AN13562

为基于 MCU 的智能传感设备构建深度学习模型并进行基准测试

第 1 版—2022 年 4 月 25 日 应用笔记

目录

1 概述

本应用笔记介绍了为智能传感设备构建和部署深度学习模型的过程。它还重点讲述了如何通过在“嵌入式传感设备”上运行不同的推理引擎来验证和评估一个模型的性能。我们所说的“嵌入式传感设备”是指基于 MCU 的设备，能够通过传感器测量多种环境参数，如加速度、振动幅度、方向、温度、压力、声音和电流。

本文可作为构建智能感知应用的指南，它们使用深度学习技术来分析传感器收集的数据，并且对 MCU 上的多种事件进行分类和检测。利用本文件可以构建广泛的用例，例如，预防性维护、状态检测、状态监测、活动识别、手势识别和声音事件检测。

本文描述了一个实际的使用场景：在宿主机上构建和训练卷积神经网络（CNN），然后在基于 MCU 的设备上进行部署和进行基准测试。

这些开发和部署主要是利用恩智浦的 SDK 和 eIQ 技术。

有关更多细节和资源，请参见基于 ML 的系统状态监控应用软件包页面。

2 eIQ 和运行时推理引擎的介绍

恩智浦 eIQ 机器学习（ML）软件开发环境使 ML 算法能够部署在恩智浦 EdgeVerse 微控制器和微处理器（包括 i.MX RT 跨界 MCU 和 i.MX 系列应用处理器）上。eIQ ML 软件包括一个名为 eIQ Toolkit 的 ML 工作流程工具，以及运行时推理引擎、神经网络编译器和经过优化的库，既包含开源技术也包含专有技术。eIQ 设备端软件完全集成到我们的 MCUXpresso SDK 和 Yocto 开发环境中，使开发完整的系统级应用变得容易。

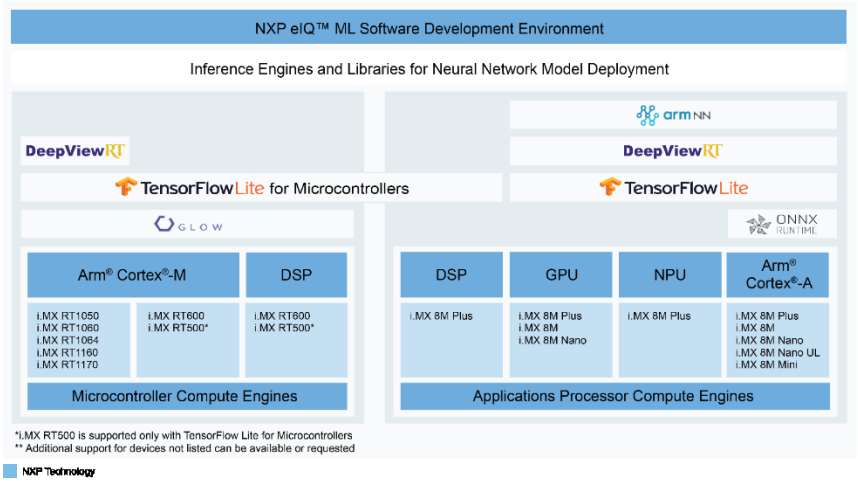

图 1. eIQ 软件生态系统

运行时推理引擎包含在用于微控制器的 MCUXpresso 软件开发工具包（SDK）的 eIQ 组件中，这些引擎将在本文中得到评估。图 1 左侧列举了：DeepViewRT、TensorFlow Lite / TensorFlow Lite Micro 和 Glow。

2.1 DeepViewRT

eIQ Inference with DeepViewRT™是一款针对 NXP 平台优化的运行时推理引擎，可扩展到各种恩智浦设备和神经网络计算引擎。该推理引擎免费提供，为资源受限的设备生成紧凑的代码，这些设备包括 i.MX RT 跨界 MCU（Arm® Cortex®-M 核）以及 i.MX 应用处理器（Cortex-A 和 Cortex-M 核、专用神经处理单元（NPU）和 GPU）。

2.2 TensorFlow Lite for Microcontrollers

TensorFlow Lite for Microcontrollers（TFLite-Micro）是 Google 公司开发和开源的推理引擎，专为在资源受限的设备上运行机器学习模型而优化，包括恩智浦的 i.MX RT 跨界 MCU。

TFLite-Micro 比原先用于机器学习的开源 TensorFlow Lite 平台更快、更小，能够以更低的延迟和更小的二进制大小进行推理。

2.3 Glow

Glow 机器学习编译器实现了提前（Ahead-Of-Time, AOT）编译。编译器将神经网络转换为目标文件，然后用户将其转换为二进制映像，与传统的运行时推理引擎相比，可以提高性能并减少内存占用。

3 应用开发工作流程

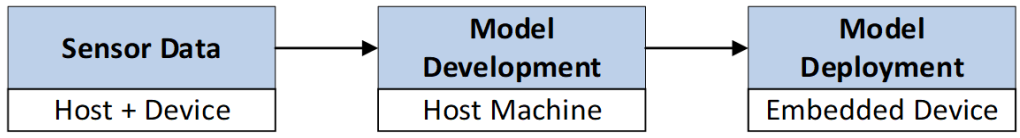

图 2. 应用开发工作流程

应用开发图 2 所示的工作流程：

1. *收集传感器数据*

   此阶段定义并创建用于模型训练和验证的数据集合。运行在嵌入式设备上的应用程序采集传感器数据，并将其传输到宿主机上。

2. *模型开发*

   eIQ 工具包集成了开发工作流工具，还集成了在宿主机上使用的命令行工具，以支持基于视觉模型的机器学习开发。非视觉模型的开发和训练则可以用现有的深度学习框架（如 TensorFlow, PyTorch 等）来完成，开发步骤如图 3 所示：

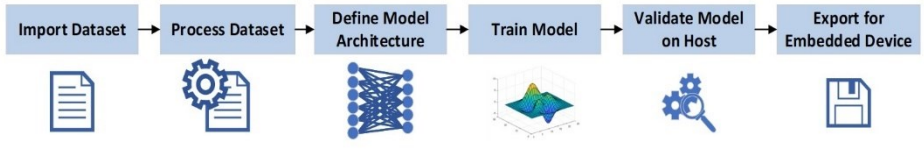

图 3. 模型开发工作流程

3. *模型部署*

   前一阶段创建的模型将通过 *eIQ 和运行时推理引擎介绍* 中定义的运行时推理引擎在嵌入式设备上进行部署和评估。

4 跑通一个真实的用例

本章按照应用开发工作流程中的工作流程图，描述了如何在嵌入式设备上构建和部署能够监测输入传感器数据并检测设备当前状态的模型。主要的目标是：

- 说明如何在宿主机上收集和记录设备状态数据集
- 在宿主机上聚合与分析传感器数据
- 在宿主机上创建、训练和验证分类模型
- 将模型转换为某种适合嵌入式设备的格式
- 评估嵌入式设备上的模型

4.1 前提条件

您需要与此文档一起提供的以下资源来实现基于机器学习的系统状态监控应用程序：

- ML_app – 在带有 Python 内核的 Jupyter Notebook 中开发的用于生成 ML 模型 的宿主机应用程序
- MCU_app – 在 MCUXpresso IDE 中开发的嵌入式设备应用程序

宿主机上的软件要求：

- 基于 ML 的系统状态监测器源代码
- Jupyter Notebook（ Python, TensorFlow, Keras 等 ）
- MCUXpresso IDE
- eIQ 工具包

硬件要求：

- Windows 宿主机
- 目标嵌入式设备
- SD 卡 – SD 卡的使用是可选的，它可以方便地记录传感器的数据
- 传感器板 – FRMD-STBC-AGM01

  如果所需的传感器没有包含在评估套件，也可以使用其它传感器子板。
- 风扇 – 该配置取决于系统设置和可用的部件

  在本案例研究中，一个 5V 的直流风扇被连接到评估套件，并通过 Arduino Proto-Shield 供电。

图 4 显示了如何在 MIMXRT117-EVK 和前面描述的部件上叠加在 Arduino 接口来构建用于评估的嵌入式设置。关于软件和硬件安装的更多细节，请参阅《实验指南》。

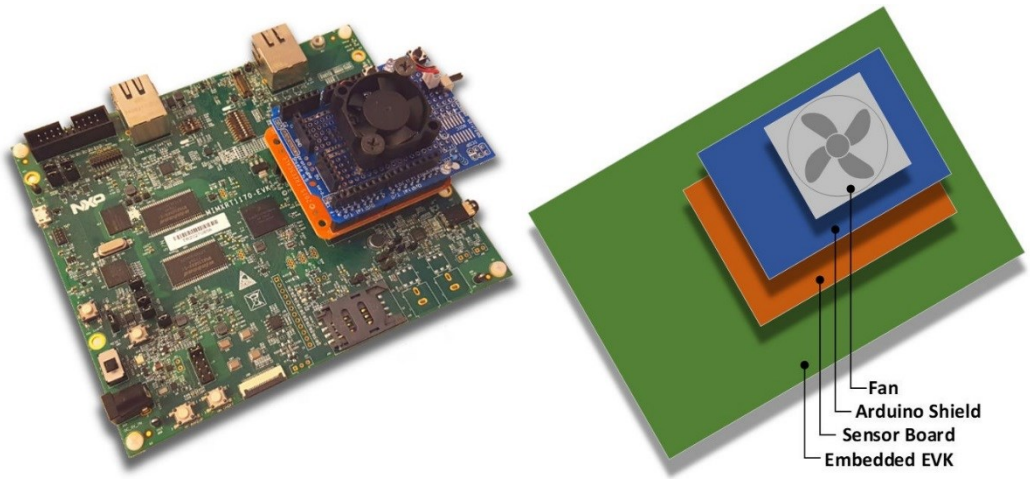

图 4. 叠加嵌入式设置

4.2 案例研究

本应用笔记中举例说明的案例研究是“使用加速计感知的风扇状态分类器”，而且它可以扩展到任何其他用例。

这个应用程序的目的是使用从加速度计读取的数据，运行 DL 模型来确定风扇的当前状态。预定义的类别如下：

- *FAN-OFF* –（关）风扇关闭
- *FAN-ON* –（开）风扇开启，并且运行正常
- *FAN-Clogged* –（堵塞）风扇开启，但是气流受阻
- *FAN-Friction* –（摩擦）风扇开启，检测到叶片过度摩擦甚至被卡住。

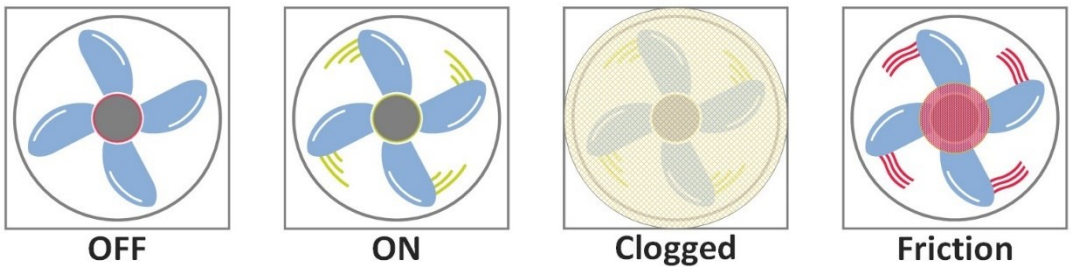

图 5. 风扇状态分类器

4.2.1 传感器数据采集

被监测的输入信号是由 FOXS8700 传感器读取的三轴加速度，其规格如表 1 所示。

表 1. 传感器数据规格

传感器	FXOS8700（6 轴传感器） 3 轴 14 位加速度计 3 轴 16 位磁强计		
使用的通道（Nch）	3 个通道（3 轴加速度计）		
采样频率（Fs）	200 赫兹（5 毫秒）		
输入-输出数据类型/大小	float32/4 字节（每个单独的通道）		
样本数量（秒）	Nch（每个记录的样本包括所有监测的通道）		
窗口尺寸（w）	128 个样本（640 毫秒）		
重叠率	50 % - 64 个样本（320 毫秒）		
输入张量的形状	（w, 1, Nch）=（128, 1, 3）		
输入张量的大小	1.5 KB（w(128) * Nch(3) * 4 字节）		
数据流格式	交错式		
训练数据收集 **输入张量计数（Nt）** **持续时间**	5.76 兆样本 90 千采样窗口 8 小时	1.44 兆样本/类别 22 千样本窗口/类别 2 小时/类别	[兆样本] [千样本窗口] [时间]
验证数据收集 **输入张量计数（Nt）** **持续时间**	1.68 兆样本 26 千采样窗口 2 小时 20 分钟	0.42 兆样本窗口/类别 6 千样本窗口/类别 35 分钟/类别	[兆样本] [千样本窗口] [时间]

图 6 和图 7 所示的交错数据流格式和移动窗口方法被用来采集和转换数据集成合适的形状，以输入给神经网络。例如，由神经网络处理的前三个张量的格式如下：

图 6. 交错数据流格式

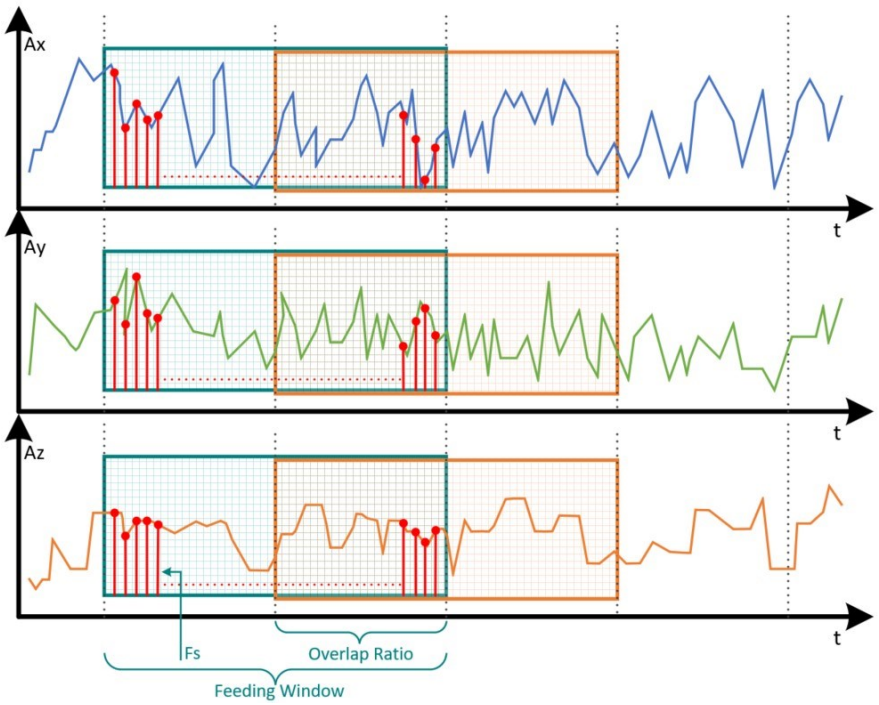

图 7. 使用交错样本和移动窗口给模型提供数据

与图像或音频不同，时间序列传感器数据对于产品设置来说往往是独一无二的，取决于传感器类型、传感器配置、位置、表面等等。因此，我们不能使用已经有的数据集，我们必须创建（收集和标注）一个与这个特定设置对应的数据集。

数据集的收集是通过启用嵌入式应用程序来记录传感器数据（存储在 SD 卡上或通过串行调试接口发送给宿主机），并在终端提示用户配置数据采集方式：

```
/* Configure the action to be performed */
#define SENSOR_COLLECT_ACTION                SENSOR_COLLECT_LOG_EXTERNALLY

Provide the required configuration on the terminal to start the recording:
Class to record (provide only the numeric index): >>> 0
Duration in minutes: >>> 5
SD card filename: >>> Vd1-clog.csv
```

嵌入式设备将通过调试控制台接口（通过 USB 转 UART，115200 bps，8 个数据位，1 个停止位，无奇偶校验，无流量控制）与宿主机进行通信（日志信息、传感器数据、输入配置），因此必须在宿主机上使用一个串行终端应用程序（例如 PuTTY）来通信（检查输出，保存传感器数据，提供配置）。

机器学习模型的目标是从实例（训练数据）中学习，使模型能够泛化到未见过的样本上（验证/测试数据）。因此，在对数据进行训练集和验证集的分割后，我们必须尽量采集到涵盖各种状态下变化的数据。

我们必须为每个类别进行配置，如图 8 所示，以捕获定义的所有状态，并且需要为每个特定的类别采集和存储数据。采集数据的主要规则是尽可能多地涵盖每种情况，而不是使用相同的数据进行训练和验证。在决定各个模式下的数据采集多久，以及如何在类别间切换时，根据实际的使用场景以及背后的数据分析机制。对于本文的案例，每个记录会话都收集一个平衡的数据集（为每个类别采集均匀的样本分布并且各类别采集相同数量的样本，有助于提高模型的精度），试图覆盖尽可能多的变化。

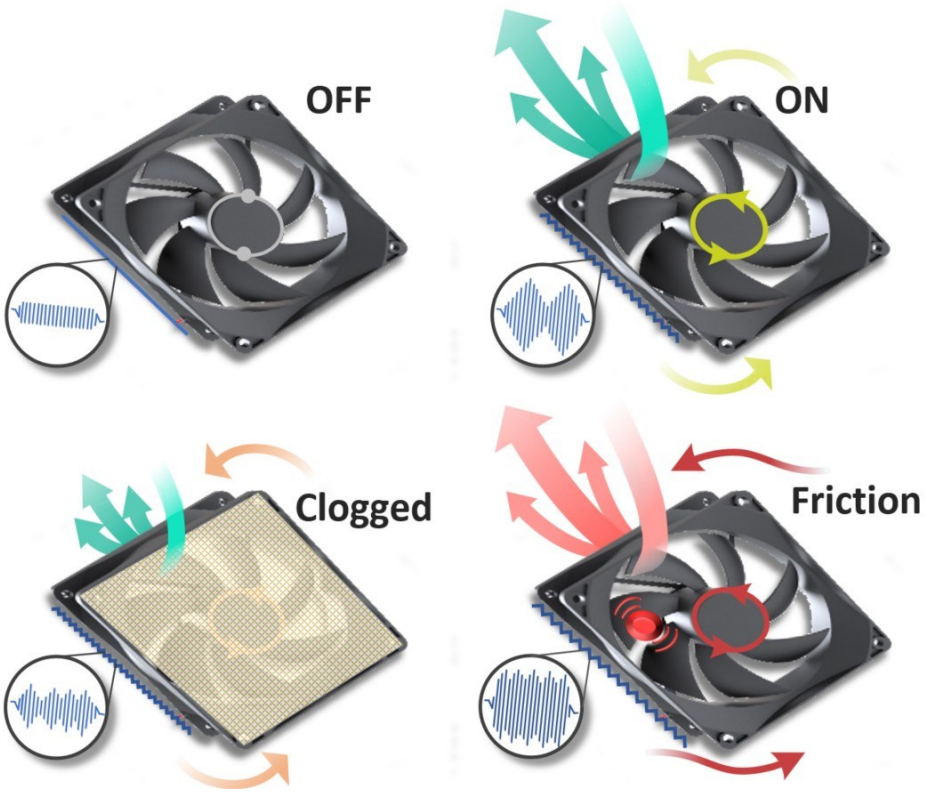

图 8. 为记录每个类别所配置的设置

图 9 显示了一个活动期间的记录，记录开始时采集了一个短时间（5 分钟）的验证数据集，随后采集了一个较长时间（1 小时+5 分钟）的训练和验证数据集，最后又采集了一个短时间（5 分钟）的验证数据集。对于所展示的案例研究，很容易采集“关”和“开”类别的数据，而对于“堵塞”类别，用一块薄纸板来阻挡气流；对于“摩擦”类别，用纸板的一角压在叶片上以产生摩擦。

图 9. 一个活动期间的数据记录的时间轴

图 10 显示了在一次录制会话期间记录的验证集数据快照。这些文件捕获了风扇的状态、采样时间、加速度计的 X、Y 和 Z 轴、磁力强计的 X、Y 和 Z 轴，以及温度（T）。我们的模型只使用时间和加速度计数据，而其他应用可能会发现这些其他领域对他们的深度学习模型很有用。

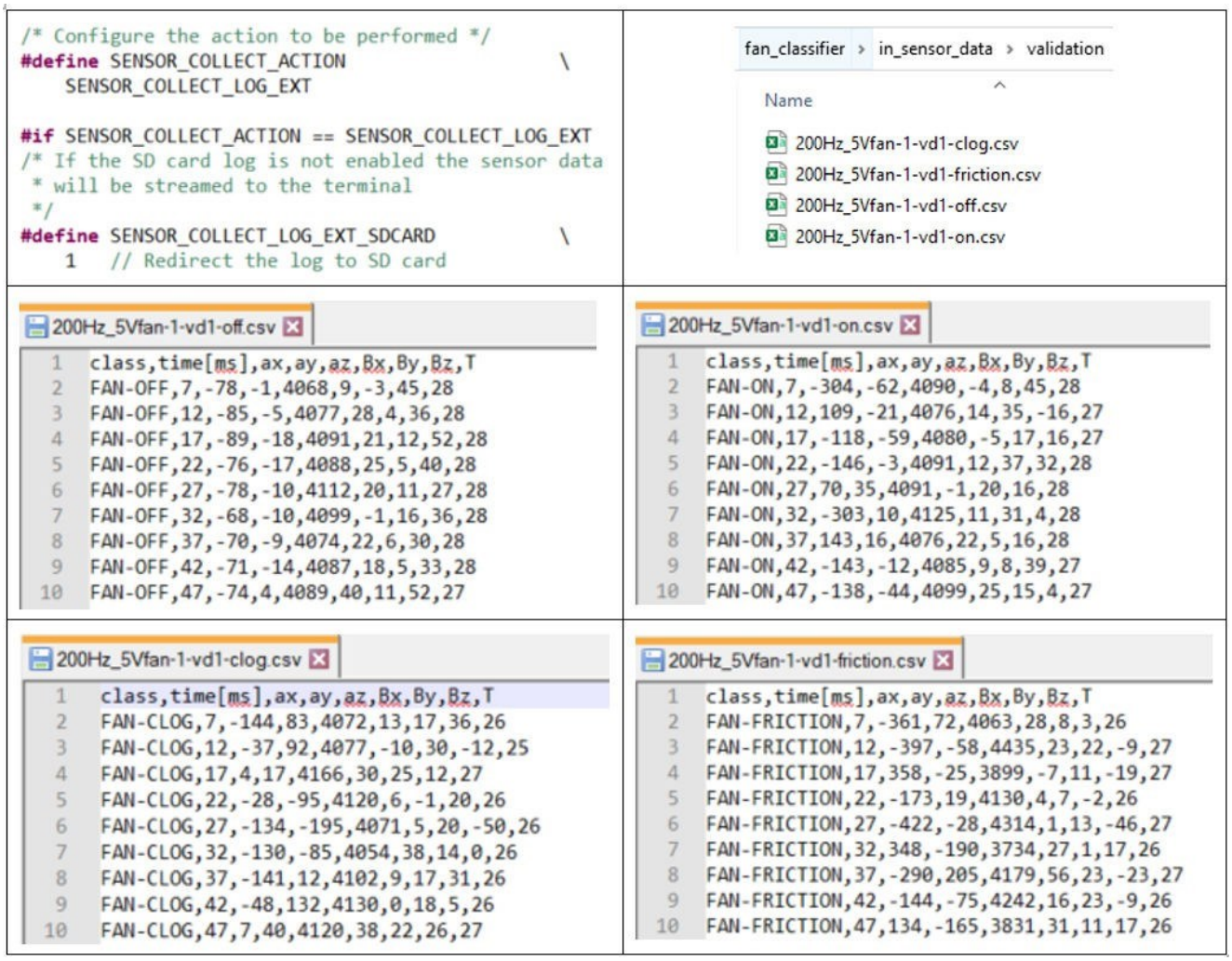

图 10. 在一个活动期间所采集的数据

4.2.2 模型开发

“应用开发工作流程”一章和图 3 中描述了模型开发流程。该流程通过在带有 Jupyter Notebook 的宿主机上运行，被应用于本案例。

4.2.2.1 导入数据集

储存在`/in_sensor_data/`目录中的传感器数据的文件是通过执行 Notebook 的前几个代码片段导入的，它们会返回导入数据集的摘要。

```
in_sensor_data\validation\200Hz_5Vfan-3-vd3-clog.csv
in_sensor_data\validation\200Hz_5Vfan-3-vd3-friction.csv
in_sensor_data\validation\200Hz_5Vfan-3-vd3-off.csv
in_sensor_data\validation\200Hz_5Vfan-3-vd3-on.csv
Classes: ['FAN-CLOG', 'FAN-FRICTION', 'FAN-OFF', 'FAN-ON']
Full dataset shape: (7440000, 9)
Training dataset size: 22499
Validation dataset size: 6561
```

	class	time[ms]	Ax	Ay	Az	Bx	By	Bz	temp
0	FAN-CLOG	7	-166	68	4090	37	18	3	25
1	FAN-CLOG	12	-31	143	4037	-6	6	-26	25
2	FAN-CLOG	17	-18	37	4150	22	11	33	26
3	FAN-CLOG	22	20	-96	4090	14	3	26	26
4	FAN-CLOG	27	-124	-205	4111	-1	24	-54	26

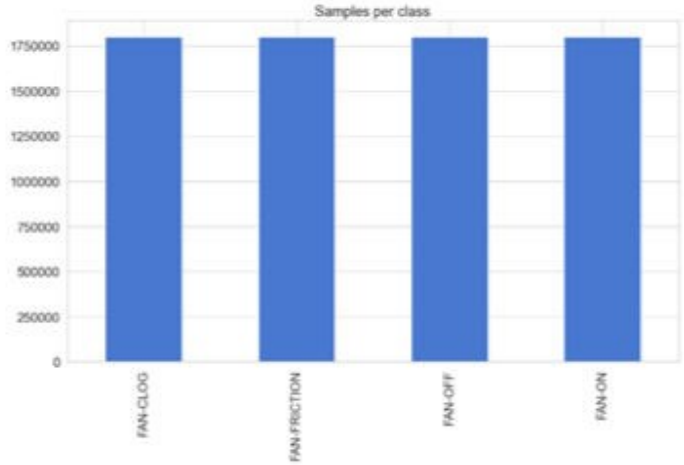

图 11. 数据集摘要

4.2.2.2 数据分析、处理和塑造

数据分析和预处理是可选的，但是这些步骤可以使待分析的数据模式更容易肉眼分辨（图 12 和图 13）。通过使用所提供的 `real_time_plot` 函数绘制实时采集的数据，也可以实现实时数据分析。

在预处理方面，在应用了固定的正常化并将数据集减少到正常范围[-1, 1]之后，可观察到模型性能增加了，如图 14 所示。此外，图 13 中突出显示的频域产生了相当有区别的频谱，它可以被提取并输入神经网络。不过，在这个特定的用例中，只考虑了时域，因为它减少了开发人员的工作和嵌入式处理的开销（也就是说，原始数据被直接送入网络，无需过多的处理），并增加了模型对各种模式的适应性。

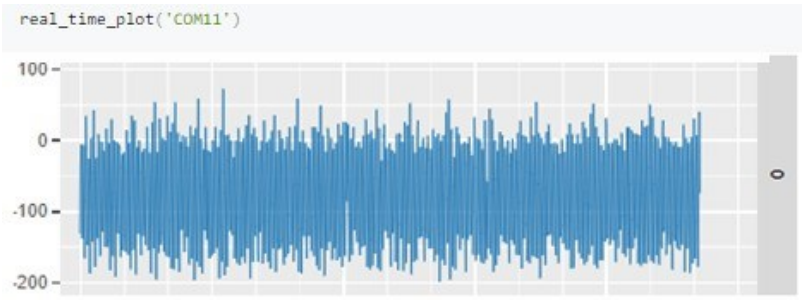

图 12. 实时数据图

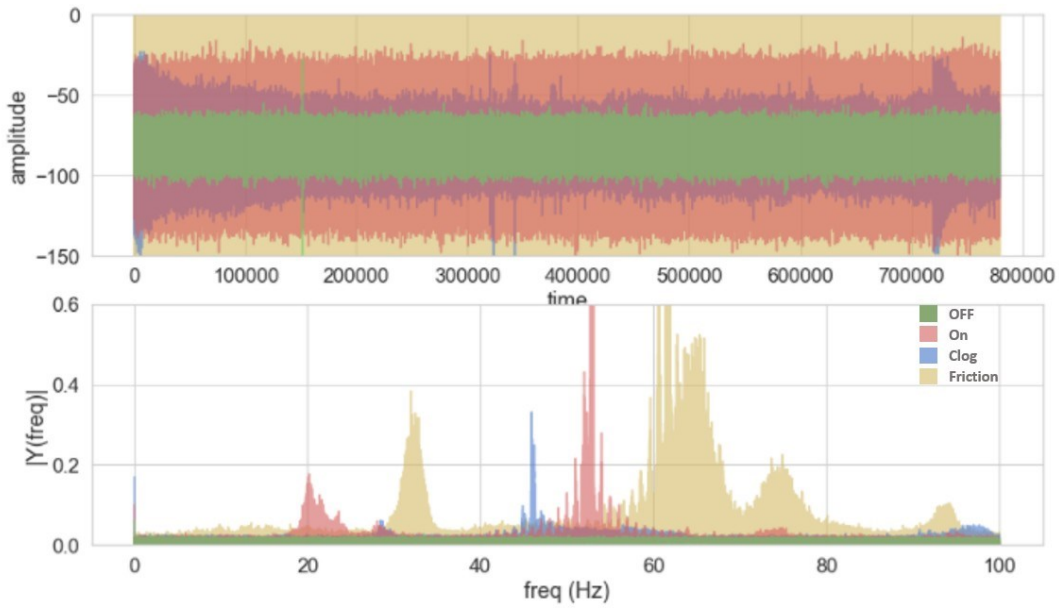

图 13. 时域和频域的数据分析

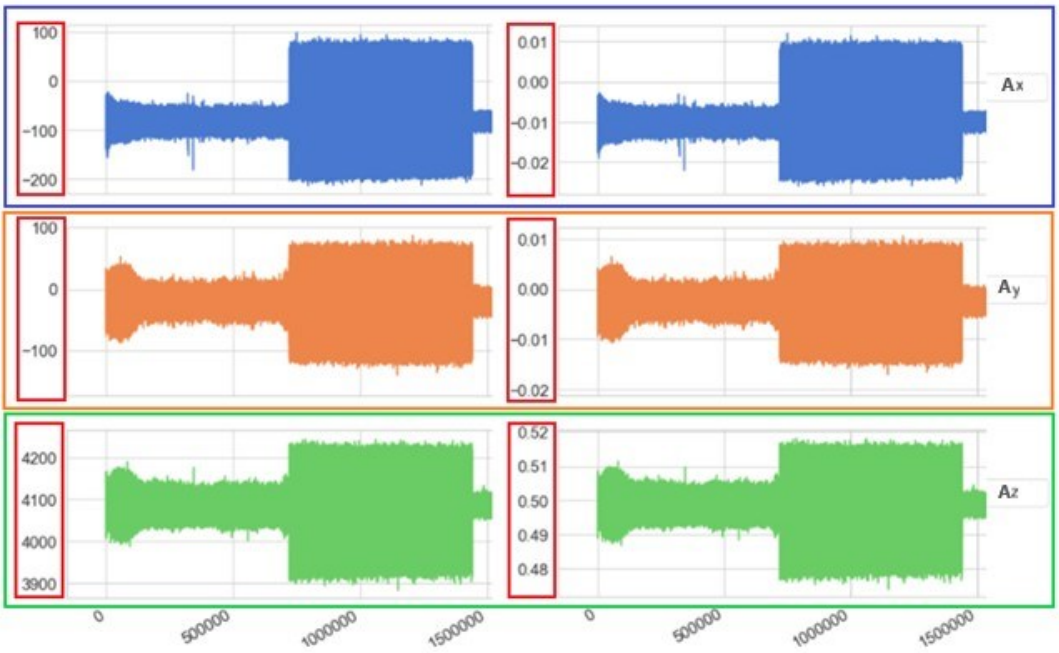

图 14. 数据集正常化

数据重塑的步骤是强制性的，以便能够给模型提供数据以处理输入的信息。数据集被分割并重塑为每个通道的二维张量，从而产生一个重塑的最终数据集，格式如下：

```
dataset = (Nt, height, width, depth) = (Nt, window, 1, Nch) = (Nt, 128, 1, 3)
Nt - the total number of input tensors
```

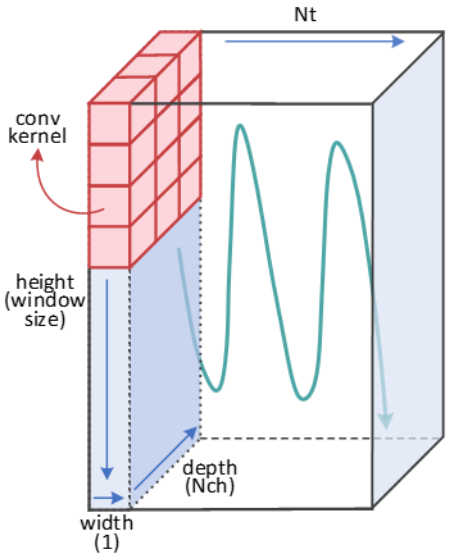

图 15. 数据集和输入张量的形状

4.2.2.3 模型的定义、训练和评估

神经网络的架构可以分为两个阶段：前一阶段用于特征提取，后个阶段用于基于特征计算结果，两者之间还有数据池化层用于去粗取精，滤出显著的特征。

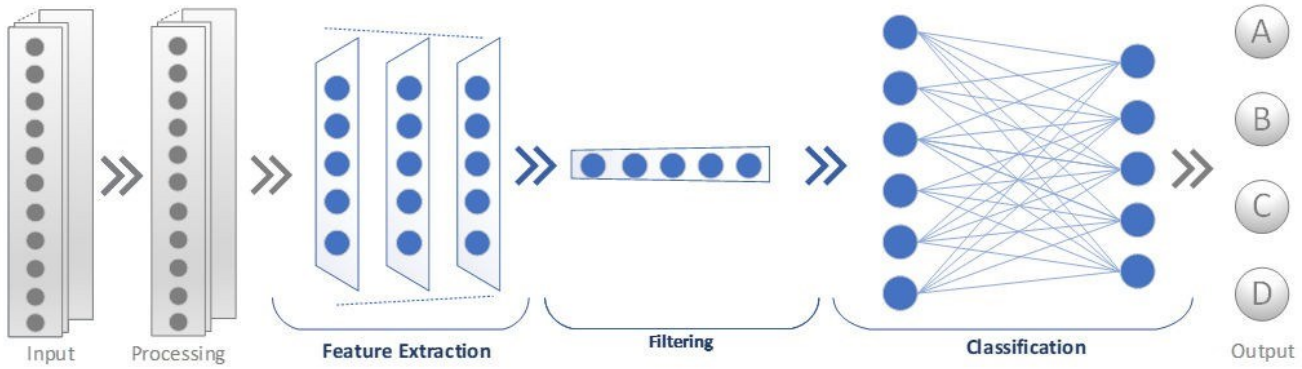

图 16. 神经网络架构

该网络是利用 Keras 框架实现的，它是 TensorFlow 的 Python 深度学习 API。网络的结构和实现如图 16 和图 17 所示，依赖于以下组件和层：

- *Conv* – 由过滤器（filter）和权重组成，把输入数据加工成输出的激活数据（activation）。
- *MaxPool* – 池化层用于减少输入的维度，MaxPool 过滤器选择最显著的特征，降低模型对噪声的敏感性。
- *Flatten* – 将输入拉长成一维特征向量，用于最后的分类阶段。
- *Dense* – 构建全连接层的阶段，用于计算分类结果。
- *Dropout 和正则化* – Dropout 会随机剔除单元/稀释权重，正则化会在误差函数中添加过大权重的惩罚项，它们用于减少过拟合效应，提高模型在新的未见过的数据上表现良好的能力，而不是仅仅记住和过分迎合训练集。
- *Adam 优化器* – 是随机梯度下降的改进版，用于训练网络并根据训练数据迭代更新模型的权重。

- *分类交叉熵* – 是训练中使用的损失函数，训练网络就是要让它的值越小越好。
- *精确度* – 是用来评估模型性能的指标。

```
def model_create(shape_in, shape_out):
    from keras.regularizers import l2

    tf.random.set_seed(RANDOM_SEED)

    model = tf.keras.Sequential()
    model.add(tf.keras.Input(shape=shape_in, name='acceleration'))
    model.add(tf.keras.layers.Conv2D(8, (4, 1), activation='relu'))
    model.add(tf.keras.layers.Conv2D(8, (4, 1), activation='relu'))

    model.add(tf.keras.layers.Dropout(0.5))

    model.add(tf.keras.layers.MaxPool2D((8, 1), padding='valid'))
    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dense(64,
                                    kernel_regularizer=l2(1e-4),
                                    bias_regularizer=l2(1e-4),
                                    activation='relu'))
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(32,
                                    kernel_regularizer=l2(1e-4),
                                    bias_regularizer=l2(1e-4),
                                    activation='relu'))
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(shape_out, activation='softmax'))
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['acc'])

    return model
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 125, 1, 8)         104
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 122, 1, 8)         264
_________________________________________________________________
dropout (Dropout)            (None, 122, 1, 8)         0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 1, 8)          0
_________________________________________________________________
flatten (Flatten)            (None, 120)               0
_________________________________________________________________
dense (Dense)                (None, 64)                7744
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 132
=================================================================
Total params: 10,324
Trainable params: 10,324
```

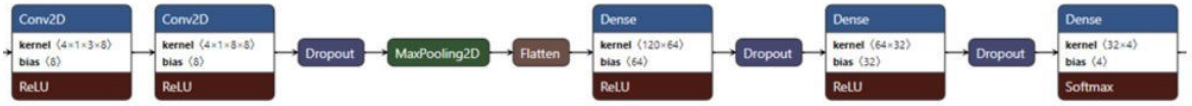

图 17. 模型结构与实现

一旦定义了网络架构，下一步就是开始训练过程。通过训练，网络从训练数据集中学习并自动更新模型参数。模型参数指定如何将输入转化为所需的输出。此外，超参数也可以在这个阶段进行调优。超参数指的是网络结构、架构、使用的优化器、学习率等，并直接影响性能。没有具体的方法来计算超参数，必须手动调优它们，以找到特定用例的最佳配置。当使用验证数据集评估性能时，这个过程通常是基于实验和重复。

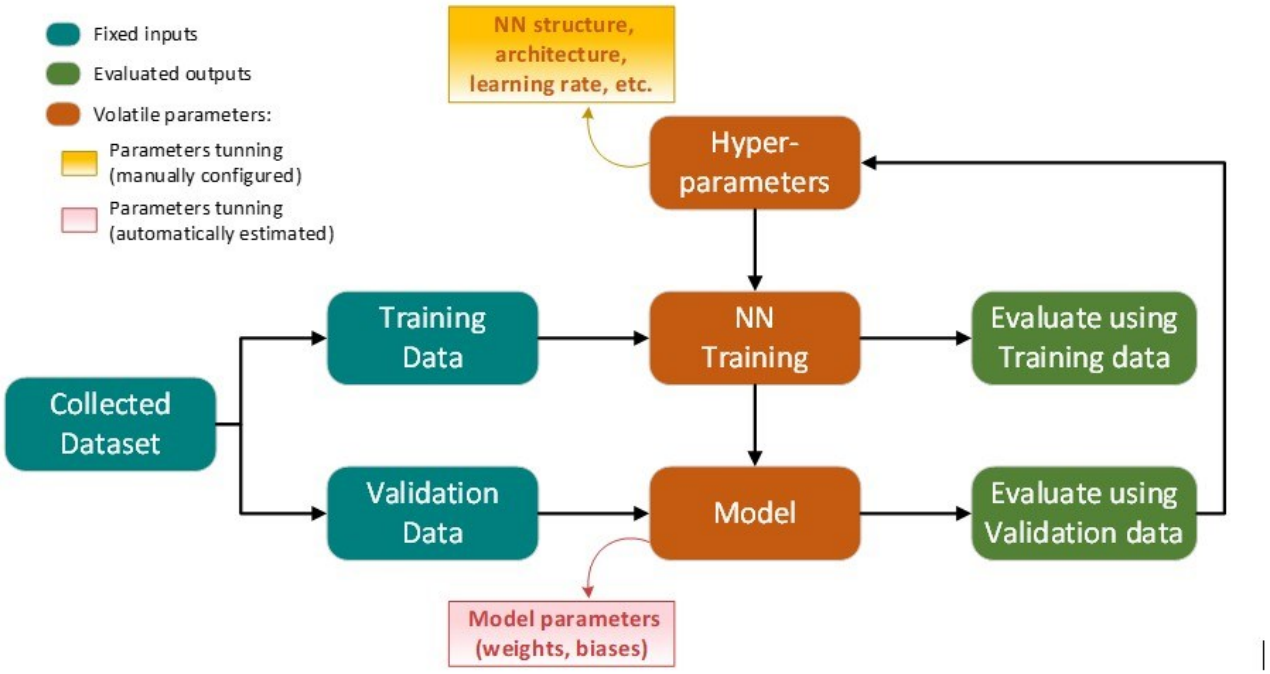

图 18. 模型训练和参数调优

该模型可以在验证数据集上进行评估，可以通过可视化分析准确度和损失，也可以通过使用如图 19 所示的混淆矩阵来评估模型。训练过程的进展可以通过绘制和分析迭代的历史进度来评估，如图 20 所示。

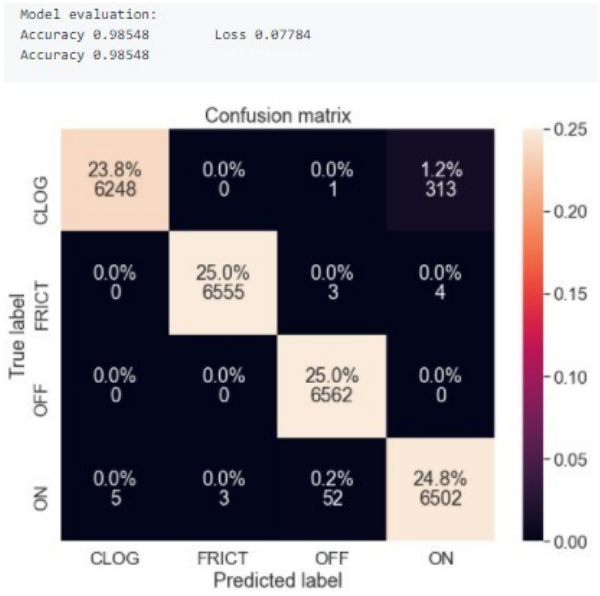

图 19. 使用混淆矩阵进行模型评估

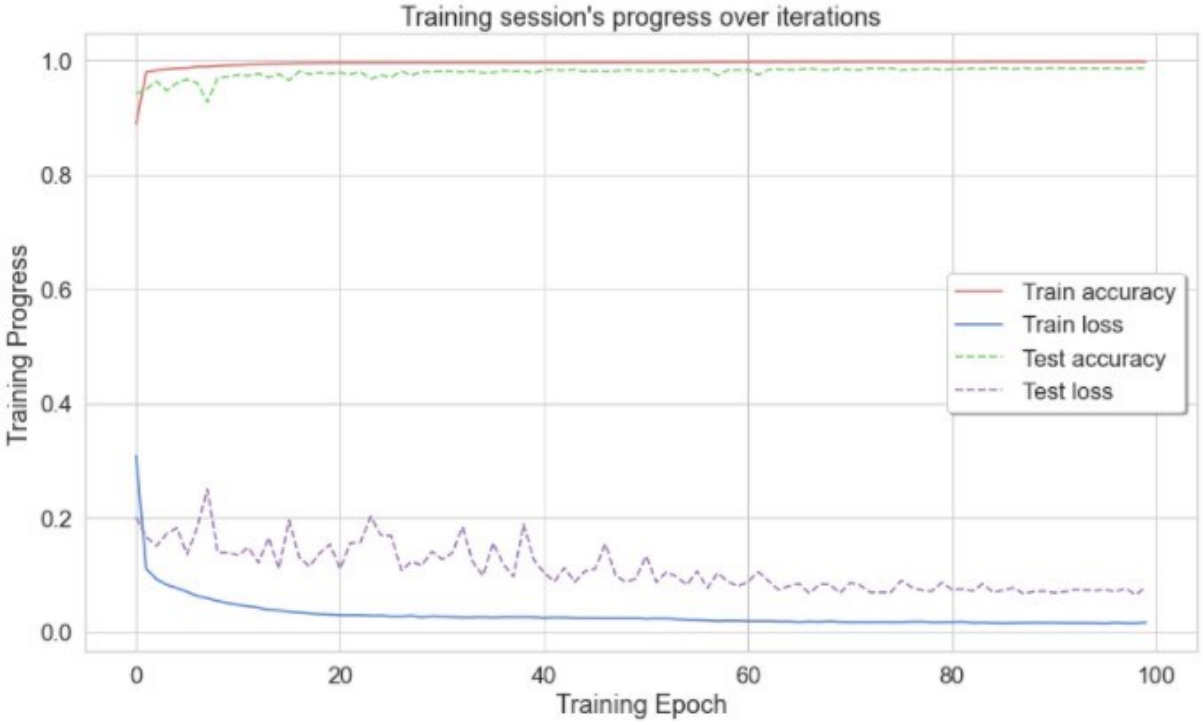

图 20. 训练期间活动评估

4.2.2.4 在宿主机上进行模型验证

如图 19 所示，所建模型获得的准确率高达 99%（训练数据为 99%，验证数据为 98%）。除了通过在以前记录的数据上运行模型来评估模型外，还有一种方法是通过读取和处理嵌入式设备实时提供的传感器数据，用新的未见过的数据来验证在 PC 上运行的模型。为了运行这个验证阶段，电路板必须连接到宿主机上，并配置为通过控制台从外部记录传感器数据。

```
real_time_validation(model, 'COM11')
```

```
Model Evaluation: insert target class: 2
Model Evaluation: insert pool size: 50
Inference FAN-OFF?FAN-OFF | count: 50/50/50
Prediction Accuracy for class FAN-OFF 100.00%
Accuracy 1.00000
```

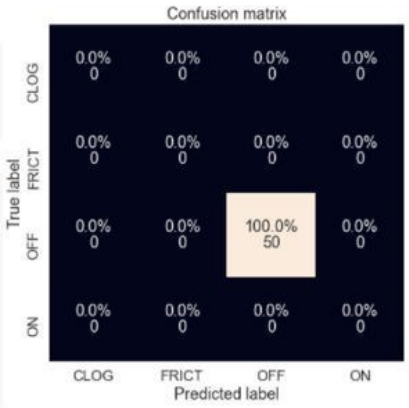

图 21. 在宿主机上进行模型验证

4.2.2.5 模型移植到嵌入式设备

通过使用所提供的 `save_model` 函数，可以将训练好的模型转换为在嵌入式板上运行。这个函数保存了三种类型的模型：完整的 Keras 模型（`model.h5`），转换后的 TFLite 模型（`model.tflite`），以及 TFLite 量化的模型（`model_quant.tflite`）。对于量化，为保持应用程序的兼容性，我们使用一种混合型方法：保持输入/输出张量为高精度的浮点类型（Float32），同时将中间层的参数量化并减少到 int8。

TensorFlow Lite Micro 推理引擎用于在嵌入式电路板上运行以 `tflite` 格式导出的模型，而对于 DeepViewRT 和 Glow，必须执行一些额外的步骤，才能将模型正确地转换为特定的格式。这些步骤依赖于 eIQ Toolkit（eIQ Portal，eIQ Model Tool，eIQ Glow）。更多信息，请参阅 eIQ 和运行时推理引擎介绍一章。

若要使用 RTM 格式（DeepViewRT），必须使用 eIQ Portal 软件套件中的 “模型工具” 进行转换，如图 22 所示。

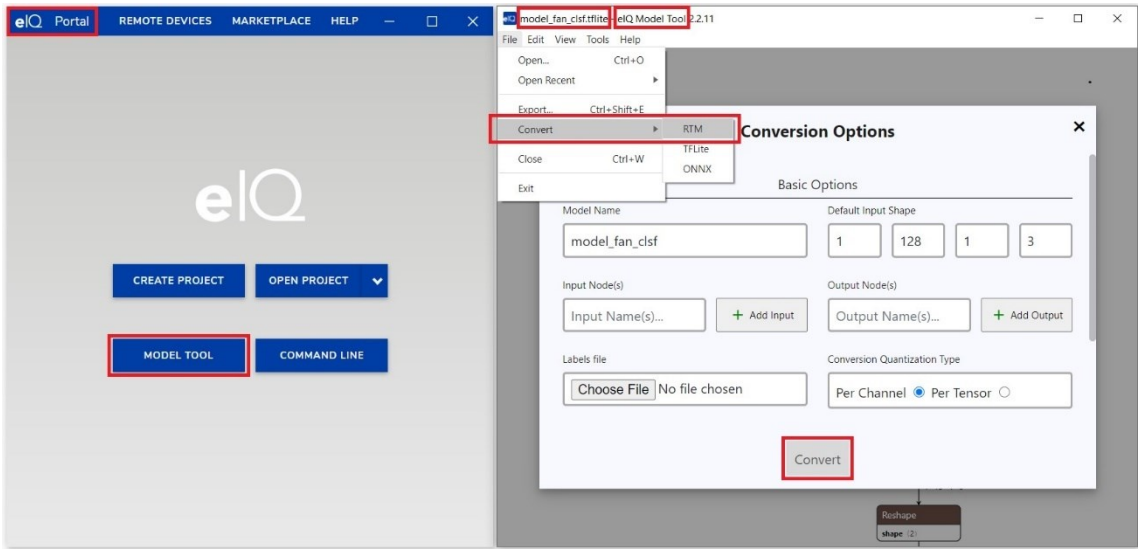

图 22. 转换为 DeepViewRT 的 “模型工具”

对于 glow 格式，需要在命令行中运行 `model-compiler.exe`，并使用以下参数，以生成图 23 所示的软件包（bundle）（`glow_model_fan_clsf.tflite` 只是将 `model_fan_clsf.tflite` 模型的重命名复本，以便于文件区分）。取决于嵌入式设备的目标，*-target* 和 *-mcpu* 参数可能需要改变。

```
model-compiler.exe -model=models\model_fan_clsf\glow_model_fan_clsf.tflite -emit-
bundle=models\model_fan_clsf\bundle -backend=CPU -target=arm -mcpu=cortex-m7 -float-abi=hard -use-
cmsis
```

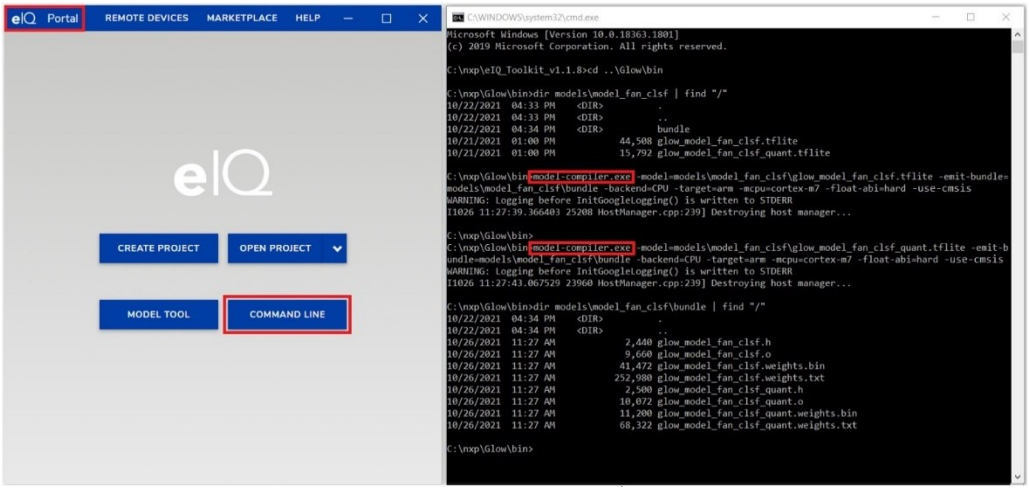

图 23. 模型转换为 Glow 格式

4.2.3 模型在嵌入式设备上的部署和评估

一旦模型生成，你可以使用所提供的 `mcu_app` 应用程序和 MCUXpressoIDE 在嵌入式电路板板上部署、运行和评估它们。

源代码结构在图 24 中突出显示，包括：

- `mcu_app/doc/readme.txt`：应用程序的简短文档

  关于硬件要求、电路板设置和运行该演示的指南。
- `mcu_app/board/board.h`：包含与电路板配置有关的设置
- `mcu_app/board/frdm_stbc_agm01_shield.h`：包含与所使用的传感器有关的设置。

  如果使用了可选的传感器工具箱（FRMD-STBC-AGM01），则连接上传感器盾板。
- `mcu_app/source/inf-eng`：包含用于运行推理引擎的 API
- `mcu_app/source/models`：包含移植到每种特定格式上的模型,以及用于测试的验证数据。按照模型移植到嵌入式设备指南来移植的模型必须存储在这个目录中。
- `mcu_app/source/models/model_selection.h`：选择量化和非量化版本，以及如何加载模型（从 Flash 或从 RAM），用户必须正确配置。

  如果选择使用 glow 量化版本，模型的名称也必须在项目设置中按照这个路径进行配置：右击项目>属性>C/C++构建>设置>工具设置>MCU C++链接器> 杂项> 其他对象（英文界面的 IDE 是 `the project> Properties> C/C++ Build> Settings> Tool Settings> MCU C++ Linker> Miscellaneous> Other Objects`）。
- `mcu_app/source/models/validation_data/`：用于离线验证的预先记录的数据；
- `mcu_app/source/sensor/sensor_collect.c`：主线程；
- `mcu_app/source/sensor/sensor_collect.h`：由用户配置和选择应用程序的运行时行为；
- 从外部记录传感器数据：

```
/* Configure the action to be performed */
#define SENSOR_COLLECT_ACTION                   SENSOR_COLLECT_LOG_EXT
#if SENSOR_COLLECT_ACTION == SENSOR_COLLECT_LOG_EXT
#define SENSOR_COLLECT_LOG_EXT_SDCARD           1  // Redirect the log to SD card
```

- 运行选定的推理引擎，在实时数据或预先记录的验证数据上计算预测结果（通过修改 `SENSOR_FEED_VALIDATION_DATA` 宏以在实时或离线验证之间切换）：

```
/* Configure the action to be performed */
#define SENSOR_COLLECT_ACTION                   SENSOR_COLLECT_RUN_INFERENCE

#if SENSOR_COLLECT_ACTION == SENSOR_COLLECT_RUN_INFERENCE
#define SENSOR_COLLECT_RUN_INFENG               SENSOR_COLLECT_INFENG_TENSORFLOW
#define SENSOR_FEED_VALIDATION_DATA             1   // Feed the model with data recorded
previously for validation
#define SENSOR_RAW_DATA_NORMALIZE               1   // Normalize the raw data
#define SENSOR_EVALUATE_MODEL                   1   // Evaluate the model performance by
computing the accuracy
#define SENSOR_COLLECT_INFENG_VERBOSE_EN        0   // Enable verbosity
```

- 当配置了离线验证时，来自 `validation_data/` 目录的预先记录的数据将被用于评估。根据评估的类别，引用验证数据的指针需要在 `sensor_collect.c` 文件中进行配置。

```
 /* Replace with the buffer that contains the data recorded
 * for the specific class that will be evaluated
 * (i.e., vdset_3_vd3_clog, vdset_3_vd3_friction,
 *  vdset_3_vd3_off, vdset_3_vd3_on) */
static const float *vdset_ptr = &vdset_3_vd3_clog[0][0];
```

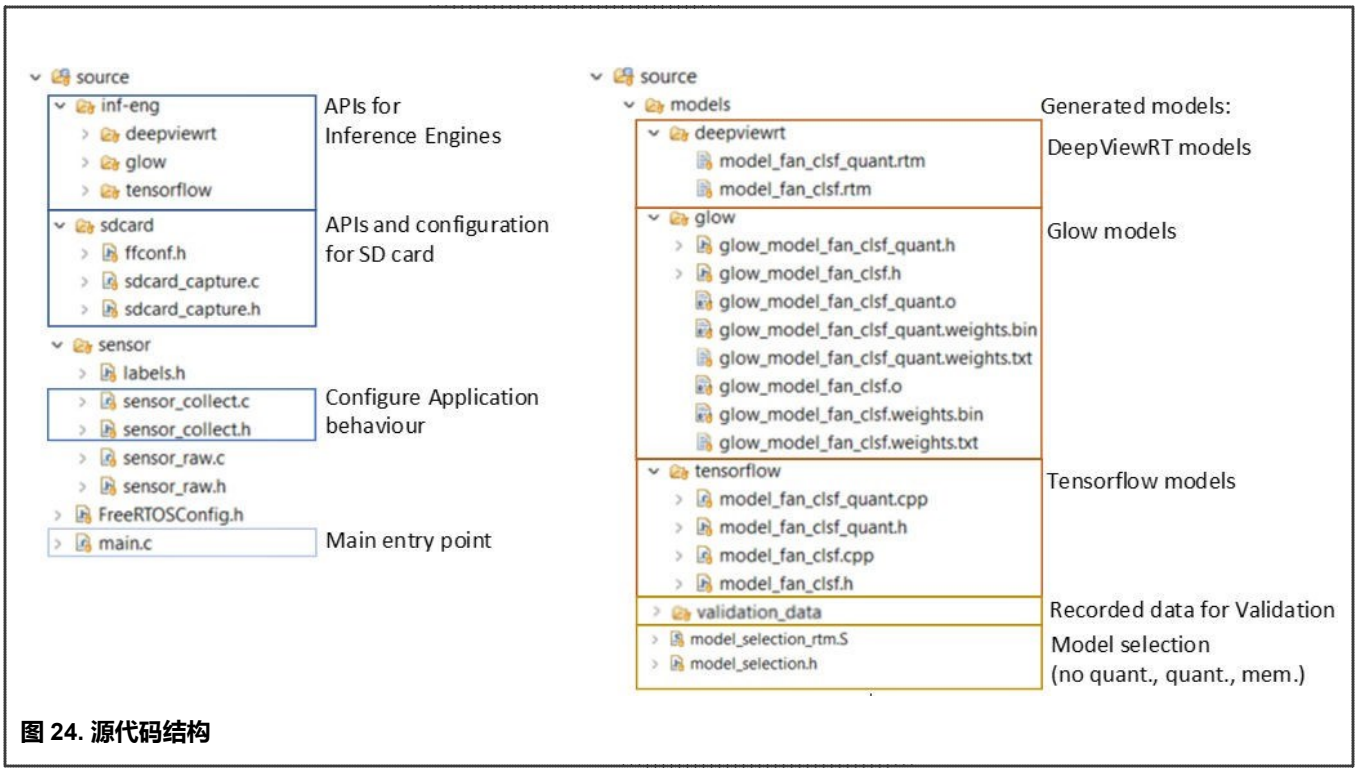

图 24. 源代码结构

设置完成后，编译下载应用程序到目标电路板上，然后通过按下复位按钮或在 IDE 中启动调试来运行应用程序。电路板闪烁后，终端显示“Start Application…”，应用程序运行。示例输出如下：

```
Starting Application...
MainTask started
SENSOR_Collect_Task started
Model loaded to SDRAM...

Model Evaluation:
Class to evaluate (provide only the numeric index):
( 0:FAN-CLOG 1:FAN-FRICTION 2:FAN-OFF 3:FAN-ON )
        >>> 0
Pool size (total number of predictions to compute):
        >>> 936

Inference 0?0 | t 825 us | count: 732/936/936 | FAN-CLOG
Prediction Accuracy for class FAN-CLOG 78.21%
```

准确率的计算方法是将*正确的预测数*除以*总的预测数*。当模型在电路板上被评估时，需要提供两个参数：

1. 目标类别 – 指定哪个类别将被评估，哪个类别的精度将被计算（即，0 为风扇-阻塞，1 为风扇-摩擦，2 为风扇-关闭，3 为风扇-开启）。
2. 池大小 – 将被计算的预测总数。在上面的快照中，阻塞状态是从 `936` 个输入张量的池中评估的。所提供的池大小值实际上是录制会话 3 中记录的最后一个验证数据集（`vdset_3_vd3_clog.h`）的大小，并导出到电路板上进行离线验证：

```
Pool size = 936 (60000/64 - 1)
Number of samples in the dataset = 60000 (5minutes of recording at 200Hz)
Moving Window size = 64 samples
Remainder = 1
```

另外，在使用同一份离线预录制的数据时，不管是在电路板上运行模型，还是在宿主机上运行模型，都会得到一模一样的结果，如图 25 和图 26。

注意

当使用浮点模型与量化模型运行预测时，可以观察到计算精度的轻微偏差（最高 0.3 %）。这种行为是完全可以预期的，因为量化是一个有损失的过程，它将模型转换为较低的精度数据。

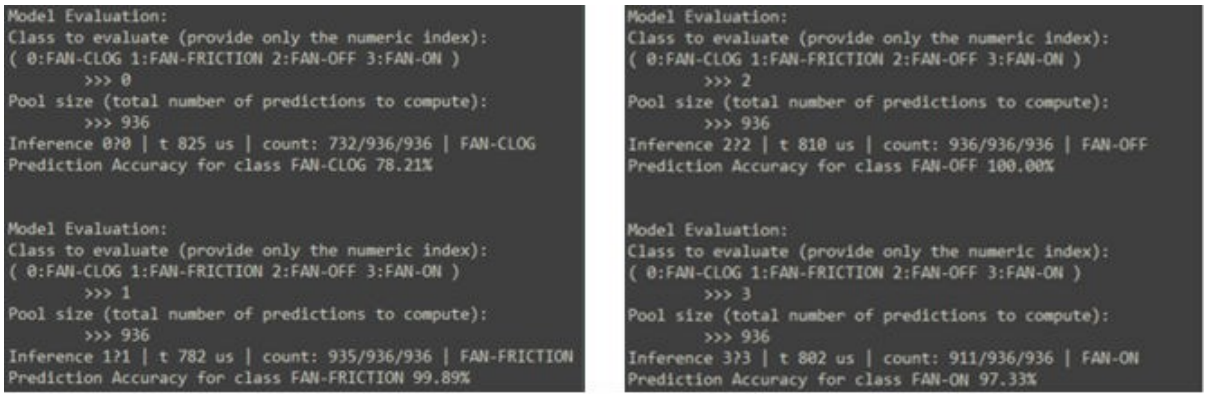

图 25. 使用预先记录的验证数据在电路板上进行模型评估

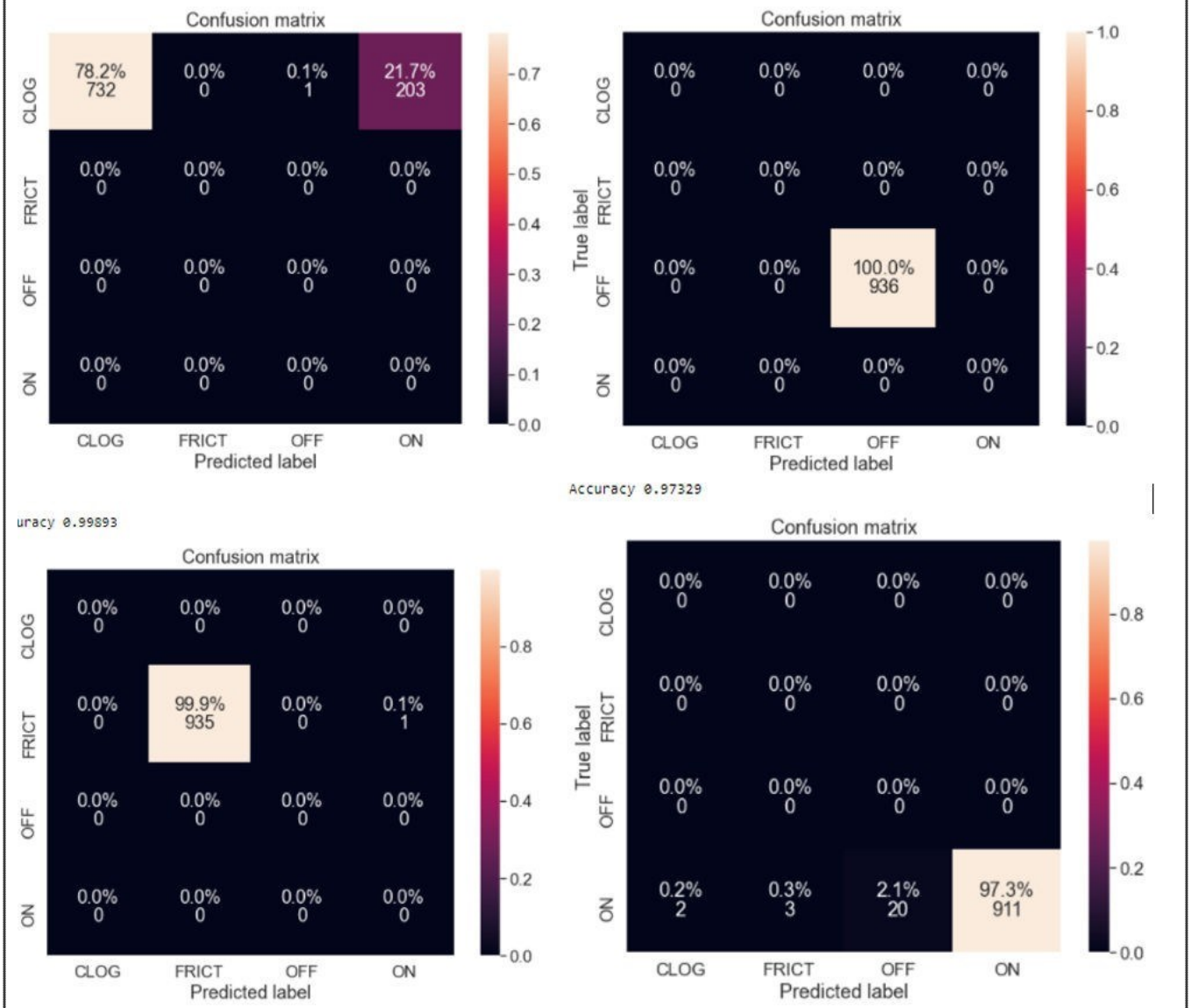

图 26. 使用预先记录的验证数据在宿主机上进行模型评估

表 2 列出了在 i.MXRT1170-EVK（Cortex-M7 核）上运行和评估这些模型得到的结果。图 27 和图 28 通过图表显示了基准测试结果。

*推理时间*是处理一个输入张量的平均时间，表示在嵌入式电路板上需要花多少时间来运行模型以得到预测结果和对单个状态进行分类。我们的实验条件包含从 RAM 中和从 Flash 中运行模型，并且分别在 996MHz 和 150MHz 的不同 CPU 频率。

内存使用情况通过每个推理引擎的*模型大小和代码大小*来呈现。

模型大小由常量参数和可变权重组成。常量参数存在于闪存中，也可以从 RAM 内存中加载，以加快推理时间；可变权重存在于 RAM 中。

在推理引擎的内存占用上可以观察到一些明显的差异，主要是因为模型非常小，但这个比例对于较大的模型来说就不那么明显了。

表 2. 在嵌入式电路板上进行模型评估

		TFLite（无量化）	TFLite（量化）	DeepViewRT[1]（无量化）	DeepViewRT[1]（量化）	Glow（无量化）	Glow（量化）
推理时间（毫秒）							
@996 MHz	RAM	0.74	0.48	1.16	1.14	0.35	0.17
	Flash	1.46	0.55	1.77	1.31	0.98	0.23
@156 MHz	RAM	4.89	1.92	3.50	4.32	2.19	1.08
	Flash	5.31	1.93	3.80	4.44	2.42	1.09
模型大小（KB）							
Flash/RAM (const)		43.5	15.4	44.3	15.4	40.5	10.6
RAM (var)		9.4	4.5	11.3	6.3	9.6	3.7
代码大小（KB）：							
Flash		56.3	60.9	109.2	109.2	10.9	10.4

1. DeepViewRT 推理引擎的评估软件版本将在 2022 年第一季度结束前发布

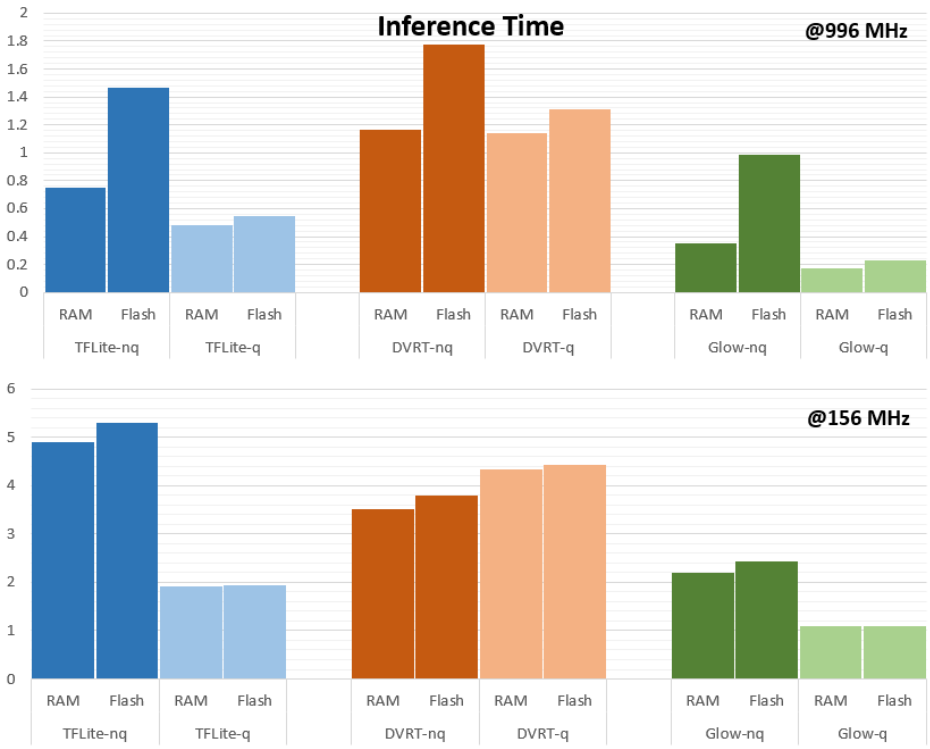

图 27. 模型性能 - 推理时间

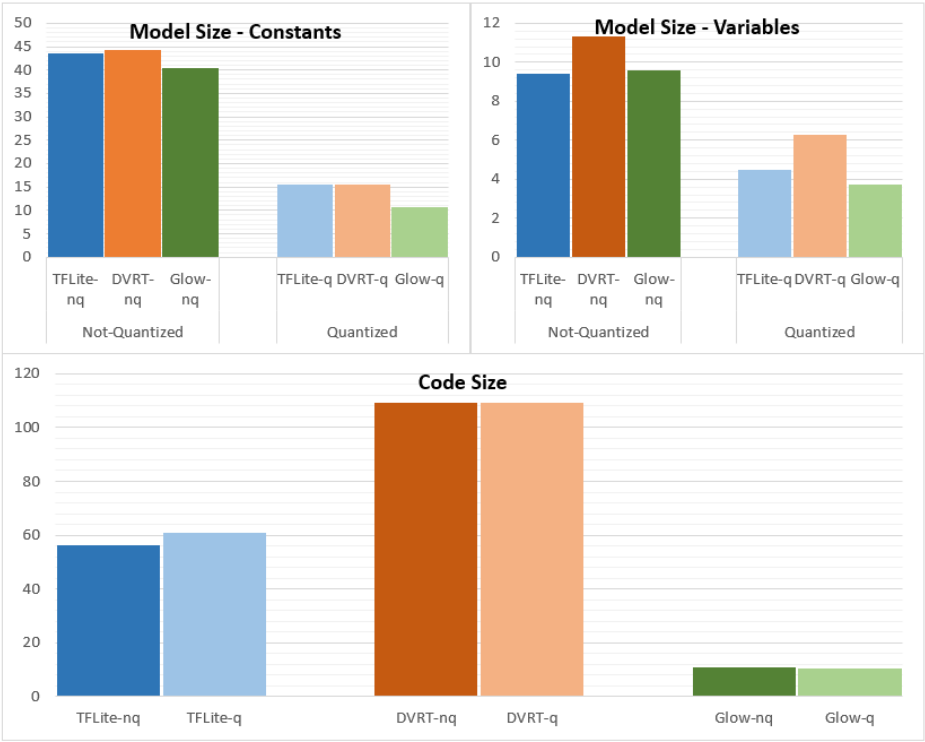

图 28. 内存使用情况

表 3 和图 29 列出了在多个嵌入式设备上运行模型所评估的推理时间，这些设备的处理核心被配置为 150MHz。

表 3. 在多个嵌入式设备上评估的推理时间

推理时间（毫秒）@150 MHz	TFLite（无量化）	TFLite（量化）	Glow（无量化）	Glow（量化）
MIMXRT1170-EVK (Arm Cortex-M7)	4.89	1.92	2.19	1.08
FRDM-K66F (Arm Cortex-M4)	8.87	4.18	4.24	5.47[1]
LPC55S69-EVK (Arm Cortex-M33)	9.95	5.77	4.16	3.88

1. 由于 Glow 模型编译器目前还不支持 Arm Cortex-M4 的 CMSIS-NN 加速，预计 Glow 量化模型比浮动模型的推理时间更长。

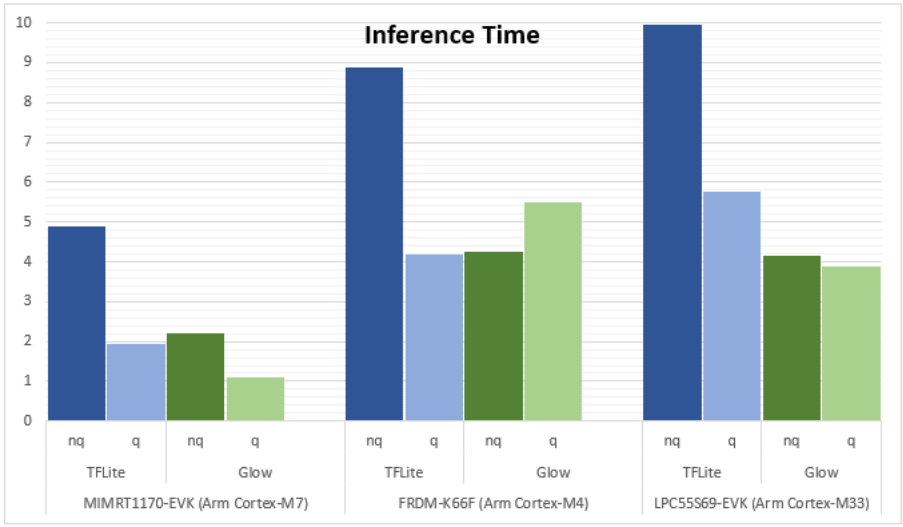

图 29. 在多个嵌入式设备上评估的推理时间

5 总结

本应用笔记提供了在 MCU 上构建和运行智能传感设备的指南，依靠 深度学习来解决特定问题。为此，本文通过一个真实的用例，说明了利用恩智浦的 SDK 和 eIQ 技术，在嵌入式电路板上生成和加工数据集、定义神经网络架构、训练和部署模型所需的步骤。

本文件还说明了使用哪些评价标准，以及如何在嵌入式电路板上评估神经网络模型的行，并且对性能、精度、RAM 占用、Flash 占用进行评测。

6 修订历史

下表总结了自初版发布以来对该文件所做的修改。

表 4. 修订历史

版本号	日期	实质性变更
1	2022 年 4 月 25 日	增加了 MIMXRT1170-EVK（CM7）、FRDM-K66F（CM4）和 LPC55S69-EVK（CM33）的推理时间基准测试
0	2022 年 2 月 03 日	初版发布

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Airfast — is a trademark of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Cadence — the Cadence logo, and the other Cadence marks found at www.cadence.com/go/trademarks are trademarks or registered trademarks of Cadence Design Systems, Inc. All rights reserved worldwide.

CodeWarrior — is a trademark of NXP B.V.

ColdFire — is a trademark of NXP B.V.

ColdFire+ — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

EdgeScale — is a trademark of NXP B.V.

EdgeVerse — is a trademark of NXP B.V.

eIQ — is a trademark of NXP B.V.

FeliCa — is a trademark of Sony Corporation.

Freescale — is a trademark of NXP B.V.

HITAG — is a trademark of NXP B.V.

ICODE and I-CODE — are trademarks of NXP B.V.

Immersiv3D — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

Kinetis — is a trademark of NXP B.V.

Layerscape — is a trademark of NXP B.V.

Mantis — is a trademark of NXP B.V.

MIFARE — is a trademark of NXP B.V.

MOBILEGT — is a trademark of NXP B.V.

NTAG — is a trademark of NXP B.V.

Processor Expert — is a trademark of NXP B.V.

QorIQ — is a trademark of NXP B.V.

SafeAssure — is a trademark of NXP B.V.

SafeAssure — logo is a trademark of NXP B.V.

StarCore — is a trademark of NXP B.V.

Synopsys — Portions Copyright © 2021 Synopsys, Inc. Used with permission. All rights reserved.

Tower — is a trademark of NXP B.V.

UCODE — is a trademark of NXP B.V.

VortiQa — is a trademark of NXP B.V.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25 April 2022
Document identifier: AN13562