Application Note

在 AM6xA MPU 上为自动零售扫描仪构建边缘 AI 应用

Reese Grimsley

摘要

本应用手册介绍了在 TI AM6xA 处理器上为视觉任务构建边缘 AI 应用程序的分步过程，并以 AM62A 和零售扫描仪演示为例展示了该过程。遵循此开发流程可大大加速新设计进程。本文档详细介绍了用于训练、编译和运行模型的 TI 处理器特定步骤。此外，本文档还提供了有关在 Linux 中创建 gstreamer 流水线的信息，该流水线利用硬件加速器进行图像处理，使开发人员无需手动对这些加速器进行编程。

内容

插图清单

表格清单

商标

1 引言

本文档旨在介绍对德州仪器 (TI) AM6xA 微处理器创建边缘 AI 视觉应用程序的工具和流程。示例应用将 AM62A 用于自动零售扫描仪或结算系统，在该系统中，顾客盘内的食品可通过深度学习神经网络快速自动识别，并由 TI 的 C7xMMA 深度学习加速器进行加速。

本应用手册遵循图 1-1 中所示的总体开发流程，从自定义数据集和 TI Model-Zoo 中经过部分训练的模型开始 [1]。神经网络模型使用 TI 支持的工具进行训练，并编译为目标器件的 C7xMMA 架构。然后，它与 Gstreamer 一起用于开发端到端媒体流水线，包括摄像头采集、预处理、深度学习推理、后处理和显示到监视器。最终应用程序用 Python3 编写，开源代码可在网上的德州仪器 (TI) Github 零售-购物目录下找到 [2]。有关如何运行演示的说明，请参阅 README.md。

该应用程序是为 AM62A 开发的，AM62A 是德州仪器 (TI) 的一款具有 2TOPS 深度学习加速功能的四核微处理器。该模型针对该架构进行编译，需要重新编译才能在另一个 TI 处理器上运行，从而充分利用加速器。要了解有关此编译过程的更多信息，请参阅 TI 边缘 AI 存储库中的相关文档 [4] 和 edgeai-tidl-tools [5]。此应用程序曾在 2023 年国际嵌入式展 (Embedded World) 上亮相，用于展示 AM62A。

在实施时，TI 在 Edge AI Cloud [6] 中的模型编写器软件套件尚不可用于此处理器 – 这些工具简化了从数据采集到模型选择再到训练、编译和评估（如图 1-1 中所示），一直到模型评估的整个神经网络模型开发流程。本文档遵循更加程序化的设计流程，为有经验的开发人员提供了更大的灵活性。

通过以下链接可在 YouTube 上找到概念演示视频：https://www.youtube.com/watch?v=jYJvtoPAW6E。

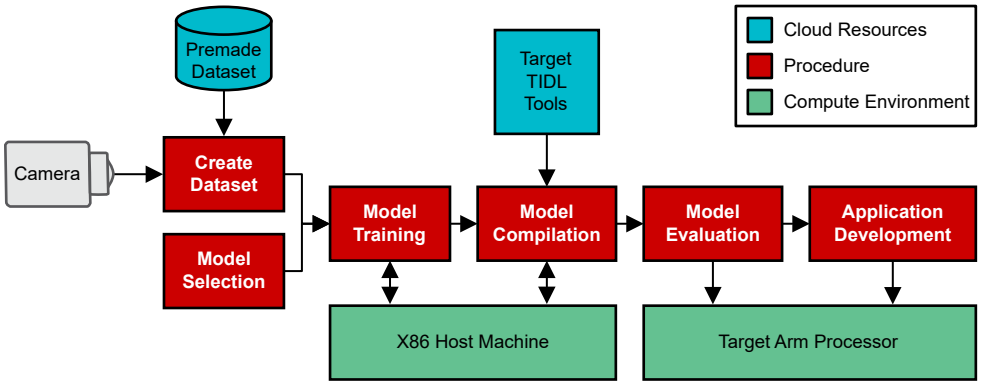

图 1-1. 开发流程图

1.1 目标读者

本文档适用于具有初级或中级深度学习经验的读者，他们希望将自己的想法带入适用于视觉应用的 TI 处理器。本文档还旨在帮助读者在实施此类系统时增加实用知识；并特意为具有理论知识但实际专业知识较少的读者提供更多提示和意见。

对于不太了解深度学习的读者，建议参考 TI 开发人员专区中 Edge AI Academy [4] 的“基本概念”页面。

1.2 主机信息

主机（*即* 笔记本电脑或台式计算机）是处理数据集准备、神经网络训练和神经网络编译所必需的。在创建此应用时，主机是运行 Ubuntu 18.04 LTS 的 64 位 x86 CPU。该机器包括 512GB SSD、16GB RAM 和 NVIDIA A2000 GPU，用于训练。

Python 3.6 是主机上的主要编程环境。有关如何设置具有必要依赖项的 Python3 虚拟环境的指导，请参阅 edgeai-modelmaker。

2 创建数据集

创建神经网络（又称“模型”）的第一步是创建/精选数据集。作为解决问题的数据驱动型方法，机器学习模型和深度学习模型的性能取决于训练这些模型时使用的数据。理想的做法是，使用针对最终任务的目的定制数据来训练模型。

COCO 或 ImageNet 等公共数据集为开发和评估深度学习模型提供了方便的途径。有许多公开可用的数据集；其中很多可以在 paperswithcode.com 等站点上找到 [5]。但是，根据许可证条款，并非所有公共数据集都可用；此外，它们的优质数据点也可能太少。同时，自定义数据集创建起来非常耗时。

在零售扫描仪应用中，可用且可许可的在线数据集的质量不足以按原样使用。许多图像都是众包的，没有很好地标记，因此经过训练的模型在验证数据子集以及在照明良好的付款台区域内的实际性能都很差。必须从头开始创建数据集。

2.1 收集图像

应在与在目标嵌入式处理器上实际使用模型类似的上下文中收集图像。例如，付款台可以使用一个具有明亮光照和中性色背景的朝下摄像头。空间可以（基本）没有其他对象，但可能显示用户的手、钱包、智能手机等。

用于收集图像的摄像头应具有良好的质量，但不需要具有摄影师所需的专业品质。深度学习过程中使用的大多数图像的缩放分辨率（< 100 万像素：例如，512x512）小于原始输入。使用最终产品将使用的同一摄像头可能会有好处，包括摄像头可能需要的相同自动增益、白平衡和其他调优设置，但这并非明确有必要。使用不同的摄像头或设置可以提高稍后必须更换摄像头时的稳健性。本文档中的零售扫描仪应用使用终端摄像头（一个 IMX219 图像传感器）和 1080p USB 摄像头进行图像采集。

在数据集中，目标是为机器学习模型提供与待解决问题相关的足够多*模式*实例。例如，香蕉应具备什么样的外观才能添加到顾客的订单中。训练样本中不应始终存在不相关的模式，例如香蕉总是在图像边缘附近或在白色背景下。为了提高模型的稳健性，如果特性可能发生变化，建议改变照明条件；对象方向、位置和角度；摄像头角度/高度以及对象自身版本（例如，不同成熟度的香蕉）。如果差异不够大，则训练数据*可能会过拟合*，这样神经网络在识别训练数据时极其有效，但无法泛化到新数据。数据集应反映终端系统实际将看到的图像。

较大数据集在训练深度神经网络方面比较小数据集更有效。大型*优质*数据集非常有效。但是，大型数据集的收集和标注很耗时。从较小的数据集开始，有助于诊断需要更多数据的问题类型。每类项目的实例数量少至 50 个的数据集足以满足首次尝试的需求。初始概念验证仅需 10 个。对于这些微型数据集而言，经过训练的模型已经在大型数据集（如 COCO 的 ImageNet）上进行过一次训练（使用 TI 工具进行训练的模型就是如此），这一点很重要。

食品检测数据集使用每个食品项的 100-200 个实例，以提高稳健性。此应用旨在演示器件功能。虽然该模型不需要有生产价值，但它确实需要足够可靠地执行，以便在随机环境中以合理的准确性进行演示。该食品数据集包括真实的假食品或包装的不易腐食品，使演示更易于在新的地方传输和重现。这样，对象的外观在训练后不会发生很大变化。图 2-1 是来自食品检测数据集的示例图像。

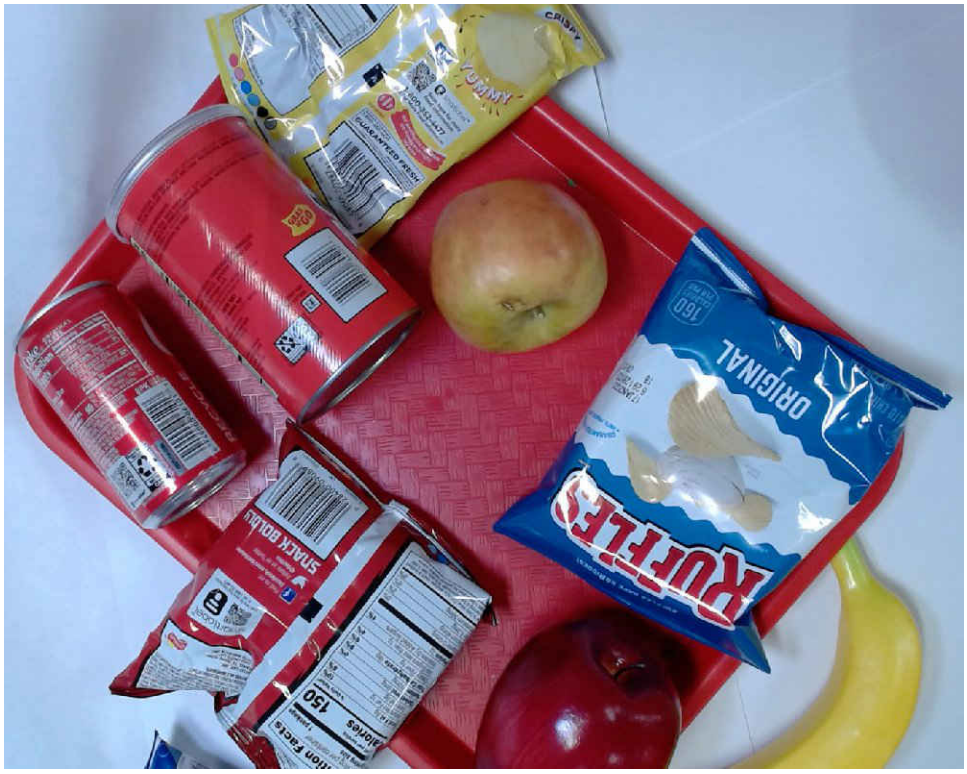

图 2-1. 来自食品识别数据集的图像

可以从此演示应用的 Github 存储库下载数据集 [2]。

2.2 标记图像

深度学习和神经网络是*受监控*的机器学习算法。这些算法要求数据具有关联的*标签*，用于告知训练，数据中有哪些相关信息。标签类型取决于要解决的问题的类型。对象检测模型需要坐标（通常为两个 2D 点，表示*边界*框）和对象类型；一个图像中可以有多个坐标。对于零售扫描仪应用，这是正确的选择，因为这种应用的部分目的是快速自动识别多个物体。

遗憾的是，标记通常是一项乏味的任务。如果做得不好，模型精度将会受到影响。有几种工具可简化此过程。TI 的 Edge AI Studio 是一款在线云工具，用于为 TI 处理器执行标记和大多数其他模型开发任务；但是，在零售结算应用的开发过程中，此工具不可用。离线/局部标签的替代工具是“label-studio”，标签界面如图 2-2 所示。此图显示了食品识别数据集中的一个图像。存在多个对象，并且这些对象上绘制有彩色框，以指示它们所属的类。

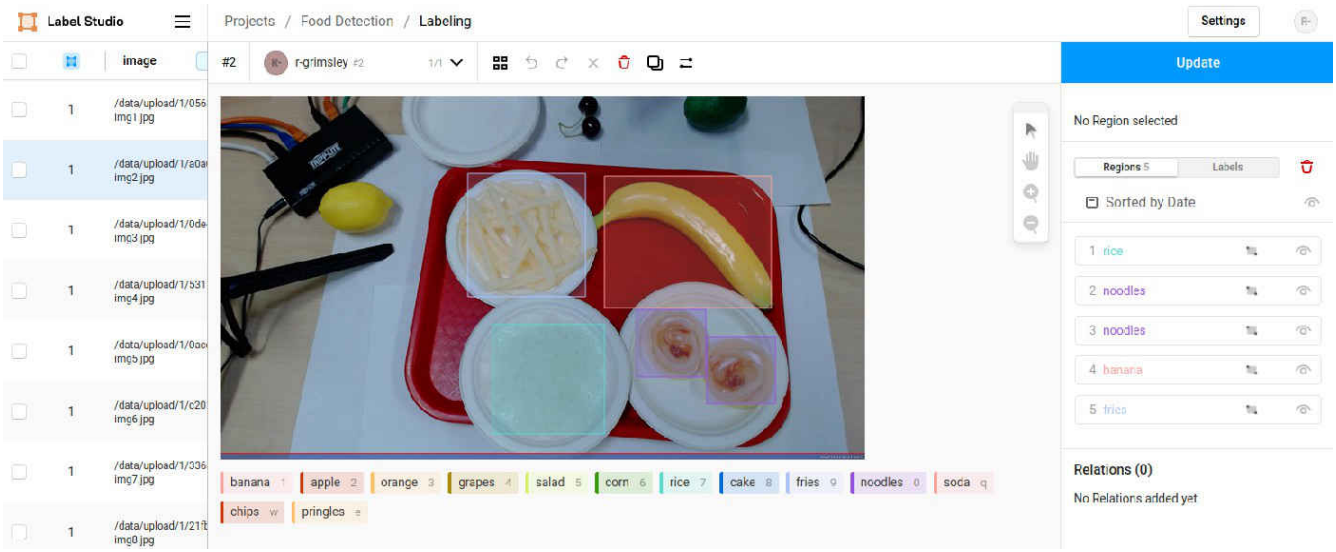

图 2-2. Label Studio 界面。（有意将其他对象保留在图像中并取消标记。允许干扰可提高稳健性和精度。）

标记完所有图像后，数据集可以多种格式之一导出。TI 的训练工具使用 COCO JSON 格式。对于此格式，输出是一个 ZIP 存档文件，其中包含一个图像文件夹和一个名为 result.json 的 JSON。

2.3 增强数据集（可选）

“增强数据集”是指在不增加数据标记负担的情况下，通过添加具有各种噪声和修改的数据副本来人为扩展。如果做得好，增强操作可提高模型的稳健性并避免过拟合。例如，当摄像头聚焦不正确时，添加模糊效果可以提高稳健性；向图像添加旋转有助于引入不属于原始数据集的对象定位。

如果正在为数据集创建*训练集和测试集分割*，则应在增强之前完成分割，以免污染测试集。有些增强的整体影响可能很小，因此测试和训练图像之间几乎完美匹配。这会不当地提高模型的计算精度。

对于食品识别模型，使用表 2-1 中所示的增强列表。

表 2-1. 增强列表

尺寸/方向变换	筛选器/局部效果	加法/乘法效应
透视变换	高斯模糊	高斯噪声
左右翻转	锐化	增加饱和度
上下翻转	动态模糊	更改色温
剪切	对比度	使色调和饱和度倍增
透视变换	JPEG 压缩	
旋转	自动对比度	
	锐化	

并非所有增强都应用于每个图像；对随机子集进行了应用。为每个原始图像创建了八个额外增强版本的图像。这些增强是根据实际情况下看到其影响的可能性来选择的。这并非所有可用和有用增强的详尽列表，但也不仅限于翻转、裁剪、调整大小、旋转、模糊和增加高斯噪声等标准增强。Python 中的 imgaug 库用于应用这些增强并在此过程中保持边界框，以避免附加标记。

鉴于此处允许的增强类型和数量，此数据集被视为“重度”增强。不建议对所有应用使用重度增强，尤其是在图像中的细微变化本身可能被视为图案的应用中，例如机器视觉应用中的缺陷检测。

此应用程序存储库中的 data_manipulation.py Python3 脚本包括用于修改图像、另存为文件以及将图像添加到描述数据标签/注释的 COCO JSON 文件中的源代码。它还包括用于重新处理数据和标签的其他函数，例如，将多个 label-studio 训练会话组合成一个大型数据集，以及将 JSON 标签文件转换为易于操作的中间格式。

3 选择模型

模型选择可以与数据集创建同时进行。要使 TI 的 C7xMMA 深度学习加速器发挥出色性能，必须在 C7xMMA 上支持网络中的每一层。不受支持的层仍然可以工作，但可能需要 CPU 资源才能运行这些层，这会降低性能。

TI Model Zoo 中的模型都包含受支持的层。TI 使用许多业界通用的先进架构。在某些情况下，这些架构经过细微修改，更加易于加速；这些模型被称为“lite”或“ti-lite”模型。请注意，在撰写本文时，AM62A 上不支持某些原本受支持的层（例如指数线性单元，即 ELU），因为 AM62A 最近刚刚推出。这意味着 AM62A 尚不支持 model-zoo 中的某些模型进行加速。

TI 的 Edge AI Cloud “模型分析器”是一款用于选择模型的有用工具。这提供了模型性能的视图，以便开发人员可以根据性能（例如，推理速度、内存使用情况）和准确性（在 COCO 等标准数据集上）等指标来选择模型或架构。模型的性能与训练模型时使用的数据集无关，但模型的准确性与训练模型时使用的数据集有关。在比较不同模型的准确性时，开发人员应确保仅比较使用同一数据集进行训练的模型之间的准确性。

对于为零售扫描仪演示而构建的食品识别模型，选择了 mobilenetv2SSD。TI 的 Model Zoo 中有一个经过预先训练的模型，可用作迁移学习的起点。在 TI 工具中，此初始模型的名称为“od-8020_onnxrt_coco_edgeai-mmdet_ssd_mobilenetv2_lite_512x512_20201214_model_onnx”。模型名称中包含大量信息：

- od：对象检测神经网络
- “8020”：用于区分模型的唯一编号，以便在使用多个模型时更快地查阅。
- onnxrt：ONNX 运行时是用于此模型的运行时/API
- coco：最初的版本使用 COCO 数据集进行训练，该数据集包含 80 多种日常对象，如人、汽车、香蕉 (!)、家用电器等。
- edgeai-mmdet：训练框架是 edgeai-mmdet，它是 TI 在 OpenMMLab 中的 mmdetection 开源工具的分支
- ssd：该网络的 *head* 或任务特定部分（用于对象检测）是 SSD，即单次检测。
- mobilenetv2：模型架构使用 MobilenetV2 在该模型中的初始层集中进行特征提取。用机器学习的行话来说，这是网络的*主干*。
- lite：该模型在原始架构的基础上进行了细微修改，对 TI 的 C7xMMA 架构更加友好
- 512x512：输入图像分辨率为 512x512
- 2020214：该模型最初使用数据集 (COCO) 进行训练的时间。并非所有型号名称都包含日期字符串

4 训练模型

选择模型架构并创建数据集后，下一步是在新数据集上重新训练模型。

TI 提供了一整套易于使用的模型训练和编译工具。任何训练框架都可用于训练深度学习模型，只要它仅包含器件加速器所支持的层即可。专家可能希望使用他们更熟悉的工具，例如使用 PyTorch 进行训练以及导出到 ONNX 进行编译。经验较少的用户可以使用 TI 的工具，如模型编写器和 edgeai-modelmaker。此处介绍的步骤使用 edgeai-modelmaker。使用 TI 工具的一个优势是，可以在训练结束时自动处理编译，这样可以完全跳过节 5。

Edgeai-modelmaker 一旦设置/安装完毕，即会使用单独的训练框架（如 edgeai-mmdetection）进行训练。这从经过预先训练的模型开始，并通过重置网络的最后一层和使用低学习速率，通过迁移学习进行微调。对于 Model Zoo 中支持的模型，这会下载经过预先训练的权重。请注意，使用此过程时，最后一层之前的层*不会*冻结，并且在训练期间会发生变化。

训练模型的步骤如下；请参考 modelmaker 自述文件获得最新的说明：

- 设置 modelmaker 存储库，其中包括设置其他训练框架和 TI 工具。它们被保存在同一父目录中。edgeai-modelmaker 存储库中有一个 setup_all.sh 脚本可以处理此问题，包括为其他训练框架进行设置。
  - 推荐使用 Python 3.6 的虚拟 Python 环境。
- 将训练样本/文件放入“edgeai-modelmaker/data/projects/PROJECT_NAME”。
  - 在其中一个示例配置 YAML 文件上运行 modelmaker 以查看这些目录的结构。它遵循 COCO 格式，具有一个“annotations”目录（包含一个 COCO 格式的“instances.json”文件）和一个图像目录。
  - 这些图像必须都位于同一目录（称为 images）中，不包含任何子目录。训练框架假定所有图像文件都位于同一“images”目录中。
- 创建一个指向项目数据的配置文件，选择模型、训练参数等。有关更多信息，请参阅 config_detect_foods.yaml 中的示例。
- 使用上面的配置文件作为第一个参数运行启动脚本“run_modelmaker.sh”。
  - 请参阅 modelmaker 的自述文件以了解如何设置器件目标，因为 AM62A 等处理器包含不同的 C7xMMA 深度学习加速器，因此该目标架构的编译工具与 TDA4VM、AM68A 等不同。它依赖于 TIDL_TOOLS_PATH 环境变量，如果未预定义该变量，SH 脚本将设置该变量。

如果训练机器上存在 GPU，强烈建议将其配置用于训练，因为它可以大幅提高速度。在配置中，重要的是确保 CUDA 版本与编译 Pytorch/torchvision 时使用的版本一致。撰写本文时随 modelmaker 一起提供的是 CUDA 11.3。正确设置这些驱动程序可能是一个棘手问题，因为 CUDA 版本与 NVIDIA 显示/图形驱动程序之间存在某种关系。

如果数据集相当小（少于 10000 张图像），可以从一个网络开始，该网络在大型通用数据集（如用于图像分类的 imagenet1k 或用于对象检测的 COCO）上经过预先训练。较大网络的参数数量增加，因此通常需要更多样本用于训练

对于包含大约 2500 张图像的食品识别数据集（在训练集和测试集分割并增强之后），在具有 12 核 CPU、16GB RAM 和 SSD 的 A2000 GPU (6GB VRAM) 上进行训练大约需要 1.5 小时，共进行 30 个 epoch 的训练。该数据集的平均精度均值（mAP；一个常用的对象检测精度指标）得分达到 0.68，非常高。这源于两个事实：

- 训练框架使用的验证集会自动包括增强后的文件，因此某些验证数据与部分训练样本非常相似，从而提高所报告的准确性。
- 环境/背景受到严格控制以提供一致性。该模型在有限的上下文中表现良好，但可能无法很好地推广到全新的设置，例如尝试在绿色或蓝色背景下识别对象。这可能重要，也可能不重要，具体取决于用例。[1]

未针对零售扫描仪模型执行最终测试集的完整评估。相反，在进入下一阶段之前进行了目视检查，确定模型的表现相当好。通过改变 epoch 数量、增强程度和增强种类，进行了多次训练迭代。

[1] 在零售扫描仪应用中，考虑到通常受控的环境，这看起来并不重要。但是，恶意分子可能会企图欺骗系统引入颜色怪异的背景（例如紫色纸张），从而使背景上的物品难以识别。新的背景可以替换为另一种形式的增强。

4.1 输入优化（可选）

训练 YAML 配置文件时有一个可选参数，用于与量化和编译相关的“input_optimization”。简而言之，它将两个预处理步骤折叠到模型的前几层中，以节省 CPU 周期和 DDR 带宽。此处提供了相关上下文以及模型量化的必要背景，这是编译模型的关键步骤。

模型几乎普遍进行了浮点模型权重和浮点值方面的训练。它们可以进行“量化感知”训练（QAT，即量化感知训练），但权重仍将是浮点格式；权重也可以在训练后进行量化（PTQ 或训练后量化）。PTQ 可将模型转换为定点格式，神经网络加速器通常针对该格式进行了优化。对于 TI 的架构，编译包含创建一个量化函数，将 float32 映射到整数（8 位或 16 位），这有时称为“校准”。此函数在整个网络中是通用的，因此，所有 float32 都处于类似的范围内非常重要，这样它们就能够一致地映射到定点，而不会将非常高或低的值“削波”到有限整数范围 [2]。

量化的一个常见步骤是在将浮点数映射到整数时限制削波，方法是将权重和值保持在 -1 到 1 或 0 到 1 的范围内。通过在训练期间使用正则化，在内层更容易做到这一点。对于模型输入，图像通常采用 8 位 RGB 格式，并通过减去均值并乘以比例因子来转换为 float32，这样进入模型的值就位于这个较小且一致的范围内。这是模型的必要预处理步骤，但在 CPU 上执行的效率不高。

“输入优化”是一种通过将预处理纳入模型中来提高预处理效率的策略。通过在输入端创建两个额外的层来在 C7xMMA 上执行此类按元素运算，在该模型中实现减法和乘法运算。这减少了 CPU 使用率和 DDR 使用率，因为可以按原样提供用于 RGB 像素的原始 8/16 位整数，而不是预处理的 32 位浮点值。

在编译过程中，该模型也可采用此 input_optimization 步骤。

5 编译模型

如上一节所述，如果使用 edgeai-modelmaker 进行训练，则可以自动处理编译。如果一个模型需要自行编译，或者用户希望更多地参与编译和校准，那么他们可以参考本节。Edgeai-tidl-tools 托管用于此目的的代码。模型编译必须在 x86 主机上完成。

edgeai-tidl-tools 存储库为模型编译提供了多个选项，例如 Jupyter 笔记本和 python3 脚本。该存储库中有用于编译自定义模型的说明。使用 Jupyter 笔记本的过程可以在 Edge AI Cloud 上完成，也可以使用自定义模型的工具完成。Jupyter [7] 笔记本是一种以模块化形式逐步显示和开发代码的理想工具，但 Jupyter 内核在执行长时间运行或内存密集型任务时可能不稳定。由于 Python3 脚本在运行时稳定性更高，因此被用于 mobilenetv2SSD 模型编译。

为了编译模型，一组图像将在模型的选择运行时（对于食品识别模型为 ONNX）传递，在此期间，TI 深度学习编译后端会分析中间值和格式。当此脚本运行时，“工件”以编译的二进制文件的形式生成，用于在 C7xMMA 上运行模型。这包括一个“io”文件（用于说明 CPU 如何向/从 C7xMMA 传递信息），以及一个对网络层本身进行编码的“net”文件。

在此过程中，一组图像用于*校准*模型，以实现从原生 32 位浮点到定点整数（8 或 16 位）的量化。这些工具会在执行期间分析模型中的中间值范围，并选择与该范围非常匹配的量化方案。在实践中，使用能更好地描述输入场景广度的较大图像集可以提高量化质量，但需要更长的时间。

edgeai-tidl-tools/examples/osrt_python/ 下的 Python 脚本可以处理模型编译的所有步骤，但与更具互动性的 Jupyter 笔记本相比，对预处理、配置等进展不太明确。要完成编译，需要对 onnxrt_ep.py 和 model_configs.py 文件进行一些更改：

- 有一个要编译的模型列表。对于食品识别模型，这包括“od-8020_onnxrt_coco_edgeai-mmdet_ssd_mobilenetv2_lite_512x512_20201214_model_onnx”。
- 该模型的配置保存在上面目录中 model_configs.py 的大 Python 字典内。键是上一个项目符号中的字符串。
  - “model_path”必须指向自定义数据集中经过训练的模型。
  - “meta_layers_names_list”必须指向描述模型架构的 PROTOTXT 文件路径。
  - 如果上述两个文件不存在，则会下载它们，前提是该文件是已受支持的模型架构，并可在提供的 URL 上下载。

[2] 如果混合网络在一些层上使用多个量化级别（如 8 位）、而在另一些层上使用 16 位，则将具有多个量化函数。尽管如此，在整个网络中保持相同的权重范围将减少削波量。

调用环境中的 TIDL_TOOLS_PATH 环境变量应指向目标处理器的 tidl_tools。默认情况下，它指向设置 edgeai-tidl-tools 时下载的工具，在此期间，应该已经设置了供器件编译的 SOC 变量，*例如* SOC=am62a。

完成后，编译的工件位于一个目录中，该目录的型号全名位于 edgeai-tidl-tools/model-artifacts 下。

在继续实际使用模型之前的最后一项（可选）：使类标签在模型输出和文本名称之间保持一致。这仅在使用 edgeai-gst-apps 或 tidlpostprocess GST 插件时才是必要的，因为它们会尝试在图像上为已识别的对象编写文本。当 modelmaker 尝试编译模型时，它会创建一个 dataset.yaml 文件来描述数据集，其中包括将输出类整数映射到文本标签。这是从保存标签的 COCO JSON 文件中检索的。param.yaml 文件还需要提供从模型输出到 dataset.yaml 中索引的映射（键 [metric][label_offset_pred]）。对于食品识别模型，它是一对一的，因为使用了所有接受过训练的类。dataset.yaml 由 TI 的训练框架自动生成，但 param.yaml 可能需要手动更新。

6 使用模型

下一步是实际使用模型。

在训练期间报告的准确性有助于确定模型的有效性，但根据实际输入进行可视化对于确信模型按预期运行至关重要。

针对目标器件上的新输入评估模型的快速方法是在 edgeai-gst-apps 中使用该模型。这是一项有价值的概念验证，用于在不编写新代码的情况下评估更符合实际的精度。将“compiled-artifacts”内的新目录复制到目标器件，并修改诸如 object_detection.yaml 之类的配置文件（如图 6-1 所示）以指向此模型目录。确保在配置文件底部的流程中使用模型。输入可以是 USB 摄像头等实时输入，也可以是图像文件的预制视频/目录。

```yaml
title: "Food Recognition"
log_level: 2
inputs:
    input0: # 720p USB camera, e.g. Logitech c270 or similar with that resolution capability
        source: /dev/video2
        format: jpeg
        width: 1280
        height: 720
        framerate: 30
    input1: #IMX219 with default settings. Run camera setup script under scripts
        source: /dev/video2
        width: 1920
        height: 1080
        format: rggb
        subdev-id: 2
        framerate: 30
    input2: #IMX219 w/ 1640x1232 RGGB10. Run camera setup script in retail-shopping
        source: /dev/video2
        width: 1640
        height: 1232
        format: rggb10
        subdev-id: 2
        framerate: 30

models:
    model0:
        model_path: /home/root/edgeai-gst-apps-retail-checkout/retail-shopping/food-detection-model-mobv2ssd/
        viz_threshold: 0.6

outputs:
    output0:
        sink: kmssink
        width: 1920
        height: 1080
        overlay-performance: True


flows:
    flow0: [input0,model0,output0,[320,180,1280,720]]
```

图 6-1. edgeai-gst-apps 配置文件示例

连接的显示器/保存的视频文件看起来如图 6-2 所示（也可能会覆盖性能信息，具体取决于输出的配置）：

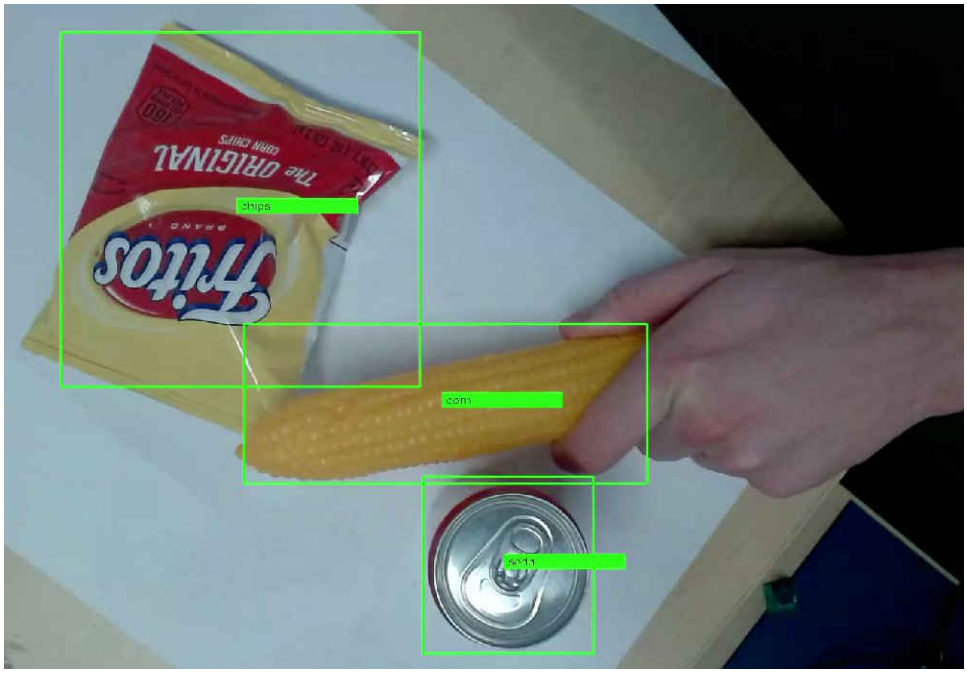

图 6-2. 对新训练的模型使用 edgeai-gst-apps 时的 Gstreamer 显示

以这种方式运行模型的另一个好处是，将一个整体 gstreamer 字符串打印到终端，这是应用程序开发的有益起点。

7 构建最终应用

一旦确认模型可正常工作，即可围绕其构建端到端应用。

端到端应用涉及获取摄像头的实时输入、预处理、运行模型以及使用适合实际用例的输出执行*某种操作*。在零售扫描仪用例中，上述*某种操作*是使用已识别的对象为顾客填充和显示订单。

要在 Linux 中构建数据处理流水线，建议使用 gstreamer (GST)。借助 edgeai-gst-apps 中使用的 GST 插件，除应用程序代码外的所有内容都可以完全通过 GST 流水线来完成。GST 流水线允许以并行流式方式进行不同的处理阶段，与单线程程序相比，这提高了整体性能。TI 提供的插件可与硬件加速器进行交互，以进行图像处理和深度学习。对于最终应用，GST 可以使用 *appsink* 和 *appsrc* 插件分别将接口暴露在应用程序代码中和应用程序代码之外。

在零售扫描仪应用中（参阅 gst_configs.py），将检索对 appsink 的引用，并将其用于“提取样本”，该样本包含一个表示数据块的原始字节数组。这可能是神经网络的图像、音频块或输出张量。需要事先了解数据的结构，例如维度和像素格式。对于图像等典型数据（在 GST 术语中，为 raw/x-video），描述 GST 插件输入和输出的“cap”提供了有关分辨率和像素格式的信息。

在首次设计零售扫描仪演示应用程序时，直接的方式是采用 RGB 格式以完整分辨率从相机中拉出输入帧。Python 应用程序代码处理了预处理、推理、后处理，并将收条式图像添加到要显示的最终帧中。该应用程序借用自主 apps_python 代码，但 display.py 中的额外图像处理速度缓慢，并且延迟变化非常大。对于刷新所有元件和测试功能，这没有问题，但对于交互式应用程序而言，在大约 5fps 至 6fps 的帧速率下，延迟长达 3 秒是无法接受的。mobilenetv2SSD 模型在大于 60fps 时识别食物的速度非常快，但 Arm CPU 上的应用程序代码太慢，无法满足模型推理或 30fps 摄像头的需求。

7.1 使用 TI 的 Gstreamer 插件优化应用

流应用可从流水线中获得很大好处，因为应用的每个组件都在单独的进程中运行。当硬件加速器和多个 CPU 内核可用以允许并发时，这会很有效。TI 的 gstreamer 插件经过优化，可以利用 AM6xA 处理器的异构架构，并通过零缓冲区副本实现，以减少内存开销。

OpTIFlow 插件（*即* tiovxdlpreproc、tidlinferer 和 tidlpostproc）有助于有效地形成 AI 应用流水线，因为它们允许并行进行预处理、推理和后处理。使用 OpTIFlow 插件，唯一需要在专用应用程序代码中运行的功能就是使用深度学习输出。在零售扫描仪应用中，这意味着创建一个列出顾客订单的小图像。图 7-1 阐明了这一流水线。有关确切的 gstreamer 流水线和详细说明，请参阅 HOW_ITS_MADE.md 自述文件。

在 AM62A（Processor SDK 8.6，入门套件 EVM 的 E2 修订版）上使用此流水线时，帧速率为 15fps 至 18fps。主要瓶颈是 Python 应用程序代码，因为这包括额外的内存副本和使用 OpenCV 进行的多个文本绘制函数调用。深度学习加速器的负载为 25% - 30%，四通道 A53 CPU 的负载为 30% - 40%（所有内核的平均值），32 位 3200 MT/s LPDDR4 的负载为 20%。

优化流水线以更好地减轻应用程序不同任务的负担，就 FPS 和延迟降低的量级而言，性能提升了 200% 以上。可以采取更多措施来进行优化，例如在 C/C++ 中开发、创建更高效的函数以便一次性将所有文本写入帧（或完全删除应用程序的那一部分），甚至将应用程序代码转换为 GST 插件。

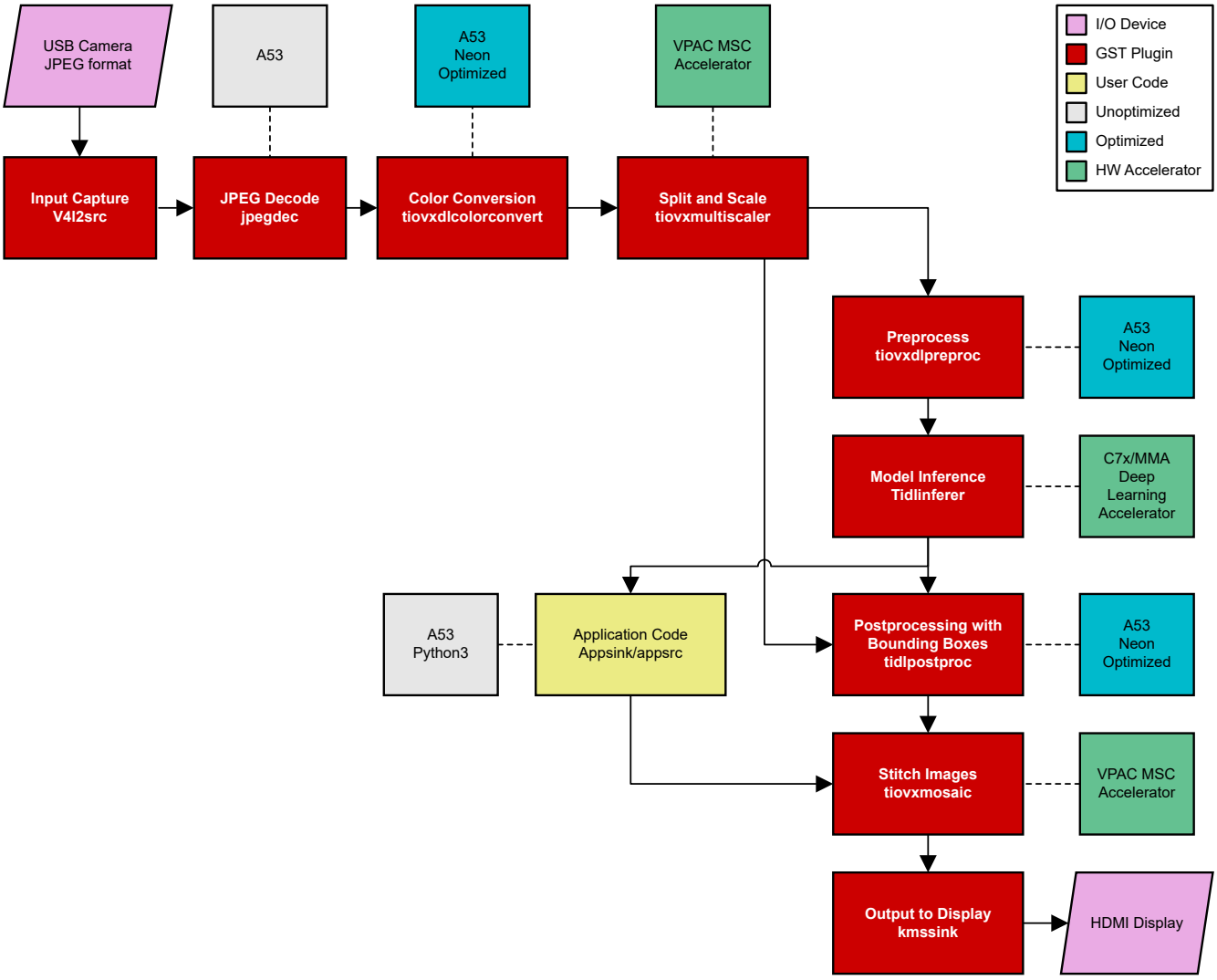

图 7-1. 零售扫描仪应用的 gstreamer 流水线方框图

7.2 使用原始 MIPI-CSI2 摄像头

TI 的边缘 AI 处理器（例如 AM62A、TDA4VM、AM68A）包括 ISP 和其他硬件加速器，用于缩放广角镜头（镜头失真校正，LDC）的图像（多标量 MSC）和去扭曲失真。ISP、MSC 和 LDC 都是 TI 在视觉处理器加速器 (VPAC) 中设计的 IP。本节将介绍如何为此应用使用 IMX219 图像传感器。

备注

零售扫描仪演示还适用于 720 和 1080p 的 USB 摄像头。运行时，请将 -c 标签与用于 720p 或 1080p USB 输入的 usb-720p 或 usb-1080p 一起使用。C270 和 C920 Logitech USB 摄像头均已经过验证可正常运行。本文档到目前为止，GST 流水线版本使用 USB 输入来进行简化。

需要 ISP（图像信号处理器）来获取图像传感器的原始输出并将其处理为 RGB 等更典型的格式。原始输出传感器的价格更低，与图像传感器中内置的 ISP 相比，外部 ISP（包括 TI 的集成式 ISP）更易于调优。可根据光照条件、透镜效应、自动曝光等对 ISP 进行调优。TI 提供了用于集成式 ISP 的调优工具和相关支持文档。

IMX219 是一款开箱即用的低成本传感器，在 15fps 时可以高达 800 万像素采集数据，在 30fps 时可以高达 200 万像素采集数据（60fps 时可以 200 万像素采集数据，但需要 SDK 文档中链接的 Linux 补丁）。

支持传感器包括多个元件，所有这三个元件均可用于 IMX219：

- 一个驱动程序，理想情况下，该驱动程序会向上流式传输到主线 LINX 内核并进行主动维护
- ISP 调优
- 用于评估的硬件模块

零售扫描仪演示在 30fps 下需要 200 万像素输入，并需要至少 70° 的视野。使用此特定传感器会给此用例带来几个难题；这些难题及其解决方案如下所述：

- 该驱动程序对不同分辨率和格式的支持有限
  - 典型的 16:9 宽高比 1920x1080（200 万像素）分辨率没有完整的视野 – 这是对完整 800 万像素帧的裁剪，因此会放大的不自然。结果远低于 70° 的 FOV 要求。
  - 1640x1232 可实现宽高比为 4:3 的全 FOV（宽度和高度为半分辨率；仍为 200 万像素）。但是，上游驱动程序仅在 10 位像素值 (RGGB10) 下正常工作。此模式用于零售扫描仪应用，并修改演示库中的 setup_cameras.sh 脚本以使用 Linux“media-ctl”命令行工具设置此配置。
  - 在 200 万像素分辨率下不支持 60fps，它以 30fps 运行。这是由于上游驱动程序造成的。TI 的边缘 AI 处理器能够为该传感器提供完整的 60fps 速率，但它需要一个 Linux 内核补丁、重新构建驱动程序模块并在目标上安装。
- 不同的分辨率可能需要更改 ISP 文件（通常位于 /opt/imaging/imx219 下）– 这需要重新生成“DCC”文件。本文档不介绍该过程。
  - 8.6 Edge AI SDK 现在包含 ISP 配置文件，可提供上游 IMX219 驱动程序所能实现的所有分辨率和位深度。

8 总结

本文档介绍了在 TI AM6xA 处理器上为视觉任务构建边缘 AI 应用的分步过程，并使用 AM62A 和零售扫描仪演示提供了该过程的示例。遵循此开发流程可以极大地加快新设计的速度。本文档详细介绍了用于训练、编译和运行模型以及在 Linux 中创建 gstreamer 流水线的 TI 处理器特定步骤，该流水线利用硬件加速器进行图像处理，而无需用户手动对这些加速器进行编程。提供的参考文献旨在为开发人员提供工具、代码和进一步指导，以便开发利用复杂处理器而不增加设计复杂性的应用。

9 参考文献

1. 德州仪器 (TI)，“边缘 AI 市场 - 零售结算”，2023 年 4 月。[在线]。网址：https://github.com/TexasInstruments/edgeai-gst-apps-retail-checkout。
2. 德州仪器 (TI)，“edgeai-gst-apps-retail-checkout”，[在线]。网址：https://github.com/TexasInstruments/edgeai-gst-apps-retail-checkout/blob/main/retail-shopping/README.md。[2023 年 4 月 18 日部分]。
3. 德州仪器 (TI)，“TI 边缘 AI - 模型开发”，[在线]。网址：https://github.com/TexasInstruments/edgeai/blob/master/readme_models.md。
4. 德州仪器 (TI)，“Edge AI Academy”，[在线]。网址：https://dev.ti.com/tirex/explore/node?node=A__ABufwRIVEGy83u-FZ.kg5Q__EDGEAI-ACADEMY__ZKnFr2N__LATEST。
5. “包含代码的论文 - 图像数据集”，[在线]。网址：https://paperswithcode.com/datasets?mod=images。
6. 德州仪器 (TI)，“Edge AI Studio”，[在线]。网址：https://dev.ti.com/edgeaistudio/。
7. “Jupyter 笔记本”，[在线]。网址：https://jupyter.org/。

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265