arXiv:2304.01555v1 [cs.CV] 4 Apr 2023

# 边缘人工智能设备上的实时驾驶员监测系统

Jyothi Hariharan、Rahul Rama Varior 和 Sunil Karunakaran

Ignitarium Technology Solutions Private Limited，印度班加罗尔{
jyothi.hariharan, rahul.varior, sunil.karunakaran}@ignitarium.com

**摘要**。 由于驾驶员注意力不集中导致的道路交通事故日益增多，自动驾驶员监测系统（DMS）逐渐得到广泛认可。本报告介绍了一种可在基于硬件加速器的边缘设备上运行的实时 DMS 系统。该系统由一台用于录制驾驶员影像的红外摄像头和一台用于处理数据的边缘设备组成。为使深度学习模型能够充分利用硬件加速器，在边缘设备上成功运行，我们对模型进行了手术。最终的 DMS 系统在 TI-TDA4VM 边缘设备上达到每秒 63 帧（FPS）。

**关键词：** DMS、驾驶员监测系统、TDA4VM、TIDL、汽车、车内监测

## 1 引言

由于驾驶员注意力不集中，道路交通事故正在增加。全球范围内，人们越来越接受在驾驶员分心、无法专注驾驶时向其发出警觉提示的自动化系统。例如，欧洲新车评估计划（NCAP）标准和欧盟委员会（EC）的法规要求从2023年起采用驾驶员监控系统（DMS）技术。美国国家运输安全委员会（NTSB）建议在半自动驾驶汽车中使用 DMS。近年来，采用人工智能（AI）技术构建的 DMS 已被证明是一项有效的安全功能，并在汽车行业受到广泛关注。本文介绍了我们自主开展的研究与开发工作，旨在构建一套基于摄像头、可在边缘设备上运行的实时驾驶员监控系统。

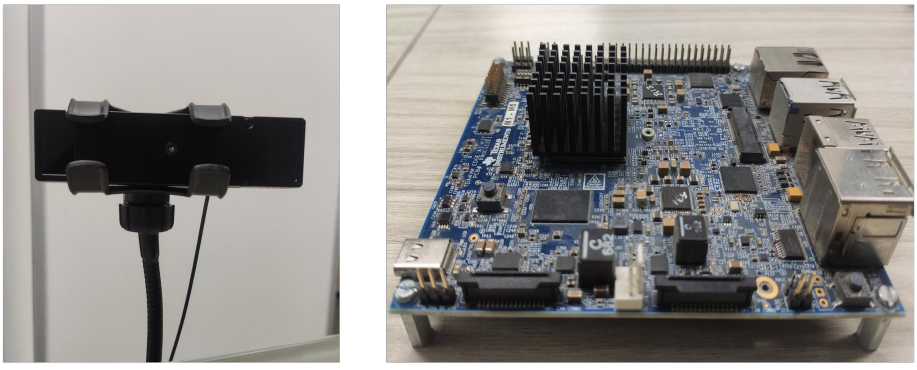
汽车中的安全功能应为驾驶员留出充足的响应时间，以便采取必要行动。因此，必须实时处理数据，并在驾驶员分心或困倦时向其发出警觉。虽然存在多种拓扑结构（包括仅使用中央处理器、中央处理器 +图形处理器、中央处理器 + 加速器等），但所选的特定计算架构通常会直接影响所选边缘设备的成本和功耗。在低成本边缘设备上部署基于深度学习（DL）的人工智能系统可能颇具挑战，因为这类设备存在内存限制且处理能力较低。此外，由于种种原因，将深度学习模型移植到边缘设备上运行，以充分利用专用的人工智能硬件加速器，也可能颇具挑战。

供应商的软件开发工具包（SDK）可能不支持所有必需的深度学习算子和参数范围。在许多情况下，SDK 虽然提供了支持，但其实现可能并未针对底层硬件进行充分优化。

本文介绍了我们在研发基于摄像头的实时 DMS 系统方面所做的工作，该系统运行于基于硬件加速器的边缘设备上。首先，我们介绍 DMS 的高层架构以及系统各个组件的详细信息。下一节将说明如何成功将模型移植到示例边缘设备硬件加速器（即德州仪器 TDA4VM）上所面临的挑战及解决方案。最后，我们介绍成功将模型移植到硬件加速器上之后，在推理时间和每秒帧数（FPS）方面取得的性能提升。

## 2 DMS 高层架构

DMS由Leopard Imaging的红外摄像头组成，用于采集视频数据。采集到的数据由运行在边缘人工智能设备上的人工智能技术栈进行处理。为便于演示，处理结果会显示在标准显示设备上。请注意，显示设备仅用于可视化，并非最终系统的组成部分。图1展示了DMS的组件。后续章节将详细介绍各个组件。



(a) (b)

**图 1.** DMS 的组件：(a) LI-USB30-AR0144IVEC-GMSL2-055H 红外摄像头；(b) TI TDA4VM 边缘人工智能开发套件

### 2.1 DMS 的组成部分

**红外摄像头** 为采集驾驶员影像，Leopard Imaging（LI）的红外摄像头 LI-USB30-AR0144IVEC-GMSL2-055H 安装在驾驶舱内的仪表板上。LI 红外摄像头有两大优势。首先，摄像头模组配备红外照明，可实现低照度成像和夜视。模组中的红外滤光片还能避免环境光照造成的干扰。其次，红外摄像头能够透过某些太阳镜“看见”眼睛，从而准确捕捉瞳孔位置。有关摄像头规格的更多信息，感兴趣的读者可参阅产品数据手册。

**人工智能技术栈**人工智能技术栈由边缘设备和感知系统或软件应用组成。

**边缘设备**：所用的边缘设备是德州仪器的 TDA4VM。TDA4VM 配备双核 64 位 Arm® Cortex®-A72 微处理器子系统、每秒最高可执行 80 GFLOPS 的 C7x 浮点向量数字信号处理器，以及每秒最高可执行 8 TOPS（8 位）的深度学习矩阵乘法加速器（MMA）。专用硬件加速器 MMA 可处理大部分深度学习工作负载，并针对 8 位精度（int8）和 16 位（int16）精度整数计算进行了优化。感兴趣的读者可在 TDA4VM 数据手册中查阅更多产品信息。我们使用的软件开发套件版本为 8.02.00。

**感知系统**：感知系统由一个基于深度学习的人脸检测模型和一个人脸关键点模型组成。视频输入中的各帧会传入人脸检测算法，以检测驾驶员的脸部。检测到的脸部会被裁剪，并作为人脸关键点模型的输入。人脸关键点算法会估计脸部轮廓、虹膜关键点、眼部关键点、眉毛关键点、上下唇关键点、鼻梁和鼻底等面部关键点。这些面部关键点用于估计每只眼睛的头部姿态、闭眼时长和视线方向，并检测驾驶员是否在打哈欠。根据驾驶员的打哈欠检测结果、闭眼时长、视线方向和头部姿态，可推断驾驶员的状态，即驾驶员是否分心、困倦或警觉。感知系统的流水线阶段如图 2 所示。

**人脸检测深度学习模型**：在人脸检测方面，我们使用 TI 边缘人工智能视觉模型库中的 Mediapipe 人脸检测模型 BlazeFace [2] 和 YOLOX- tiny 模型 [5]进行了实验。

**人脸关键点深度学习模型**：在人脸关键点估计方面，我们使用 Google Mediapipe 人脸网格模型 [6] 和实用人脸关键点检测器（PFLD） [4] 模型进行了实验。下一节将介绍模型移植和优化工作。

## 3 TDA4VM 模型移植

TDA4VM 配备了 MMA，这是一种用于处理深度学习工作负载的硬件加速器。然而，并非所有深度

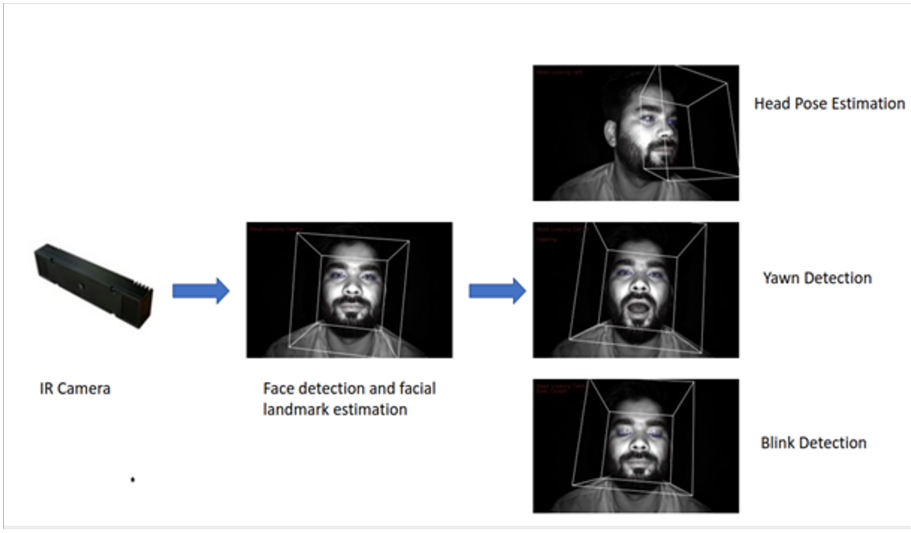



**图 2.** 感知系统的流水线阶段

学习算子或算子配置都支持在 MMA 上执行。在这种情况下，不受支持的算子通常会在 ARM CPU 核心上执行，从而降低深度学习模型的整体推理速度。为解决这一问题，需要调整预训练模型，使其功能不受影响，并完全由加速器执行。以下部分介绍了我们为移植模型所做的工作，以便将其完全卸载到 MMA 上。

加载深度学习模型以在 MMA 上运行时，模型的输入数据将从 CPU 内存复制到加速器内存。如果深度学习模型中的某个运算符不受设备硬件加速器库支持，该运算将由 ARM CPU 执行。执行相应层所需的输入数据必须从加速器内存复制到 CPU 内存。层执行完成后，输出需要复制回硬件加速器内存，以便处理其余层。CPU 与加速器之间的数据传输开销很大，会导致推理速度不理想。

为了充分利用硬件加速器，模型的所有层都必须卸载到 MMA。因此，模型冻结图中不支持的层/算子必须替换/移除（模型图手术或模型手术），并换成 MMA 支持且功能准确的层 / 算子。为了执行模型手术，我们开发了一个模型图转换库（或工具包）。该模型图转换库有助于从冻结的 TFLite 或 ONNX 图创建可训练模型，并可在可训练模型上执行手术。使用此库的优势包括：

1. **不同平台之间的转换**– 例如，TFLite 模型可以转换为 ONNX 模型，反之亦然。2. **从冻结图创建可训练模型**– TFLite 或 ONNX 冻结图可以分别转换为可训练的 Keras [3] 模型或 PyTorch [7]模型。此转换的好处是，如果我们希望在更新层之后训练或微调模型，就可以使用相应的 PyTorch 或 Keras 模型。3. **提取或修改层**或算子，比直接编辑冻结图更加方便。4. **训练后模型量化**– TensorFlow [1] 要求输入模型采用 Keras 格式。步骤 2 中所述的过程，即从冻结图创建可训练模型（本例中为 Keras 模型），有助于使用 TensorFlow 的训练后量化流程。

下一节将介绍我们为将上述模型移植到边缘设备并在不改变其功能的前提下实现最高运行速度而进行的模型图手术示例。

### 3.1 模型图手术

我们针对 MediaPipe BlazeFace 人脸检测模型 [2], MediaPipe 人脸网格模型 [6] 和 PFLD 人脸关键点模型 [4]进行了模型手术实验

**MediaPipe 模型中的反量化算子和 FP16 运算** MediaPipe 模型以 16 位浮点数参数发布。人脸检测模型和人脸关键点模型包含**反量化** 算子，可将模型的 16 位浮点数参数转换为 32 位浮点数，以便使用 32 位浮点数执行运算。目前，TDA4VM SDK 不支持反量化算子。通过模型图转换，我们能够将模型参数的数据类型转换为 32 位浮点数，并移除反量化算子。

**按通道填充算子** MediaPipe 人脸检测和人脸关键点 TFLite 模型包含按通道填充算子。tf.pad 算子支持使用该算子的填充属性添加常量值。填充层的配置参数用于指定算子输入数据各个轴上的填充量。在 MediaPipe 模型中，该算子配置为沿通道方向用零值（常量值）填充输入。不过，TDA4VM SDK 不支持所指定的填充操作。因此，我们尝试了以下替代算子，以在不影响层功能的情况下替换该算子。

1. **使用零值进行拼接**：由于使用零值进行逐通道填充会沿通道轴添加零，因此 Padding 运算符可以

替换为拼接操作。需要填充的零的数量由填充层的填充属性确定

2. **Conv2D 操作**：在 Conv2D 操作中使用零填充单位矩阵作为滤波器，即可实现与零填充操作等效的结果。

上述两种实现都可作为替代方案，用于底层硬件不支持逐通道填充的情况。然而，我们发现，若采用与零张量拼接的方式，则所需尺寸的零张量（根据填充配置确定）必须作为模型的另一个输入，这会增加从 CPU 内存到 MMA 内存的数据传输量，并降低模型的整体推理速度。因此，我们用 Conv2D 操作替换了填充算子。下面给出一个简单示例，说明如何使用矩阵乘法算子在数学上实现填充算子，并且这一方法也可扩展到 Conv2D 操作。

例如，如果该操作需要在输出中添加 4 个零，则可以通过将一个单位矩阵（维度与输入相同）和一个零矩阵（按要求填充，并调整维度以适配单位矩阵）拼接起来构造滤波器，从而实现相同效果。示例如下。沿宽度方向填充四个零：

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

$$Input \qquad Filter \qquad Output$$

### 3.2 量化

模型成功移植到 MMA 上运行后，使用 TIDL 编译工具对模型进行量化，并以 8 位整数精度运行推理。C7x DSP/MMA 针对整数计算（8 位、16 位）进行了优化，对 float32 预训练模型进行量化有助于显著缩短推理时间，同时降低带宽和存储需求。

## 4 结果

我们使用表中提到的模型进行了实验，并报告了模型图手术前后的帧每秒（FPS）结果。所有模型均已量化为 int8。

表 1：模型手术前后的每秒帧数（FPS）结果

| Sl No | 模型名称 | FPS before 模型手术 | FPS after 模型手术 | 相对每秒帧数提升 (%) | 手术执行步骤 |
|---|---|---|---|---|---|
| 1 | Mediapipe face detection 检测-模型 [2] | 121.55 | 259.84 | 113.77 % | 填充 with 放置二维卷积 |
| 2 | YOLOX-tiny [5] | 170 | - | - | No 模型手术已执行 was |
| 3 | Mediapipe face landmark 模型 [6] | 163.60 | 500.97 | 206.22% | 填充 with 放置二维卷积 |
| 4 | PFLD face 关键点模型 [4] | 688.59 | - | - | No 模型手术 |

### 4.1 DMS 系统级性能

表 1 的结果表明，人脸检测和人脸关键点任务的最佳模型分别是 Mediapipe 人脸检测模型和 PFLD 人脸关键点模型。然而，由于边界框预测 $[0-1]$ 与人脸关键点 [0–图像宽度/高度] 的输出张量值动态范围差异极大，因此观察到了严重的量化误差。我们提出将输出张量拆分为不同分支，分别处理边界框和人脸关键点。不过，这项工作留待未来开展，因为它超出了模型手术实验的范围。因此，我们使用了 YOLOX-tiny 模型 [5]，该模型已完成优化并移植到 TDA4VM 设备上，运行速度为 170 帧/秒。对于人脸关键点模型，我们选择了 PFLD [4] 模型，其运行速度已经快于 Mediapipe 模型。

采用 YOLOX-tiny 模型和 PFLD 人脸关键点模型后，DMS 达到了 63 帧/秒，超过了 30 帧/秒的目标速度。DMS 的详细计时信息如下表 2 所示

尽管此版本的 DMS 未使用经过模型手术的模型，但我们仍然观察到，移植后的模型仍能让系统实现实时性能。此外，修复量化问题后，BlazeFace 模型的人脸检测速度可以快于 YOLOX-tiny。我们计划将这些工作纳入边缘人工智能硬件优化的持续路线图。

表 2：DMS 中每个感知层的处理时间。**帧/秒 = 1000/15**。**8 = 63.29**

| 处理步骤 | 执行时间（毫秒） |
|---|---|
| 输入图像预处理 | 0.2 |
| 人脸检测推理 | 6.1 |
| 人脸检测后处理 | 0.4 |
| 关键点图像预处理 | 1.8 |
| 关键点模型推理 | 3.2 |
| 关键点结果后处理 | 0.3 |
| 眨眼/打哈欠/头部姿态计算 + 可视化 time | 3.8 |
| 总时间 | 15.8 |

## 5 结论

近年来，政府法规推动自动驾驶员监测系统受到广泛关注。由于成本优势，在边缘设备上实施 DMS 更具吸引力。TI TDA4VM 等功能强大的边缘人工智能设备配备了专用硬件加速器，有助于加快计算密集型深度学习模型运算。然而，要在这些启用加速器的边缘设备上实现高性能 DMS，仍面临诸多挑战。为了让深度学习模型完全在硬件加速器上运行，必须替换或移除不支持的层/算子，同时确保模型功能不受影响。我们观察到，模型完全在 MMA 上运行后，推理延迟有所降低。当深度学习模型完全在 MMA 上运行时，我们的 DMS 可达到 63 帧/秒，超过了目标速度。

# 参考文献

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irv- ing, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Man´e, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Vi´egas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow：异构系统上的大规模机器学习（2015年），https://www.tensorflow.org/，软件可从 tensorflow.org 获取 2. Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., Grundmann, M.: BlazeFace：在移动端GPU上实现亚毫秒级神经网络人脸检测。CoRR**abs/1907**05047 **305047** 2019 1907 2015.( )，http://arxiv.org/abs/ 。. Chollet, F.，等：Keras。https://keras.io（ ）4. Guo, X., Li, S., Zhang, J., Ma, J., Ma, L., Liu, W., Ling, H.: PFLD：实用的人脸关键点检测器。CoRR **abs/1902.10859**（2019），http://arxiv.org/abs/1902.10859 5. Instruments, T.: edgeai-modelzoo。https://github.com/TexasInstruments/edgeai-modelzoo/blob/master/models/vision/detection/readme.md# 人脸检测与edgeai-mmdetection（2021年）6. Kartynnik, Y., Ablavatski, A., Grishchenko, I., Grundmann, M.: 在移动端GPU上通过单目视频实时生成面部表面几何。CoRR **abs/1907.06724**（2019 06724 7 1907），http://arxiv.org/abs/ 。. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch：一种命令式风格的高性能深度学习库。载于：Wallach, H., Larochelle, H., Beygelzimer, A., d'Alch´e Buc, F., Fox, E., Garnett, R.（编），《神经信息处理系统进展第32卷》，第8024–8035页。Curran Associates, Inc.（2019年），http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library .pdf