如何在 OpenMV 生态系统中集成 STM32Cube.AI 生成的代码

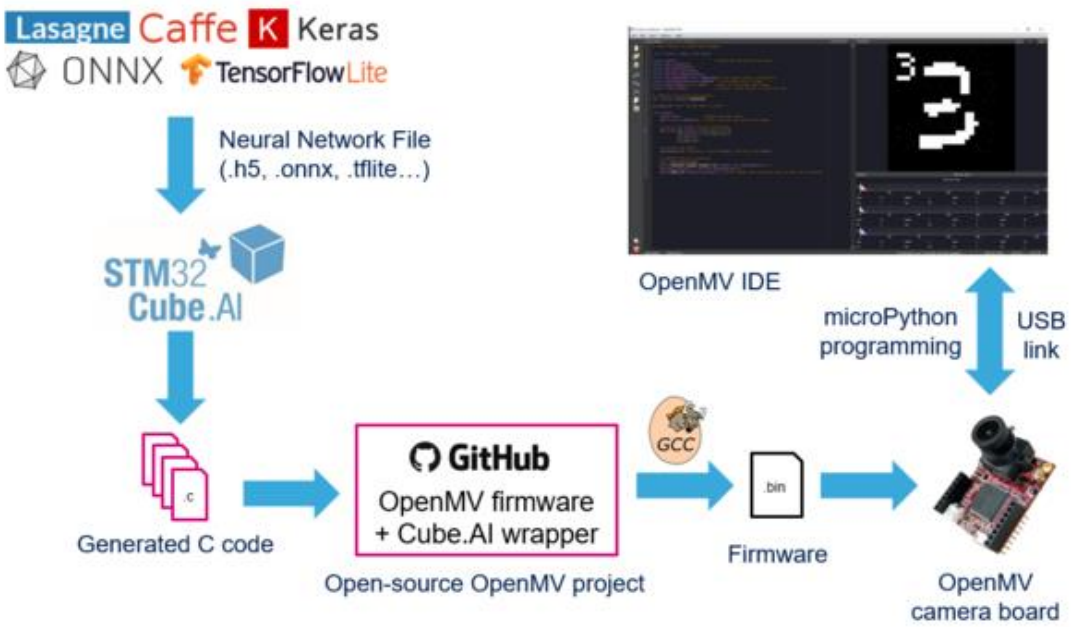
目录↑

1 STM32Cube.AI 启用状态的下 OpenMV 固件

本教程将引导您将自己的神经网络集成到 OpenMV 环境中。

OpenMV 开源项目提供的源代码，使您可在 STM32Cube.AI 启用的状态下编译 OpenMV H7 固件的源代码。

下图描述了 STM32Cube.AI 和 OpenMV 结合使用的过程。



1. 使用您喜爱的深度学习框架来训练您的神经网络。
2. 使用 STM32Cube.AI 工具将训练好的神经网络转换为优化的 C 语言。
3. 下载 OpenMV 固件源代码
4. 将生成的文件添加到固件源代码
5. 使用 GCC 工具链编译
6. 使用 OpenMV IDE 烧录
7. 使用 microPython 对板进行编程并执行推理

ⓘ	许可证信息： X-CUBE-AI 是根据 Mix Ultimate Liberty + OSS + 3rd-party V1 软件许可协议 SLA0048 交付的

1.1 先决条件↥

本文假定您使用 Linux 环境（已在 Ubuntu 18.04 上进行了测试）。

ⓘ	Windows 用户，对于 Windows 用户，强烈建议您安装 Windows Subsystem for Linux (WSL) Ubuntu 18.04，该子系统可提供 Ubuntu Linux 环境。 系统安装完成后，您可从 Windows 文件资源管理器在以下位置访问 WSL Ubuntu 文件系统。

	C:\Users\<username>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc\LocalState\rootfs

所有以此语法开头的指令都需在 Linux 控制台中执行：

```
PC $> <mycommand>
```

1.2 要求

1.2.1 确保您的环境更新至最新版

```
PC $> sudo apt update

PC $> sudo apt upgrade

PC $> sudo apt install git zip make build-essential tree
```

1.2.2 创建您的工作区目录

```
PC $> mkdir $HOME/openmv_workspace
```

	此处仅为目录组织的建议。 以下所有指令行将引用此目录。

1.2.3 安装 stm32ai 指令行以生成优化代码↑

- 从 ST 网站中将最新版的 X-CUBE-AI 扩展包下载到您的 OpenMV 工作区目录。

Windows 用户可在以下地址复制下载的压缩文件

```
C:\Users\<username>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc\LocalState\rootfs\home\<username>\openmv_workspace
```

随后您需要关闭并重启 WSL Ubuntu 控制台。

- 提取存档

```
PC $> cd $HOME/openmv_workspace

PC $> chmod 644 en.x-cube-ai-v5-0-0.zip

PC $> unzip en.x-cube-ai-v5-0-0.zip

PC $> mv STMicroelectronics.X-CUBE-AI.5.0.0.pack STMicroelectronics.X-CUBE-AI.5.0.0.zip

PC $> unzip STMicroelectronics.X-CUBE-AI.5.0.0.zip -d X-CUBE-AI.5.0.0
```

- 将 **stm32ai** 指令行添加到您的 PATH。

```
PC $> export PATH=$HOME/openmv_workspace/X-CUBE-AI.5.0.0/Utilities/linux:$PATH
```

您可以确认指令行是否正确安装：

```
PC $> stm32ai --version

stm32ai - Neural Network Tools for STM32 v1.2.0 (AI tools v5.0.0)
```

1.2.4 安装 7-2018-q3 版本的 GNU Arm 工具链来编译固件

```
PC $> sudo apt remove gcc-arm-none-eabi

PC $> sudo apt autoremove

PC $> sudo -E add-apt-repository ppa:team-gcc-arm-embedded/ppa

PC $> sudo apt update

PC $> sudo -E apt install gcc-arm-embedded
```

您可确认 GNU Arm 工具链是否正确安装：

```
PC $> arm-none-eabi-gcc --version

arm-none-eabi-gcc (GNU Tools for Arm Embedded Processors 7-2018-q3-
update) 7.3.1 20180622 (release) [ARM/embedded-7-branch revision 2
61907]

Copyright (C) 2017 Free Software Foundation, Inc.

This is free software; see the source for copying conditions.  Ther
e is NO
```

```
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

1.2.5 安装 OpenMV IDE ↥

从 OpenMV 网站中下载 OpenMV IDE
IDE 用于开发 microPython 脚本和烧录固件。

1.3 步骤 1-下载并准备 OpenMv 项目↥

在这一步骤中，我们将克隆 OpenMV 项目，检测一个已知的工作版本并创建一个分支。

然后初始化 git 子模块。这将克隆 OpenMV 依赖项，例如 microPython。

请确保在 oopenmv 目录中的路径中没有空格，否则将编译失败。您可通过在 **openmv** 目录中运行 **pwd** 指令来检查。若存在空格，请将此目录移动至没有空格的路径。

1.3.1 克隆 OpenMV 项目↥

```
PC $> cd $HOME/openmv_workspace

PC $> git clone https://github.com/openmv/openmv.git
```

1.3.2 检测一个已知的工作版本↥

```
PC $> cd openmv

PC $> git checkout b4bad33 -b cubeai
```

1.3.3 下载 micropython 子模块和必需的依赖项↥

- 下载 micropython 子模块

```
PC $> git submodule update --init
```

- 仅下载必需的依赖项(lib/berkeley-db-1.xx and lib/stm32lib)

```
PC $> cd src/micropython

PC $> git submodule update --init lib/berkeley-db-1.xx lib/stm32lib
```

1.4 步骤 2-将 STM32Cube.AI 库添加到 OpenMV ↥

OpenMV 固件下载完成后，您需将 STM32Cube.AI 的运行时库和头文件复制到 OpenMV 项目中。

```
PC $> cd $HOME/openmv_workspace/openmv/src/stm32cubeai

PC $> mkdir -p AI/{Inc,Lib}

PC $> mkdir data
```

从 STM32Cube.AI 中将文件复制到 AI 目录中：

```
PC $> cp $HOME/openmv_workspace/X-CUBE-AI.5.0.0/Middlewares/ST/AI/
Inc/* AI/Inc/
```

```
PC $> cp $HOME/openmv_workspace/X-CUBE-AI.5.0.0/Middlewares/ST/AI/Lib/ABI2.1/STM32H7/NetworkRuntime*_CM7_IAR.a AI/Lib/NetworkRuntime_CM7_GCC.a
```

操作完成后，AI 目录应如下所示

```
AI/
├── Inc
│   ├── ai_common_config.h
│   ├── ai_datatypes_defines.h
│   ├── ai_datatypes_format.h
│   ├── ai_datatypes_internal.h
│   ├── ai_log.h
│   ├── ai_math_helpers.h
│   ├── ai_network_inspector.h
│   ├── ai_platform.h
│   ├── ...
├── Lib
│   └── NetworkRuntime_CM7_GCC.a
└── LICENSE
```

1.5 步骤 3-生成神经网络模型所需的代码↥

在此步骤中，您将训练卷积神经网络来识别手写数字。然后借助 STM32Cube.AI，为此网络生成 STM32 优化的 C 代码。这些文件将被添加到 OpenMV 固件源代码中。然后将生成一个优化的 C 代码

1.5.1 训练卷积神经网络↥

或者，您可以跳过此步骤，使用已完成训练的 **mnist_cnn.h5** 文件（见下章）。

来自 Keras 的用于数字分类的卷积神经网络（MNIST）将用作示例。若您想训练网络，则需安装 Keras。

运行以下指令，来训练网络并将模型保存到磁盘：

```
PC $> cd $HOME/openmv_workspace/openmv/src/stm32cubeai/example

PC $> python3 mnist_cnn.py
```

1.5.2 STM32 优化代码的生成↥

生成 STM32 优化代码，请按照以下要求使用 stm32ai 指令行工具：

```
PC $> cd $HOME/openmv_workspace/openmv/src/stm32cubeai

PC $> stm32ai generate -m example/mnist_cnn.h5 -o data/
```

以下文件生成于**$HOME/openmv_workspace/openmv/src/stm32cubeai/data：**

```
* network.h
```

```
* network.c

* network_data.h

* network_data.c
```

1.5.3 预处理↥

若您需要在运行推断之前进行一些特殊预处理，则必须修改 `src/stm32cubeai/nn_st.c` 中的 `ai_transform_input` 函数。默认情况下，该代码执行以下操作：

- 简单的大小调整（二次采样）
- 从无符号字符转化为浮点数
- 将像素从[0,255]缩放到[0, 1.0]

提供的示例对您的应用程序而言可能是开箱即用的，但您可能需要查看一下这一函数。

1.6 步骤 4-编译↥

- 编辑 **omv/boards/OPENMV4/omv_boardconfig.h** 76 行并将 **OMV_HEAP_SIZE** 设置为 **230K**。这会降低 RAM 中的堆部分，从而为神经网络激活缓冲区留出了更多空间。

- 执行以下指令行进行编译：

```
PC $> cd $HOME/openmv_workspace/openmv/src/

PC $> make clean

PC $> make CUBEAI=1
```

这可能需要一段时间，您可通过在 make 指令中添加-j4 或更多（取决于您的 CPU）来加快处理速度。不过这时去喝杯咖啡不失为上策。

	若编译失败并显示.heap 部分使 RAM1 溢出，则您可编辑文件 **src/omv/boards/OPENMV4/omv_boardconfig.h** 并将 **OMV_HEAP_SIZE** 降低几千字节，然后尝试再次构建。 请记得在构建间隔时**执行清理 make clean**。

1.7 步骤 5-烧录固件

- 使用 micro-USB 线将 OpenMV 摄像头插入电脑。
- 打开 OpenMV IDE
- 在工具栏中选择**工具 >运行引导程序**。
- 选择固件文件（位于 **openmv/src/build/bin/firmware.bin**）并按照指示操作。

	Windows 用户的固件在以下位置： `C:\Users\<username>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc\LocalState\rootfs\home\<username>\openmv_workspace\openmv\src\build\bin\firmware.bin`

- 完成后，您可单击位于 IDE 窗口左下方的**连接按钮**。

1.8 步骤 6-使用 micropython 编程

- 打开 OpenMV IDE，并单击位于 IDE 窗口左下方的**连接按钮**
- 创建一个新的 microPython 文件**文件>新文件**
- 您可以从此示例脚本开始，该脚本运行业已嵌入固件的 MNIST 神经网络

```
'''

Copyright (c) 2019 STMicroelectronics

This work is licensed under the MIT license
```

```
'''



# STM32Cube.AI on OpenMV MNIST Example



import sensor, image, time, nn_st



sensor.reset()                      # Reset and initialize the sensor.

sensor.set_contrast(3)

sensor.set_brightness(0)

sensor.set_auto_gain(True)

sensor.set_auto_exposure(True)

sensor.set_pixformat(sensor.GRAYSCALE) # Set pixel format to Grays
cale

sensor.set_framesize(sensor.QQQVGA)   # Set frame size to 80x60

sensor.skip_frames(time = 2000)     # Wait for settings take effec
t.

clock = time.clock()                # Create a clock object to track t
he FPS.



# [STM32Cube.AI] Initialize the network
```

```
net = nn_st.loadnnst('network')



nn_input_sz = 28 # The NN input is 28x28



while(True):

    clock.tick()            # Update the FPS clock.

    img = sensor.snapshot()  # Take a picture and return the image.



    # Crop in the middle (avoids vignetting)

    img.crop((img.width()//2-nn_input_sz//2,

              img.height()//2-nn_input_sz//2,

              nn_input_sz,

              nn_input_sz))



    # Binarize the image

    img.midpoint(2, bias=0.5, threshold=True, offset=5, invert=Tru
e)
```

```
    # [STM32Cube.AI] Run the inference

    out = net.predict(img)

    print('Network argmax output: {}'.format( out.index(max(out))
 ))

    img.draw_string(0, 0, str(out.index(max(out))))

    print('FPS {}'.format(clock.fps())) # Note: OpenMV Cam runs abo
ut half as fast when connected
```

取一张白纸，用黑笔画数字，将相机对准纸。该代码必须产生以下输出：

摄像机的输出

2 MicroPython STM32Cube.AI 包装器的文档

本节提供有关 OpenMV microPython 框架添加的 2 个 microPhyton 函数的信息，以便实现初始化和运行 STM32Cube.AI 优化的神经网络推断。

2.1 loadnnst

```
nn_st.loadnnst(network_name)
```

初始化名为 **network_name** 的网络。

参数：

- **network_name** ：字符串，通常为**'network'**

返回值：

- 网络对象，用于预测

示例：

```
import nn_st

net = nn_set.loadnnst('network')
```

2.2 预测

```
out = net.predict(img)
```

使用 **img** 作为输入运行网络预测。

参数：

- **img** ：图像对象，来自 nn_st 图像模块，通常取自 **sensor.snapshot()**

返回值：

- 网络预测作为 python 列表

示例：