song Yan

（NXP Semiconductor (Beijing) Co., Ltd., Beijing 100022, China）

Abstract: The rise of deep learning technology on microcontrollers (TinyML) marks a major innovation in the deployment of artificial intelligence on the vast number of embedded platforms, which will greatly boost the development of AIoT. There are many kinds of microcontrollers, and different chip development methods and basic APIs are also different. Compared with the PC side, the development of the reasoning technology of the model is not easy, and certain embedded development experience is required. We try to combine high-level language Python with microcontrollers to provide beginner developers with a simple and reliable low-code development kit called "OpenART", which can greatly improve the development efficiency and reduce learning efforts of deploying, evaluating, and benchmarking deep learning on microcontrollers.

Keywords: OpenART; TinyML; i. MX RT; low-code inference technology

0 引　言

微控制器广泛应用于多种电子设备，早期实现以控制和通信为主的功能，但现在微控制器的计算能力和存储容量都有了长足的进步，例如已经有高达 1 GHz 主频的基于 Cortex－M7 内核的微控制器 i. MX RT1170，可以支持 2 MB 片上内存，外部扩展数十 MB 的 SDRAM 和 Flash 空间，使得微控制器用于完成的计算密集型任务日益增多，包括近十年非常受欢迎的深度学习。不过，当前流行的深度学习开发工具和模型架构设计主要还是以传统的计算机和移动平台为主，它们能运行全功能的操作系统，算力强大，开发工具比较统一。然而，在微控制器上，完全不同的基础软件和开发方式使得现有的生态无法复用。虽然 Google tflite－micro 和 GLOW 为微控制器推理运行提供了基础支持，但是微控制器上的开发比 PC 和移动设备更加原始和晦涩（如几乎只能使用 C/C＋＋），不同微控制器的硬件和基础 API 区别也很大，工程人员难以在微控制器平台上部署和测试深度学习技术。为此，本文结合 Micropython 项目，为 tflite－micro 和 GLOW 提供了 Python 绑定，并且围绕推理引擎，为常见的深度学习应用封装了方便易用的 API，尽量呈现出与在 PC/服务器上开发深度学习一致的体验。

1 简　介

如果完全站在微控制器的角度，似乎它们薄弱的配置更适合使用传统的机器学习技术——事实上也的确有不少基于微控制器的嵌入式系统已经在部署，软件实现方式各不相同。那为何还要部署深度学习技术呢？其实答案已经不言而喻：一方面，微控制器的发展有长足的进步，性能提升极大推动了 AIoT 的发展，试问哪个嵌入式开发工程师能够拒绝万物“智能互联”的诱惑呢；另一方面，深度学习天然有处理复杂问题的优势以及灵活的伸缩性。加之深度学习模型的小型化研究与实践不断取得突破，使得深度学习模型越来越易于部署在微控制器平台上。

在 PC 上，我们可以轻松实现深度学习的全套工作流程，包括数据集的采集与加工，模型的设计、训练与验证，模型的优化，模型的部署和与应用的集成以及完整应用的开发与发行。在微控制器上，深度学习应用也要走完相同的过程。不过，微控制器的资源和软件生态都难以支撑除了运行最终成果之外的步骤，因此，如同嵌入式软件开发的“交叉编译”，深度学习应用的生成过程也需要借助 PC 上的开发工具。在 MCU 上集成 AI 的整体流程如图 1 所示。

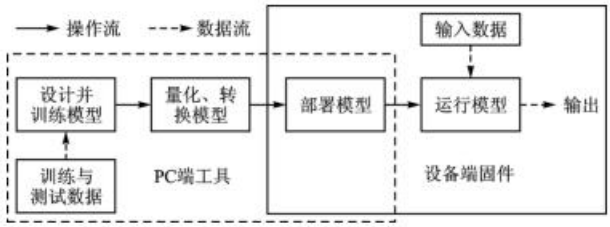


图 1　在 MCU 上集成 AI 的整体流程

PC 端主流的模型训练框架（如 TensorFlow 或 Pytorch）都是基于 Python 语言实现的，凭借丰富的运算库，Python 可以很容易地实现数据流的操作（数据采集，数据分割、处理以及模型训练量化）。作为脚本语言的范例，Python 语言执行调试便利，语法简单、容易上手，对非计算机专业的人士非常友好，大大降低了深度学习开发门槛。非计算机专业的初学者可以根据书籍或开源资料很轻松地在 PC 端完成模型设计、训练到部署全套工作流程。

设备端的部署工作则完全不同，微控制器平台大多基于 C、C++语言实现，这就需要开发者必须具有嵌入式 C 开发基础，而且不同厂家不同芯片的硬件资源和驱动程序接口不同，需要开发者对微控制器平台与工具非常了解，这无疑更是提高了入门门槛。

以一个常见应用“欢迎光临”感应器为例。这种感应器多采用红外线感应，若识别到有物体经过，则会发出“欢迎光临”语音。这种感应器功能单一且不能过滤非人物体。借助于神经网络深度学习给这个感应器加上智能识别功能：人体检测加计数统计，既可以识别客人进入门店也可以统计客流量，这就是 AIoT 的一个典型应用。具体工作流程如图 2 所示。



图 2　“欢迎光临”感应器工作流程

开发难点包括：

① 摄像头驱动开发。需要熟悉摄像头采集模块（如 CSI）驱动开发、摄像头外设驱动开发。

② 图像数据流处理。使用环形队列管理确保图像实时性。

③ 模型推理引擎移植。根据训练框架移植推理引擎（比如 TensorFlow Lite Micro）到微控制器平台。

④ 图像数据预处理。根据模型输入要求对图像进行缩放、格式转换及归一化处理等。

⑤ IoT 模块。熟悉 IoT 模块驱动，把模型输出结果通过云端推送至手机。

对经验丰富的开发者来说，以上开发都是不小的工作量，对于入门初学者更是望而却步了。MicroPython 的出现弥补了微控制器平台高级语言的缺憾。MicroPython 是 Python3 的精简版本，包含了 Python 标准库的一个子集并优化为可以在微控制器和受限环境中运行。通过 MicroPython 提供的库和函数，开发者可以快速控制硬件外设，实现了底层硬件驱动的封装。这样不仅降低了开发难度，而且减少了重复开发工作，从而加快开发速度、提高工作效率。

恩智浦推出了基于 MicroPython 的低代码 AI 开发套件 Open AI on RT（RT 是恩智浦微控制器 i. MX RT 产品系列），简称 OpenART，主要实现 AI 教学、模型验证。其支持基于传感器时间序列数据的机器学习（异常检测、姿态识别、智能控制策略），基于语音的音源定位、语音触发，基于图像数据的机器视觉以及基于载具云台的智能控制等；集成了 OpenMV 机器视觉库，可以方便操作摄像头并将图像数据抽象数据对象，提供丰富的视觉运算和神经网络推理运算及针对常见任务类型（如分类、检测、个体识别等）的后处理功能。开发者可以仅用几行代码轻松实现图像抓取、模型推理运算。

2　实　现

OpenART 的硬件平台采用高性能跨界处理器 i. MX. RT 1060 系列，集成了摄像头、麦克风等数据采集外设，并支持 LCD 显示设备。软件系统基于 RT－Thread 实时操作系统，集成了 MicroPython 解释器。

OpenART 的软件架构如图 3 所示。由图 3 可知，通过 Python API bindings 实现了 Python 对 TFLM 推理库、LVGL 图形库、OpenMV 机器视觉库的封装。MicroPython Runtime 的优化极大降低了 Python 语句解析到 C 库映射的消耗，从而提升了运行效率。得益于 Python 解释执行的特性，软件开发工作脱离了嵌入式原有开发环境和开发流程，既不需要交叉编译工具链，也不需要下载程序，只需要将编写好的脚本在 Python 执行环境下运行就可得到想要的结果。脚本或单行的 Python 命令既可以在交互式命令界面中下达并执行，也可以存储在文件系统中，由 OpenART 在复位后自动加载执行 main. py。OpenART

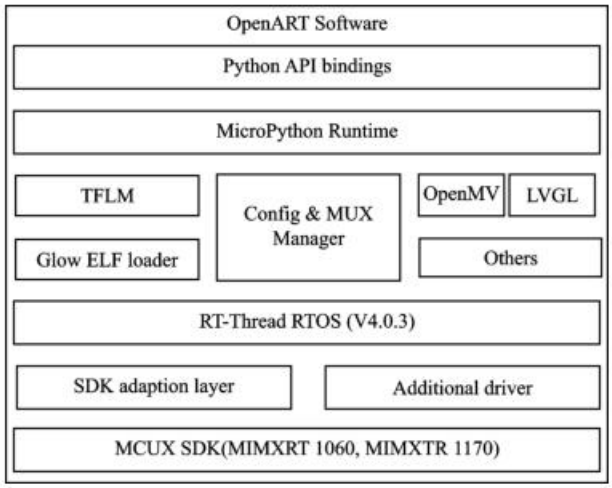


图 3　OpenART 的软件架构

支持把 TF 卡通过 USB 模拟成 U 盘方式，使用户可以方便地在电脑与 OpenART 硬件之间读写文件。

图 3 中负责模型推理的主要是 TFLM 模块，它使用 tflite－micro 作为模型推理引擎。我们在集成时做了两个主要的改进：一是优化了片内快速内存的访问效率；二是根据应用的类型对 tflite－micro 返回的结果做了进一步的封装，根据不同任务需求，在 TFLM 的基础上封装了不同的 API 接口（主要用于数据回归、图像分类、物体检测、模型推理），并且添加了性能分析接口。

（1）数据回归

Regress(net,data,data_size,scale,offset)

- net：tflite 文件路径或模型对象。
- data：输入数据。
- data_size：输入数据长度。
- offset：输入数据移位参数。
- scale：输入数据缩放参数。

（2）图像分类

Classify(net,image,offset,scale,bgr)

- net：tflite 文件路径或模型对象。
- image：通过摄像头抓取的图像对象。
- offset：输入数据移位参数。
- scale：输入数据缩放参数。
- bgr：模型输入数据的颜色空间，bgr＝1，表示颜色数据是 blue、green、red。

参数 offset、scale 用于输入数据归一化处理运算公式：(image－offset)/scale。与模型训练时归一化操作一致，例如归一化范围为－1～1 时，offset＝128，scale＝128。

函数返回类别概率列表。

（3）物体检测

Detect(net,image,offset,scale,bgr)

- net：tflite 文件路径或模型对象。
- image：通过摄像头抓取的图像对象。
- offset：输入数据移位参数。
- scale：输入数据缩放参数。
- bgr：模型输入数据的颜色空间，bgr＝1，表示颜色数据是 blue、green、red。

函数返回物体位置信息、分类信息、置信度及物体数量。

（4）模型推理

Invoke(net,image,offset,scale,bgr)

- net：tflite 文件路径或模型对象。
- image：通过摄像头抓取的图像对象。
- offset：输入数据移位参数。
- scale：输入数据缩放参数。
- bgr：模型输入数据的颜色空间，bgr＝1，表示颜色数据是 blue、green、red。

函数直接返回模型的输出。

（5）性能分析

Profile(net,image)

- net：tflite 文件路径或模型对象。

该接口函数会在串口端输出模型的推理时间以及每个算子运行时间，开发者可以根据统计结果有针对性地优化模型结构。

在 OpenART 下，开发者可以使用 Python 脚本采集视频数据，从文件系统加载和执行模型文件，并在 LCD 上实时显示运算结果。系统运行流程如图 4 所示。

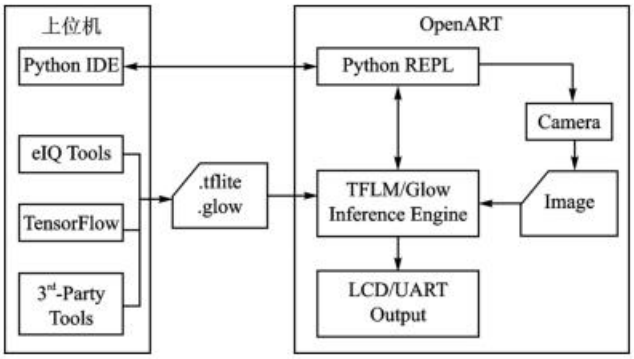


图 4　系统运行流程图

其中，上位机的 Python IDE 既可以直接使用串口终端来操作，也可以使用进一步的封装（如 RT－Thread 为 Visual Studio Code 开发的 Micropython 插件）。此外，还可以借助 OpenART 集成的 OpenMV 模块利用 OpenMV IDE 来开发和测试。OpenART 与上位机之间通过 USB 通信。

OpenART 支持两种格式的模型：一种是 tflite－micro 所使用的 tflite 格式；另一种是神经网络编译器 GLOW 产生的目标码链接成可执行 ELF 文件格式（.glow）。我们

对 tflite 格式的神经网络做的优化更为充分，建议首选使用。tflite 文件可以通过 TensorFlow 自带的 API 生成，也可以通过 NXP 的免费开发工具 eIQ Toolkit 中附带的 eIQ Converter 工具来生成。tflite 格式支持量化后的模型，推荐使用int8来量化模型以获得更小的文件尺寸和更高的推理性能。

3 应用示例介绍

以上文提到的“欢迎光临”系统为例。首先，在 PC 端训练一个精简版的 SSD 模型，如图 5 所示。

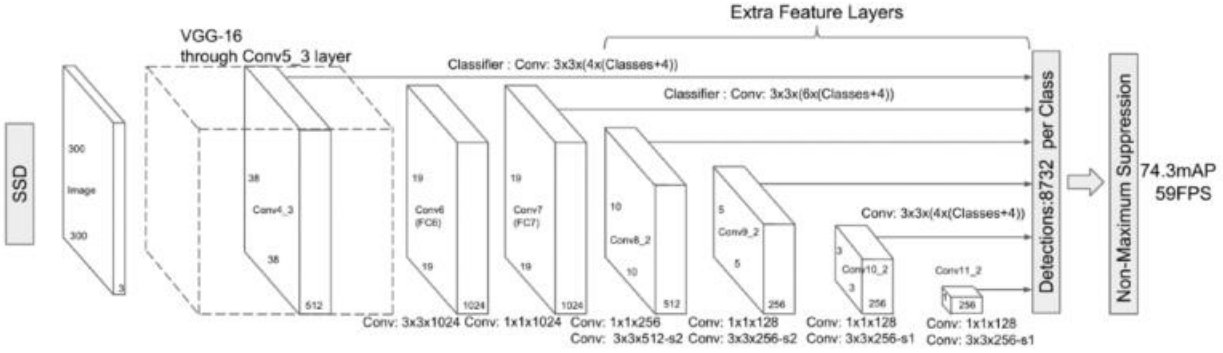


图 5 SSD 模型

模型输入大小是 192×256×3，模型输出检测框位置信息、分类信息、置信度信息和目标数量信息，使用 TensorFlow 训练并量化成 int8 类型。模型推理代码如下：

```
import sensor,image,time,tf
sensor.open()
sensor.reset()
sensor.set_pixformat(sensor.RGB565)    #设置图像格式
sensor.set_framesize(sensor.QVGA)    #设置图像分辨率
net=tf.load('/sd/person_detect.tflite')    #加载模型
total_count=0
while(True):
    img=sensor.snapshot()                #获取摄像头图像
    objs=tf.detect(net,img,bgr=1,offset=128,scale=128)
    for obj in objs:
        x1,y1,x2,y2,lab,scores=obj
        if scores>0.4:
            #检测到行人经过
            total_count=total_count+1    #人流量统计
            upload(total_count)      #上报流量信息
```

OpenMV IDE 中运行效果如图 6 所示。



图 6 OpenMV IDE 中运行效果

4 测试结果及分析

上文示例通过简单几行代码实现了完整的应用逻辑，推理速度与 C 语言实现相比，结果如表 1 所列。

由以上可知，基于微控制器平台的低代码推理技术性能损失几乎可以忽略不计。当然，为了能在微控制器上快速运行这个模型做了很多架构上的精简。如果需要尝试不同的模型，只需要把对应的模型文件存储在文件系统中，修改脚本替换为新文件的名称，再复位硬件（在 OpenMV IDE 中调试时可以直接重新运行）即可。

表 1 推理速度对比

C 实现推理时间/ms	130
Python 脚本推理时间/ms	132

本文介绍的 OpenART 项目源码开源且遵循 Apache 2.0 协议，代码仓库地址为 https://github.com/NXPmicro/OpenART。

参考文献

[1] David R, Duke J, Jain A, et al. TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems[J]. 2020.
[2] https://micropython.org.
[3] https://docs.openmv.io.
[4] https://github.com/tensorflow/tflite-micro.

张岩、宋岩（系统工程师），主要研究方向为嵌入式平台的 AI 应用及开发。通信作者：张岩，tony.zhang@nxp.com。

（责任编辑：薛士然 收稿日期：2022-08-08）