arXiv:2304.06327v1 [cs.AR] 13 Apr 2023

目标检测近似局限性的汽车案例研究

Mart´ı Caro[†,‡], Hamid Tabani[†], Jaume Abella[†], Francesc Moll[†,‡], Enric Morancho[‡], Ramon Canal[†,‡], Josep Altet[‡], Antonio Calomarde[‡], Francisco J. Cazorla[†], Antonio Rubio[‡], Pau Fontova[†,‡] 和 Jordi Fornt[†,‡]

[†]巴塞罗那超级计算中心（BSC）
[‡]加泰罗尼亚理工大学（UPC）

摘要

基于深度学习的基于摄像头的目标检测（CBOD）通常仅根据帧中的真实物体来评估精度。然而，这种简单的评估忽略了许多不重要的物体体积小、距离远或位于背景中这一事实，因此，在自动驾驶等领域，这些物体的漏检比近处、较大且位于前景中的物体影响更小。此外，偶发性漏检无关紧要，因为检测结果的置信度通常会在连续帧之间取平均值，而且检测设备（例如摄像头、激光雷达）通常具有冗余，从而提供容错能力。

本文利用基于摄像头的目标检测（CBOD）流程固有的容错能力，并通过一项汽车案例研究，评估 CBOD 在多大程度上能够容忍来自多个来源的近似，例如低精度算术、近似算术单元，甚至是由低电压运行等因素引起的随机故障。我们表明，即使同时考虑这三个近似域，这些近似来源对精度的影响仍在基线的 1% 以内。因此，可以利用多个近似来源，为汽车中的 CBOD 构建高效加速器。

1 引言

如今，基于人工智能的系统正广泛应用于各个领域的多种场景。基于深度学习的系统尤其常用于目标检测；为在有限时间内实现更高的预测精度和置信度而设计的硬件与软件（例如以较高帧率处理摄像头拍摄的图像）也在不断改进。

基于深度学习构建的基于摄像头的目标检测（基于摄像头的目标检测）已部署于多个领域的各种应用中，其中一些应用具有安全要求，因此对可容忍的错误程度有所限制。例如，

自动驾驶（AD）中的 CBOD 案例，其中准确的目标检测对于确保车辆安全驶向目的地至关重要。例如，*You Only Look Once*(YOLO) [1,2] 是一款屡获殊荣、广受欢迎且高效的实时 CBOD 系统，已用于 NVIDIA Drive [3] 和百度 Apollo [4]，等工业级自动驾驶系统。百度 Apollo 是一款广受欢迎的工业级自动驾驶软件框架，已用于多种原型车辆（包括自动驾驶卡车和自动驾驶出租车）。

YOLO 实现了一种基于卷积神经网络的计算密集型函数。它基于 IEEE 754 32 位浮点运算开发和训练，并已被证明具有很高的有效性和效率。

对于每一帧（图像），YOLO 都会提供检测到的对象列表及其各自的置信度。默认情况下，只有检测置信度高于 50% 的对象才会被 YOLO 视为真实对象。在采用 YOLO 架构的框架（如 Apollo）中，也会考虑并相应实现类似的阈值，甚至可能采用完全相同的阈值。

总体而言，CBOD，尤其是 YOLO，由于建立在神经网络之上 [5]，并提供带有关联置信度的结果，因此具有随机性。这种随机性常被利用来采用低精度运算（例如使用 16 位浮点而非 32位浮点），从而在保持精度的同时降低计算成本，并且在某些情况下仅以可忽略的精度损失为代价。

虽然可以在图像粒度上评估目标检测的有效性，即判断是否检测到了物体，并识别出假阳性和假阴性，但对于汽车目标检测，如果决策是像 YOLO 那样通过对多帧结果取平均得出的，那么假阳性/假阴性的语义影响在单张图像的粒度上并不重要。相反，系统会综合摄像头拍摄的多个连续帧中的检测结果，因为只有当某个物体在多帧中（不一定是严格连续的帧）都以足够高的置信度被检测到时，才会将其视为真实物体。例如，某个物体只在一帧中被检测到，而在相邻帧中没有出现，就可以直接忽略。同样，在一段连续帧序列中，如果某个物体在其他帧中都被检测到，只在其中一帧里未能以足够高的置信度检测到，那么这种情况极不可能改变最终结果，因此该物体仍会被视为真实物体。

自动驾驶框架会利用多个帧以及多个传感器（摄像头、激光雷达和雷达）的检测结果来增强鲁棒性。因此，单个帧中的假阳性或假阴性本身不会造成语义错误。这为容错能力带来了额外的维度，可利用这一点进一步降低硬件复杂度。

本文针对自动驾驶系统中的基于摄像头的目标检测，研究精度与复杂度之间的权衡。具体而言，我们首次在一项汽车案例研究中表明，可以同时利用多个近似域，而对精度的影响微乎其微：

A）降低精度算术（例如 16 位浮点），与现有文献保持一致。B）近似算术，例如近似加法和乘法，其中某些输入操作数的计算结果可能存在系统性误差。C）例如因低电压运行导致的随机故障，这可能使某些运算偶尔产生任意错误的结果。

因此，这项工作表明，未来可以通过采用低电压、低精度的近似算术，设计功耗更低、频率更高的特定于 AD 的 CBOD 加速器，而不会影响整个目标检测过程的语义。

本文其余部分安排如下：第2节介绍相关工作。第3节介绍目标检测系统的背景，以及提高深度学习模型效率的不同方法所达到的先进水平。第4节介绍不同的近似域及其对基于摄像头的目标检测结果的预期影响。第6节介绍并评估该案例研究。最后，第7节总结本文。

2 相关工作

许多研究从不同角度致力于实现高效的目标检测，包括减少获取和处理的数据量，以及降低获取和处理这些数据的成本；这两类解决方案彼此正交且互为补充。前一类研究包括：通过实时视频分割来压缩帧内及帧间的时空冗余 [6], ；设计用于低成本显著目标检测的新型 CNN 架构 [7], ；以及实现 CNN 模型压缩 [8]。这些研究最终减少了从内存中获取的数据量，并降低了对所获取数据执行的运算次数。后一类研究则包括使用替代数据表示、近似算术和低电压运行的相关工作，接下来将对此进行讨论。

在神经网络降低精度领域，相关研究不胜枚举。Tang 等人 [9] 提出了 MLPAT，这是一个面向机器学习加速器的功耗、面积和时序模型框架。该研究探索了自动驾驶加速器中精度与架构之间的权衡设计空间，并评估了在 Google 开发的 TPU-v1 [10], ASIC 上采用 16 位浮点、16 位脑浮点、16 位整数和 8 位整数时的精度、功耗和面积。Wu 等人 [11] 提出了一种用于基于 FPGA 的加速、无需重新训练的定制低精度（8 位）浮点量化方法，并在 VGG16 [12] 和 tiny-YOLO [13]上研究了该方法。在我们的工作中，我们使用完整版本的 YOLO v3 [2] 和标准浮点格式。Mellempudi 等人 [14]提出对权重、激活、误差和梯度采用 8 位浮点表示，不过该研究重点在于训练深度神经网络。

多项研究探讨了其他可能的数字格式，例如定点格式，并在训练和推理过程中 [15], [16], [17], [18], 得出结论：在卷积神经网络的应用场景中，采用替代性的数字格式表示是可行的，且不会对精度造成显著影响。然而，在实践中，目前使用标准浮点数字表示法仍然更为可行，因为硬件已针对其进行了高度优化，而其他格式可能需要进一步优化。

已有多项研究探讨了将近似算术应用于卷积神经网络。Hammad 等人 [19] 研究了使用动态可配置的近似乘法器，对采用 ImageNetV2 数据集 [23]的 VGG19 [20], Xception [21], 和 DenseNet201 [22] 网络进行 CNN推理。Ibrahim 等人 [24] 探讨了通过采用近似乘法器和加法器，将近似计算用于机器学习。

更多相关研究可参见 Manikandan [25]的综合综述。据我们所知，目前尚无相关研究专门探讨自动驾驶系统中基于摄像头的目标检测的近似算术；而其他研究近似算术的工作采用的网络复杂度较低。在我们的工作中，我们以基于简化全加器（SFA）的加法器 [26]和欠设计乘法器（UDM） [27] 的一种配置作为近似算术单元的示例。其他研究则探讨了不同的近似单元。

故障对卷积神经网络的影响已从不同角度进行了分析。其他一些工作研究了应用于卷积神经网络的激进低电压运行的影响。Salami等人 [28]开展了一项实验研究，分析了在真实现场可编程门阵列中实现的、具备激进降低的电源电压能力的卷积神经网络加速器的功耗-性能-精度特性，尽管该研究并未聚焦于自动驾驶领域。Chandramoorthy [29] 也对低电压运行进行了评估，但研究对象是字符识别，这与基于摄像头的目标检测是不同的应用。

A. Ruospo等人 [30] 刻画了软件注入的永久性故障对 LeNet-5 卷积神经网络的卷积神经网络权重的影响，该网络使用 MNIST数据集 [31] 进行手写数字分类 [32] 。作者针对 32位浮点算术以及不同的定点算术配置（32、18、16、10和 6位）进行了刻画。同样，F.Libano 等人 [33] 研究了辐射诱发的硬件故障对不同低精度模型的影响。具体而言，该工作分析了极简卷积神经网络（7 层）的 32 位和 16 位浮点版本，以及8 位整数版本；这些版本也使用 MNIST 数据集 [32] 进行评估，以完成手写数字分类，这是一项比 CBOD 更简单的任务。

其他研究 [34,35]着眼于在自动驾驶领域利用时空冗余进行故障检测。作者利用连续帧之间具有高度视觉相似性这一事实来检测故障。作者通过直接比较连续帧的结果来检测硬件故障。相比之下，我们采用 Darknet 框架提供的默认实现，对最近三帧的检测结果置信度取平均，以减少一些偶发性漏检，但没有加入任何专门的故障检测或纠正机制。在我们的案例中，用于缓解模型不准确性的同一机制也被用于实现容错能力。

总的来说，我们的论文首次在一项实际案例研究中，对逐步应用于自动驾驶系统的不同近似领域进行了全面评估，并考虑了这些系统凭借时间和空间冗余所具备的容错能力。

3 背景

本节介绍基于摄像头的目标检测流程、YOLO，以及算术精度与近似计算的相关背景。

3.1 基于摄像头的目标检测

自动驾驶框架中的感知过程（例如 Apollo）以互补且冗余的传感器为基础，实现对场景的理解。这包括不同类型的传感器，例如摄像头、激光雷达和雷达，也包括同类型的传感器，这些传感器可能会以冗余方式覆盖周围的某些区域（例如具有重叠视角的摄像头）。这

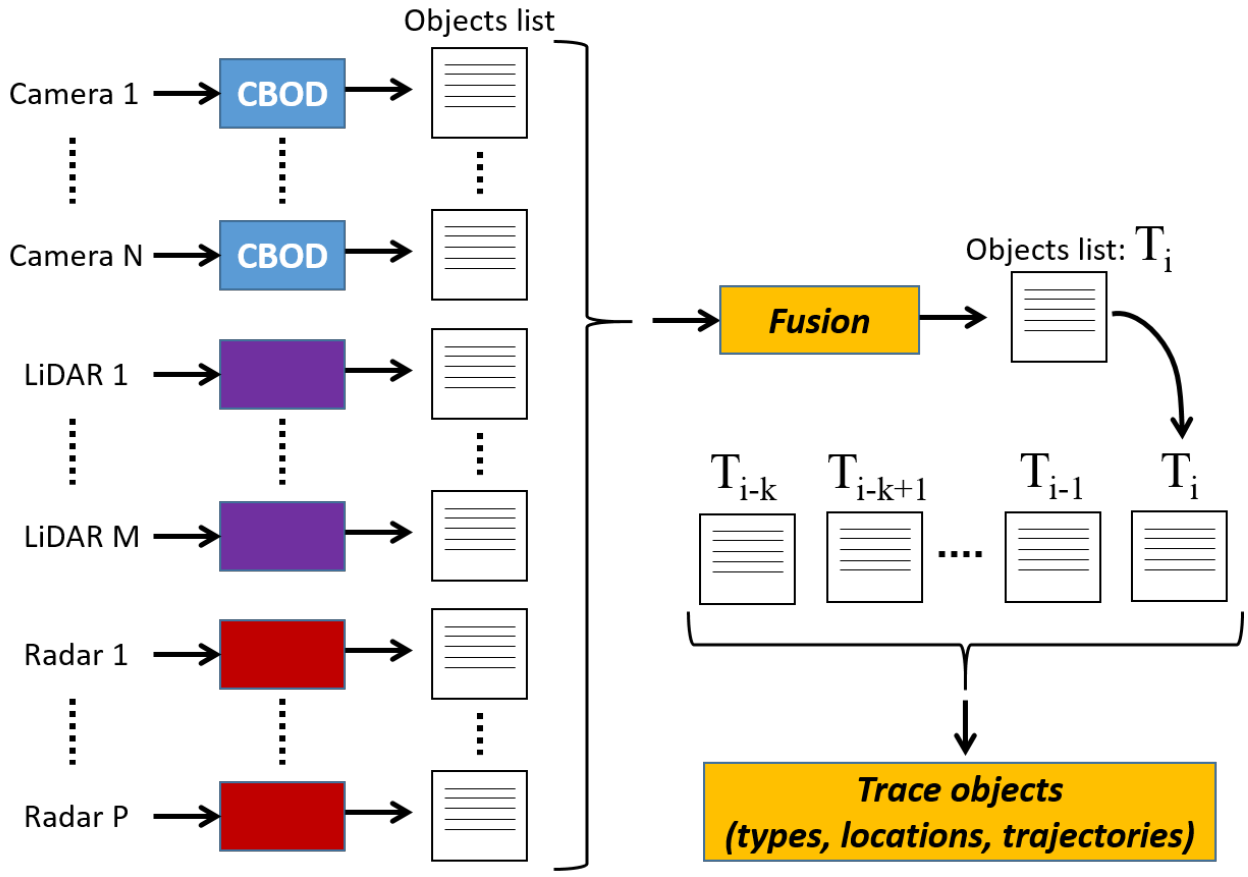


图1：自动驾驶中的感知过程。

如图1所示。每个传感器处理过程都会定期生成新的目标列表（例如，对于帧率为 25 帧每秒 的基于摄像头的目标检测（CBOD），每 40 毫秒生成一次），然后将这些列表融合为一个目标列表。

融合过程需要对汽车周围的物体提供一致的识别结果，这意味着要匹配由不同传感器识别出的物体，同时考虑它们的类型、大小、位置以及检测置信度。该过程本质上具有容错性，因为如果一个传感器由于对某个真实物体的检测置信度较低而将其丢弃，但覆盖同一区域的其他传感器以较高置信度检测到该物体，则会在融合过程中考虑该物体。

融合过程会定期生成新的目标列表（例如，在 T_{i-2}、T_{i-1}、T_i 时）。随后需要处理这类目标列表序列，以便随时间跟踪目标，从而准确预测其轨迹，使自动驾驶系统能够做出安全驾驶决策。对连续目标列表的处理本质上同样具有容错性，因为会利用一段时间内各个列表中的检测结果，因此零星的假阳性/假阴性不会对自动驾驶目标检测过程产生语义影响。举例来说，如果在车辆的前视图中检测到一辆汽车（例如，在 T_{i-k}、T_{i-k+1}、……、T_{i-1} 时的目标列表中），但它突然未出现在某个目标列表中（例如 T_i），只要这辆汽车未被检测到的情况发生，轨迹预测模块仍能够将其纳入考虑。

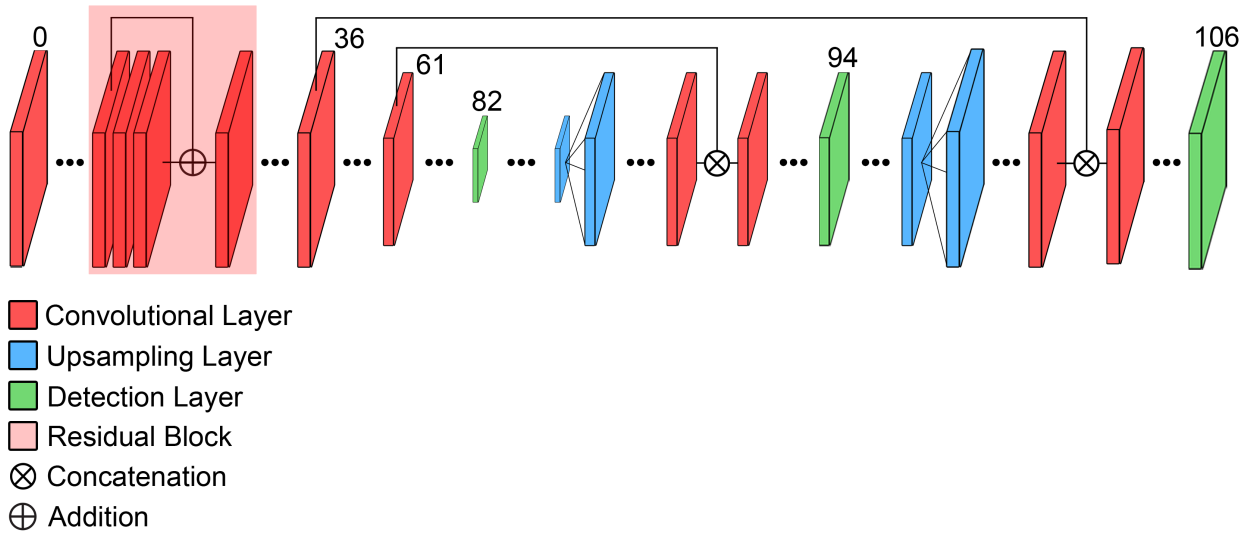


图2：YOLO v3 架构概览。

零星地。类似地，如果在仅一个目标列表（例如 T_i）中错误地检测到一辆汽车位于车辆前方，那么轨迹预测过程将能够检测出这一错误，因为不可能在过短的时间间隔内（例如 40 毫秒）有这样一辆汽车驶到车辆前方：它在 T_{i-3}、 T_{i-2} 和 T_{i-1} 中缺失，而在 T_i 中却位于某个特定位置（例如靠近该汽车）。

总体而言，自动驾驶框架的感知过程具有容错能力：单个传感器独立产生的误报或漏报，或多个传感器共同产生的误报或漏报（例如融合过程生成的单个目标列表中的误报或漏报），都不会对驾驶决策产生任何语义影响。

在这项工作中，我们分析了自动驾驶车辆单个摄像头拍摄的若干视频，并使用 YOLO v3 对其进行处理，将其作为基于摄像头的目标检测的代表。因此，由于缺乏适用于此目的的数据（例如，驾驶过程中由多个摄像头拍摄的数据），我们不利用冗余传感器带来的容错能力。不过，我们展示了该过程对错误的容忍能力，这种能力源于 YOLO 检测过程内在的随机性，以及连续视频帧间目标检测的容错能力。

3.2 YOLO 实时目标检测系统

YOLO 是一种先进水平的实时基于摄像头的目标检测系统，被用作多个工业自动驾驶软件框架中基于摄像头的目标检测的主要组件 [3,4,36,37]。YOLO 使用 32 位浮点数据，因此，尽管我们的发现同样适用于其他数值表示（例如整数），但本文其余部分均以这种数值表示作为基线。

图2展示了 YOLO v3 架构概览。YOLO v3 由 106 层组成，包括卷积层、上采样层、快捷连接层、路由层和检测层。残差块由若干卷积层和用于特征提取的跳跃连接组成。该架构的主要独特之处在于，它会在三个不同尺度上进行目标检测。具体而言，检测过程在第 82、94 和 106 层执行，并且每次检测过程后都会

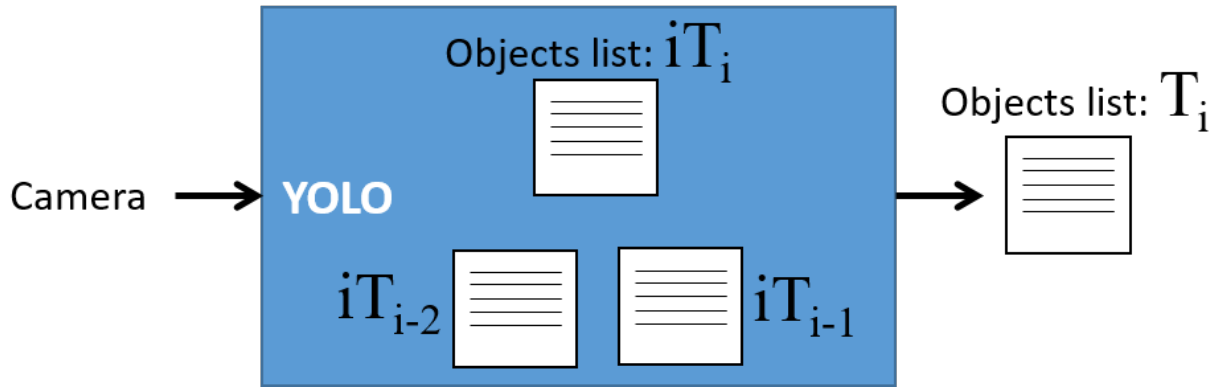


图 3：YOLO 案例研究评估结果。

将输入图像放大 2 倍，从而更好地检测不同大小的目标。图像被划分为网格，每个单元负责预测三个边界框（即界定目标区域的矩形）。对于这三个边界框中的每一个，都会计算其坐标、目标置信度分数（即该单元包含目标的概率）以及类别分数（即针对每个类别的分数，用以表示目标属于该类别的概率）。该模型定义了 80 个类别，包括动物、食物、车辆和行人等。许多边界框可能不包含目标、与包含同一目标的其他边界框重叠，或者包含目标和／或属于特定类别的概率很低。因此，系统会应用过滤过程，保留置信度最高且高于设定阈值（默认设为 50%）的预测，并丢弃其余预测。如图3所示，这可以针对内部目标列表（iT_i）（即独立分析该帧）进行，也可以在更高的抽象层级上针对生成的输出列表（T_i）进行；该列表汇总了最近三帧的检测结果，以确定是否应将某项检测纳入考虑（即平均置信度高于 50% 阈值）。

3.3 算术中的精度与近似

低精度算术。 低精度算术是一种广泛采用的模型压缩方法 [10, 11, 15, 38]。例如，使用 16 位浮点或 16 位整数来减少数值表示所用的位数，可以立即缩小模型尺寸。将高精度算术中的数值映射到低精度算术中的过程称为量化。例如，某个无法用 16 位浮点完全精确表示的 32 位浮点数，可以通过将其映射为另一个非常接近的 16 位浮点数来进行量化 。低精度算术可通过降低硬件复杂度，节省芯片面积、功耗和时序开销。

近似算术。近似算术 [19, 24, 39] 是指采用简化的算术单元（如加法器和乘法器）进行实现，其结果与位数相近的全精度算术相比精度较低。近似

设计的优势在于显著降低了硬件复杂度，从而节省面积、功耗和时序开销，代价通常是精度略有降低。

上述方法可以结合使用，以进一步压缩模型并提升能效，但可能会降低精度。

4 近似域及其影响

本节将分析 YOLO 基于摄像头的目标检测，尤其是基于深度学习的目标检测通常如何执行目标检测并轻松容忍误差；我们还将回顾本文所考虑的三个近似域，以及它们如何影响目标检测结果。

4.1 近似对 YOLO 的影响

YOLO v3 使用 $100+$层实现目标检测，其中计算密集度最高的是 53 个卷积层，它们的大部分时间都用于矩阵乘法，而矩阵乘法是大多数深度学习框架的核心 [40]。推理通过依次处理多个矩阵来完成。因此，前面层中出现的任何不精确或计算错误都会传播到后续层。

从推理过程的计算特性来看，一个矩阵中某个单元格的内容会用于计算另一个矩阵中的多个单元格，而后者又会以类似方式进行处理。因此，尽管某个单元格结果中的误差或不精确性会传播到网络后续层中越来越多的单元格，但由于权重的作用，这种传播的强度会逐渐减弱，因为误差只会部分传播到每个单独的单元格。总体而言，一个单元格中的局部误差会在后续层的多个单元格中变成一系列小得多的误差。此外，这些误差还可能部分相互抵消。例如，一个略微高估的值（例如 0.5，而非 0.48）与一个略微低估的值（例如 0.4，而非 0.43）相加，可能会减轻不精确性（例如 $0.5 + 0.4 = 0.9$ 而非 $0.48 + 0.43 = 0.91$）。在其他领域也观察到了类似的补偿效应，例如电路的关键路径延迟：路径中的逻辑门数量越多，由统计补偿导致的变化越小 [41]。

从语义上看，这种误差传播会导致多个被检测物体的置信度和位置出现一些轻微变化，而不是造成影响单个或极少数物体的较大偏差。使用较低精度算术、近似算术，甚至出现偶发错误值时，情况也是如此。因此，可以预期，只有置信度接近接受阈值（例如 50%）的检测结果，其置信度才可能发生足够大的变化，从而导致语义输出不同。例如，如果置信度最多变化 $\pm 5\%$，那么只有原本置信度处于 $[45\%, 55\%]$ 范围内的检测结果才可能被分类为不同类别。

如果我们进一步考虑，融合过程利用了来自不同冗余或部分冗余传感器[1]，然后这类关于物体检测的信息-

[1]如前所述，本文的应用案例并未对使用多个冗余传感器所带来的容错能力进行定量分析。

此外，由于近似也会被应用于周期性检测流程（例如每40 ms），我们可以预期，处于临界置信度范围内的对象会在不同传感器之间发生变化，并且对于同一传感器，随着时间推移也会发生变化。此外，我们还可以预期，不准确性并不总是以相同的方向影响给定对象（例如，在多个传感器之间以及随着时间的反复变化中，将其从略高于阈值移动到低于阈值，或反之）。

总而言之，基于深度学习的目标检测过程具有随机性，因此结合估计的置信度值和阈值，以及传感器和时间冗余，本质上具备容错性，从而适合采用以牺牲部分精度为代价来节省功耗和设计复杂度的技术。

4.2 目标检测中的近似域

我们的近似目标是算术单元，因为我们假设它们通常占据脉动阵列的大部分面积和功耗。此外，我们基于这样的假设：算术单元的延迟决定了加速器的工作频率，或者至少对其有很大影响。

低精度算术。降低精度（例如使用 16 位操作数代替 32 位操作数）有诸多好处，例如存储数据以及用于处理这些数据的算术单元所需的面积更小，算术单元处理较小操作数时的延迟更低，计算功耗也更低。另一方面，降低精度能够编码的数值更少。因此，如果待存储的值无法用特定表示方式和精度下可用的位数精确表示，就会改用一个可表示的近似值。通常，精度越高，能够精确表示的数值越多；将无法表示的值舍入为相近的可表示值时引入的误差也越小。

以 YOLO 为例，与许多深度学习模型类似，其模型参数采用浮点运算表示。YOLO 默认使用符合 IEEE 754 标准的 32 位浮点数值（也称为 float），其中 1 位表示符号（s），8 位表示指数（e），23 位表示尾数（m）。而标准的 16 位浮点数值（也称为半精度浮点数）则用 1 位表示 s，5 位表示 e，10 位表示 m。除非正规数和其他特殊情况外，所表示的数值形式如下：

$$half_float(s,e,m) = (-1)^s \cdot 2^{e-15} \cdot (1.m_2)$$

例如，半精度浮点数 0011010101010100_2 对应以下实数：

$$\begin{aligned} half_float(0,01101,0101010100) = \\ (-1)^0 \cdot 2^{13-15} \cdot 1.0101010100_2 = 0.333007813 \end{aligned}$$

请注意，对于 float，所表示的数值形式为 $float(s,\ e,\ m) = (-1)^s \cdot 2^{e-127} \cdot (1.m_2)$。

一般而言，用 b 位表示数据，意味着最多可以表示 2^b 个值。因此，减小 b 会使可表示的值减少。这会产生较大

还是较小的影响，完全取决于所表示的值。例如，如果它们的指数通常处于 $[-14, 15]$ 范围内，那么半精度浮点数的 e 位就足够了，所有 m 位都可以用于表示有效数字，而不是非规格化值和超出范围的值，因此精度损失可限制在 0.05% 以下（即 $_{1\ 2048}$）。对于 YOLO，我们注意到大多数值都处于这个范围内。此外，超出该范围的值通常会与指数更大的值相加，因此在一定程度上会减小它们对精度的影响。

总体而言，如果没有过度降低精度（例如仅使用 8 位浮点表示），低精度算术预计只会对 YOLO 目标检测产生有限影响。

近似算术。近似算术有大量算术单元设计，例如近似加法器和乘法器。由于指数对所表示数值的影响呈指数级，因此其对误差的容忍度低得多，如 $[30, 42, 43]$, 所示；而操作数的尾数则可以进行运算，尤其是其低位部分能够容忍误差，这可能会产生类似于低精度算术的效果。

近似算术背后的原理是，逻辑电路中很大一部分用于为少量数值的计算提供精度。因此，可以移除或精简这部分逻辑，从而显著降低成本，同时将对结果的影响限制在某些输入上；而且在这些情况下，只对最低有效位进行近似，也能将影响降至较低水平。

正如在降低精度的情况下所讨论的，YOLO 使用的数值在大多数情况下，其尾数的高位都具有重要意义。因此，近似带来的影响非常有限。与降低精度的情况类似，近似算术造成的不准确性会影响低位，从而减轻这些不准确性带来的相对影响。此外，与降低精度的情况一样，近似算术导致的误差也可能相互抵消一部分。

总体而言，与降低精度的算术类似，预计近似算术对 YOLO 目标检测的影响有限。

另一方面，近似算术可以降低功耗并缩短关键路径长度（即最低可实现的周期时间）。例如，对于我们在评估中考察的近似加法器，作者展示了功耗降低 28%、延迟降低 50% [26]（更多细节见第6节）。对于乘法器，功耗降低幅度最高可达 31% [27]。

激进低电压运行。动态能量是组合逻辑中能耗的主要来源，例如用于目标检测等计算密集型工作负载的算术单元。众所周知，动态能量与供电电压的平方成正比（V_{DD}），因此，降低 V_{DD}是此类单元最有效的节能方法之一，尽管较低的 V_{DD} 电压运行可能会对延迟产生负面影响（ [44]）。

电路通常会采用一定的裕量进行供电和定时，以确保能够在周期结束前完成其功能。然而，给定一对$< V_{DD}, T_{cycle} >$，其中 T_{cycle} 表示周期时间，并且能够确保正确运行，可以在保持 T_{cycle} 不变的同时降低 V_{DD} ，从而消耗更低的能量，但代价是会出现一些偶发的时序错误，进而导致错误输出。此类错误可能发生在某些特定输入触发电路中的特定延迟路径时，这些路径会受到负-

由于工艺变化，在某些特定温度范围内，或在这些因素的特定组合下，可能会受到影响。另一种更简单的方法是使用功耗更低、但速度更慢的晶体管，而不改变 V_{DD} （例如，在综合电路中面向略低的工作频率设计这些部分，同时在运行期间保持原有频率），这样既能节省功耗，又能保持 V_{DD} 不变。晶体管速度较慢会导致路径变慢，在特定运行条件和输入数据转换下可能会产生一些错误；错误的影响和功耗节省完全取决于具体实现。此类时序错误的影响通常难以预测，因为在多种角落条件下，通常存在许多延迟接近T_{cycle}的关键路径。

总体而言，大幅降低 V_{DD} 能显著节省动态能量，但代价是偶尔会出现一些错误，对算术单元的输出结果产生任意影响。不过，只要这些错误足够罕见，它们对目标检测过程的整体影响在语义层面预计微乎其微。

5 评估框架

在本节中，我们介绍评估框架，讨论用于评估的数据集、所评估的近似设置，以及用于评估模型精度的精度指标。

5.1 数据集

为了评估不同近似域对基于摄像头的目标检测流程的影响，我们考察了默认的 YOLO v 3 [2] 版本，它使用一组预训练权重，并在 Darknet 框架 [45], 中实现，用于处理一组带有标注目标的图像——COCO 数据集 [46] ——以及若干具有代表性的驾驶视频录像 [47–52]。

标注数据集由一组准确标注了标签的图像组成。在目标检测中，会为每张图像中的每个物体标注一个边界框及其对应的类别。这些信息称为真实标注，之后会用于与基于摄像头的目标检测流程的输出进行比较。本案例研究使用 COCO 数据集的测试子集，因为该数据集包含数千张高质量图像。我们可以预期，使用该数据集会得到相对准确的结果，因为我们采用的是使用 COCO 训练子集训练的 YOLO 预训练版本。COCO 数据集包含常见物体处于实际场景中的图像。不过，由于该数据集具有广泛的通用性，其中包含许多与驾驶场景完全无关的图像。因此，我们筛除了不含行人或车辆的图像，以减少无关图像的数量。不过，即使图像中包含车辆，也不意味着这些图像一定属于驾驶场景。

为了补充实验分析，我们研究并纳入了一组由采集车辆专门为开发基于摄像头的目标检测而录制的未标注视频。由于使用软件实现的算术单元执行目标检测的成本很高（在我们的案例中，每种配置处理每帧需要 45 分钟），因此我们只处理了每段视频中的部分帧（每段视频 ≈ 30 秒）。

我们考虑过使用若干自动驾驶标注数据集，例如 CityPer- sons [53], KITTI [54] 和 ApolloScape [55], 等（实际最多考虑了 12 个数据集）。然而，据我们所知，目前没有可公开获取的数据集能够以足够的精度评估预训练的 YOLO v3 模型。由于以下至少一个原因，我们舍弃了这些数据集：（i）无法公开获取；（ii）对于我们的案例研究而言代表性不足，因为其中没有自动驾驶场景下最相关的目标类别（即至少包括车辆和行人）；（iii）缺少 2D 标注，而预训练的 YOLO v3 模型会生成 2D 边界框目标检测结果，因此必须具备此类标注；（iv）采用了与预训练 YOLO v3 模型相近的标注准则。该模型使用 COCO 数据集进行训练，采用相近的标注准则可确保结果具有意义。

5.2 近似设置

我们已将SoftFloat库 [56] 集成到Darknet 框架中，以在软件层面实现浮点数，用其软件仿真替代原生浮点运算，从而允许我们使用任意表示形式和算术运算。我们评估了四种不同的场景：IEEE 754 32 位浮点完全精确版本（YOLO32full）、IEEE75416位浮点完全精确版本（YOLO16full）、使用近似乘法器和加法器的16位浮点版本（YOLO16approx），以及注入一些故障以模拟激进低电压运行导致的错误输出的 YOLO16approx（YOLO16appfault）。

如前所述，YOLO16approx 对应于 YOLO16full，但尾数部分使用了近似乘法器和加法器，并在 SoftFloat 库的基础上实现，以模拟其行为。具体而言，YOLO16approx 实现加法运算时，不会将加法低半部分产生的进位传递到高半部分 [26]。如 [26], 所示，这项优化显著缩短了关键延迟路径的延迟，因此在给定的 V_{DD} 和 T_{cycle} 功耗预算内，可以使用功耗更低的逻辑门，同时仍能保证最高有效位相加时的完整精度，从而将近似带来的影响降至最低。对于近似乘法器，我们采用了 [27], 中的设计，该设计提供了一种构建 2x2 近似乘法器的方法，也提供了使用多个 2x2 小型乘法器模块构建更大乘法器的方法。按照此方法，给定两个操作数（A 和 B），分别将其拆分为高半部分（AH 和 BH）和低半部分（AL 和 BL），我们实现了一个近似乘法器，其中 ALxBL、AHxBL 和 ALxBH 模块均为近似模块，而 AHxBH 模块则是精确模块，因为该模块对乘法结果的影响更大。虽然还可以考虑其他近似算术单元，但如本节后文所示，所考虑的单元效果良好，因此足以说明，采用近似算术是自动驾驶系统中基于摄像头的目标检测的一条可行途径。

最后，YOLO16appfault 实现了 YOLO16approx，但在 SoftFloat 库内部的软件层面，模拟低电压运行导致的故障影响。我们通过以每个结果 p_{faulty}的故障注入概率随机翻转浮点乘法或加法结果中的一位，来建模与低电压相关的故障，这足以以任意方式破坏结果。特定的 p_{faulty} 数值如何与特定的

V_{DD} 值取决于技术，超出了本工作的范围。只翻转尾数中的位。鉴于指数和符号所占位数较少，我们假设处理它们的逻辑电路延迟较低（例如，以扇出 4、FO4、逻辑门来衡量），并且能够轻松容忍一定程度的延迟增加，在低电压运行下仍产生正确结果，同时不影响周期时间。相反，我们假设尾数位处于延迟关键路径上，因此，低电压运行导致处理尾数位的逻辑电路延迟增加，可能会产生错误输出。因此，可以在一定程度上降低 V_{DD}，而不影响指数位和符号位，并且只在尾数位中出现故障。此外，后文将展示，尾数位相对容错，因此可以用尾数中的一些故障换取功耗节省。我们评估了一些 p_{faulty} 值，以衡量不同 V_{DD} 值（或在尾数运算中采用低功耗晶体管的激进程度）与故障率之间各种权衡的敏感性；故障率通过提高 p_{faulty} 而产生，直到对结果产生明显但可接受的影响。请注意，处理每一帧都涉及运算数量级为 10^{10} 16的浮点值，因此每帧会有 10^{11} 个尾数位（16 位浮点数的尾数恰好为 10 位）。因此，每帧注入的故障数量约为 $10^{10} * p_{faulty}$。所以，我们每帧注入了介于 10^4 （$p_{faulty} = 10^{-6}$）和 10^7 （$p_{faulty} = 10^{-3}$）之间的故障。

5.3 精度指标

为了评估不同近似域对精度的影响，我们采用 [57] 中所述的交并比（IoU）、平均精度（AP）和平均平均精度（mAP）指标。 IoU 的计算方式如下：

$$IoU = \frac{Area_{gt} \cap Area_p}{Area_{gt} \cup Area_p} \tag{1}$$

其中， $Area_{gt}$ 是对应对象的真实边界框面积，$Area_p$ 是其预测边界框的面积。 IoU 的取值范围为 0（边界框不相交时）到 1（边界框的面积和位置完全相同时）。利用 IoU，我们可以设定阈值 t，只有当 $IoU \geq t$ 时，才将预测视为正确，否则视为错误。与 PASCAL VOC 挑战赛 [58] 一致，我们在本例中假设 $t = 0.5$。利用 IoU，我们可以将对象分类为正确检测（真正例，TP）、错误检测（假阳性，FP）和漏检（假阴性，FN）。

我们使用了一个公开可用的框架 [57]来测量平均精度均值，并获取不同实验设置下真正例、假阳性和假阴性的数量。

目标检测器的评估主要基于精度(P)，即仅检测相关对象（也就是在真实标注中表示的对象）的精度指标，以及 召回率(R)，即检测所有相关对象的精度指标。精度和召回率定义如下：

$$P = \frac{TP}{All\ detections} = \frac{TP}{TP + FP} \tag{2}$$

$$R = \frac{TP}{All\ groundtruths} = \frac{TP}{TP + FN} \tag{3}$$

优秀的检测器应识别出所有真实标注目标，并避免预测不存在的目标。因此，FP 和 FN 应接近 0，而 P 和 R 应接近 1。

基于这两个概念，对于给定的目标类别和集合 IoU，平均精度（AP）采用全点插值法进行测量。AP 定义如下：

$$AP_{0.5} = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1}) \tag{4}$$

其中，$AP_{0.5}$ 表示使用 $IoU = 0.5$ 测量 AP，并且

$$P_{interp}(R_{n+1}) = \max_{\{\tilde{R}:\tilde{R} \geq R_n+1\}} P(\tilde{R}) \tag{5}$$

如果不同目标类别分别得到 AP 值，可以对这些值取平均，得到一个称为平均平均精度（mAP）的单一数值。该指标对所有目标类别赋予相同权重，不考虑每种类型的目标频次 [57]。mAP 定义如下：

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{6}$$

6 结果

本节将讨论标注数据集评估、视频案例研究评估和检测结果的置信度范围，并提供一些补充实验，分析漏检区域以及车辆类型的误分类情况。

6.1 标注数据集结果

我们计算了标注数据集中所有类型车辆（汽车、卡车、摩托车和公交车）以及人员类别目标的 AP 值。结果如表1（顶部数据行）所示，其中列出了真正例、假阳性和假阴性的数量，以及四种车辆类别的 mAP，还有四种车辆类别与人员合并后的 mAP。具体而言，我们展示了 YOLO32full、YOLO16full、YOLO16approx 和 YOLO16appfault 的结果，其中 p_{faulty} 在 10^{-6} 和 10^{-3}之间变化。如图所示，降低精度会使真正例和假阳性的数量略微减少，并使假阴性的数量略微增加，但在本例中，最终的 mAP 值仍与 YOLO32full 几乎相同。如果进一步引入近似算术（YOLO16approx），结果仍几乎保持不变。若引入故障，结果在最高达 $p_{faulty} = 10^{-4}$时仅略有变化；故障率更高时，由于假阳性数量增加，差异会变得明显。

总体而言，这些结果表明，由于激进低电压运行，CBOD 对降低精度、近似算术甚至某些中等故障率等多领域近似具有很高的容忍度。

6.2 案例研究（视频）结果

已标注数据集分析的主要缺点是，COCO 数据集中与驾驶场景相关的图像不多（例如由

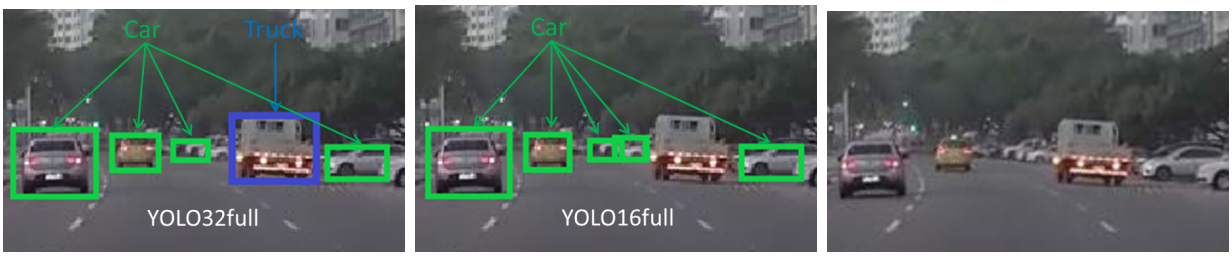


图 4：YOLO32full（左）、YOLO16full（中）检测出不同（但确实存在）的目标，以及原始帧（右）。

车载摄像头拍摄的道路场景图像），因此这项分析的结果在自动驾驶领域的价值有限。

我们还分析了真实驾驶条件下的视频，尽管这些视频没有标注，但仍包含相关数据，正如第 5.1 节所述。使用未标注视频的主要缺点是，缺少真实标注对象，无法评估模型的精度。因此，评估模型精度的唯一方法是将基线 YOLO32full 用作参考配置。然而，基准配置并非 100% 精确。因此，在比较不同配置时，除非对相应帧进行视觉检查，否则无法判断它们之间的差异究竟是由某个配置中的错误检测造成，还是由另一个配置中的错误检测造成。例如，图 4 中的帧就说明了这一点：我们可以看到，YOLO32full 和 YOLO16full 都有 5 个真正例和至少 1 个假阴性。然而，如果不进行视觉检查，而是以 YOLO32full 为参考，我们就会认为 YOLO32full 有 5 个真正例且没有假阴性，而 YOLO16full 有 4 个真正例、1 个假阴性和 1 个假阳性。需要澄清的是，本文提供的所有定量结果都严格根据 YOLO 的输出计算，且各项指标均通过科学框架得出，以避免任何形式的主观视觉检查。因此，视觉检查仅用于定性讨论。这在一定程度上可能导致精度结果偏低。不过，我们认为，在此背景下分析未标注视频仍然十分重要，因为这使我们能够考察真实驾驶条件。

表1（中间行和底部行）展示了与标注数据集类似的结果，但针对的是未标注视频。表中，iT_i 对应单帧目标检测结果（YOLO内部结果），而 T_i 对应如图3所示，对最近三帧的目标检测结果取平均后的结果。

请注意，表中省略了 YOLO32full，因为它用作比较其他设置的参照。结果分别展示了单帧的内部目标列表（中间一组行），以及对连续 3 帧的置信度取平均值后的目标列表（底部一组行）。首先可以观察到，对于所有配置和对象类型（即车辆和人员），相对于 YOLO32full，mAP 值都非常高，但 $p_{faulty} \geq 10^{-4}$时除外。对于 iT_i 和 T_i 目标列表，这一结论均成立。

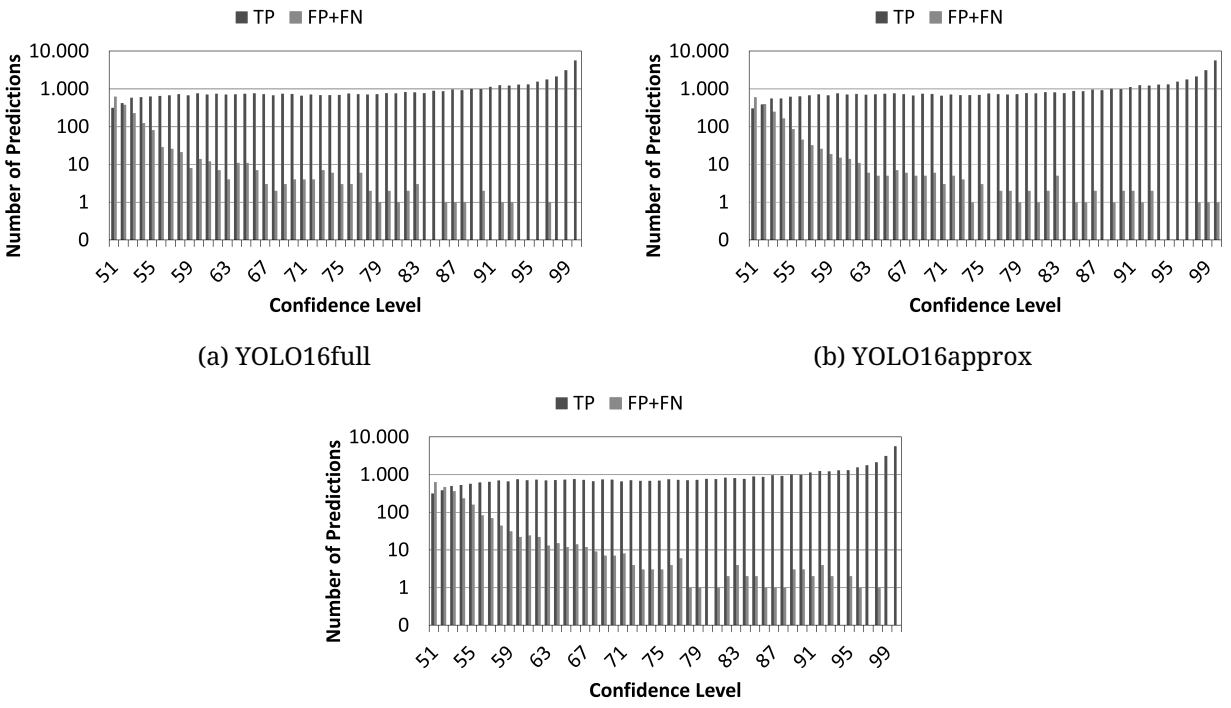
比较这两个列表的结果时，我们发现 T_i 的真正例、假阳性和假阴性值都更低。真正例较低并不意味着有更多物体未被检测到，而是表示即使 YOLO32full 检测到的物体也更少，因为 YOLO 跨帧对置信度取平均所具有的固有容错能力，可以去除偶发的边缘检测（例如物体

表1：带标签数据集（顶部）以及视频用例中的 iT_i 和 T_i 的结果。灰色背景行表示在近似程度方面最激进的配置，其导致的准确率下降可忽略不计（低于1%）。

带标签数据集					
配置	TP	假阳性	假阴性	mAP 车辆	车辆平均精度均值和行人
YOLO32full	5269	1076	4135	59.73	60.47
YOLO16full	5226	1038	4178	59.79	60.39
YOLO16approx	5230	1023	4174	59.49	60.16
YOLO16appfault(10^{-6})	5238	1030	4166	59.26	60.00
YOLO16appfault(10^{-5})	5214			59.29	59.99
YOLO16appfault(10^{-4})	5227	1034	4177	59.20	59.95
YOLO16appfault(10^{-3})	5210	1219	4194	58.53	59.39
视频用例：iT_i – 单帧检测结果					
配置	TP	假阳性	假阴性	mAP 车辆	平均精度均值 车辆和行人
YOLO16full	51297	857	1472	94.78	95.00
YOLO16approx	51142	797	1627	93.69	93.96
YOLO16appfault(10^{-6})	51123	842	1646	95,50	95,50
YOLO16appfault(10^{-5})	5			94,07	94,19
YOLO16appfault(10^{-4})	508041	223	1965	90,71	91,21
YOLO16appfault(10^{-3})	488533	562	3916	83,14	83,77
视频用例：T_i –跨帧平均置信度					
配置	TP	假阳性	假阴性	mAP 车辆	平均精度均值 车辆和行人
YOLO16full	49720	462	1184	96.25	96.21
YOLO16approx	49552	395	1352	97.29	97.14
YOLO16appfault(10^{-6})	49514	408	1390	97,06	96,92
YOLO16appfault(10^{-5})	4			95,42	95，62
YOLO16appfault(10^{-4})	49307	710	1597	92,27	92,84
YOLO16appfault(10^{-3})	476592	422	3245	89,35	88,88

（其置信度仅偶尔略高于 50%）。其他配置也会出现完全相同的效果。由于过滤掉了偶尔出现的不同置信度值，所有配置的 FP 和 FN 值也会降低。总体而言， iT_i 和 T_i 的 mAP 值差异不大。然而，如果我们考察这两个列表的容错能力，就会发现 T_i 对近似的容忍度高得多，因为随着我们引入更多近似域并提高 p_{faulty}， mAP 的性能会缓慢下降。例如， iT_i 的最后两种配置分别导致 mAP 下降约 3% 和 7%；而对于 T_i ，两种情况下的降幅都在 3% 至 4% 左右。

我们还观察到，正如预期，近似对不同对象类型的影响相似，因为这两者彼此正交。相反，对象的尺寸、占用面积与近似的影响之间存在关联，因为检测结果



(a) YOLO16full

(b) YOLO16approx

(c) YOLO16appfault

图 5：使用 T_i 目标列表时，所有对象类型在各置信度下的真正例、假阳性+假阴性。配置分别为 (a) YOLO16full、(b) YOLO16approx 和 (c) YOLO16appfault(10^{-4})。

小目标的置信度通常较低，而近似的影响可能使这些数值降至接受阈值以下或升至其上，因此会产生语义影响。

最后，仅考虑预测高于还是低于阈值，会忽略这些预测的置信度信息。因此，我们测量了真正例和假阳性+假阴性置信度的分布。图 5 展示了使用 T_i 目标列表时，YOLO16full、YOLO16approx 和 YOLO16appfault(10^{-4}) 配置下所有对象类型（不仅限于车辆和人员）的这些数值。请注意，y 轴采用对数尺度。

所有配置都呈现出相似的趋势。如图 5 所示，真正例在不同置信度下分布相当均匀。然而，假阳性+假阴性集中在低置信度范围内，这表明差异对高置信度检测到的目标影响微乎其微。这说明在自动驾驶场景中，目标检测过程天然具有容错性。此外，视觉检查发现，高置信度目标的差异与以下情况有关：(1) 目标的位置和类型识别正确，但由于近似配置或参考配置（YOLO32full）将其他目标也纳入边界框，导致边界框明显不同；(2) 目标相互重叠，各配置检测到的目标子集不同。

总的来说，尽管本研究仅利用了部分冗余级别，但我们已经证明，汽车系统中的基于摄像头的目标检测本质上是一个容错过程，这有助于采用基于多个近似域的专用加速器，例如

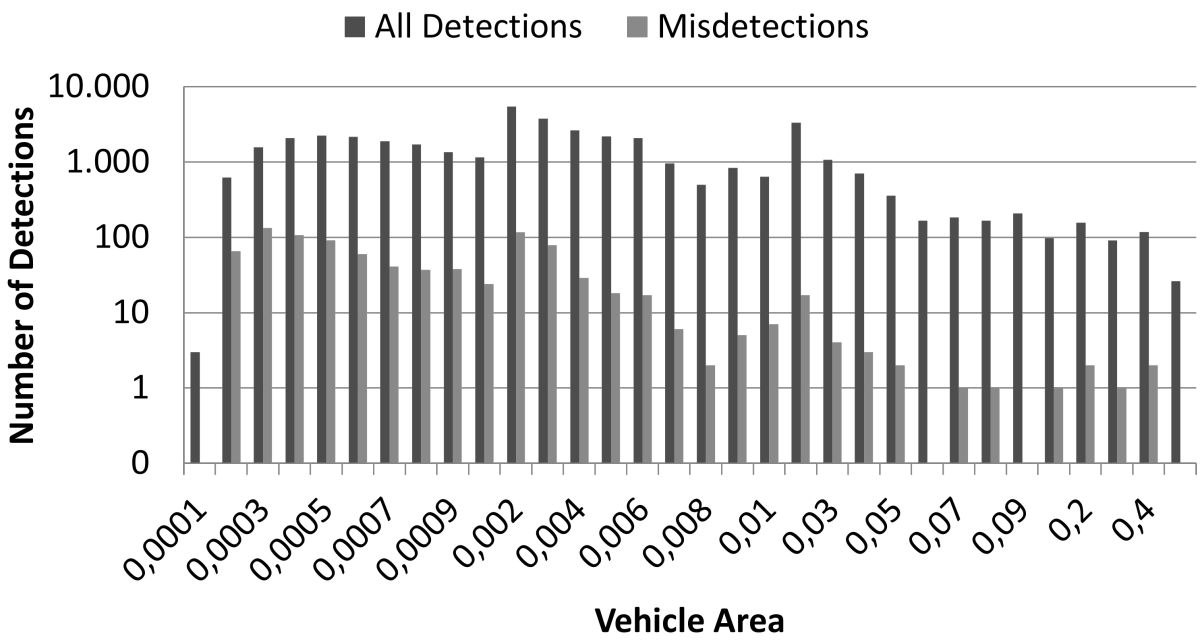


图 6：YOLO16Approx T_i 视频用例中所有检测到和漏检对象的面积。

降低精度、近似算术，以及为实现低功耗运行而采取的激进电压缩放。本研究以渐进的方式分析了不同的近似来源。不过，我们认为，通过分析得出的结论也可以推广到其他情况。例如，降低精度算术和近似算术会对尾数的最低位产生一定影响，但很少会传播到最高位。因此，无论各情况使用何种基线，这些影响预计都同样成立。类似地，当故障注入发生在尾数的低位时，其影响很小。如果应用于 YOLO32full，预计部分故障会影响尾数的最高 10 位（即平均每 23 个故障中有 10 个），因此其影响与在 YOLO16full 中进行故障注入类似；其余故障（即平均每 23 个故障中有 13 个）则会影响尾数的最低 13 位，因此其影响小于在 YOLO16full 的尾数第 10 位注入故障所造成的影响。

6.3 补充实验与结果

6.3.1 漏检面积

另一个需要考虑的因素是特定类型漏检对象的面积。例如，边界框面积很小的汽车意味着它距离较远。因此，只要在距离缩短、汽车边界框面积显著增大时能够检测到它，那么即使有几帧漏检这辆汽车也无妨。

图 6 展示了 YOLO16approx T_i 视频用例中所有车辆检测结果（真正例）和未检测到的对象（假阴性）的面积范围。对象的面积定义为

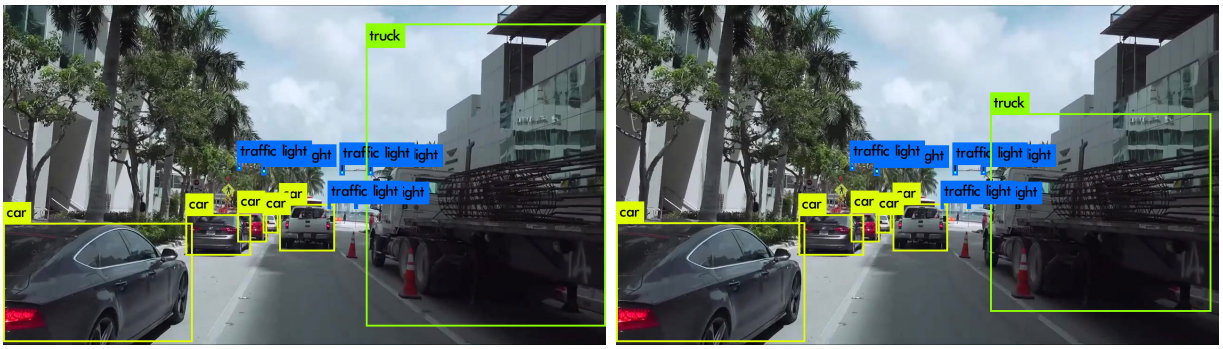


图 7：使用 YOLO32full（左）和 YOLO16full（右）时，对大面积卡车的漏检示例。

边界框面积与图像总面积之比，因此面积值的范围为 0 到 1。例如，面积为 0.5 表示对象占据了图像的一半。请注意，y 轴采用对数尺度。另请注意，x 轴每隔 10 步按数量级递增。

我们观察到，对于面积小于 0.006、被视为小目标的车辆，漏检总数会增加。具体而言，小目标的漏检数通常比检测结果总数低一个数量级，而大目标的漏检数则低两个数量级。因此，我们可以得出结论：目标越大，发生漏检的概率越低。

对于报告中面积较大的目标漏检情况，我们对其中一些案例进行了目视检查，并发现所有案例在实际中都可以认为并未发生漏检。例如，图 7 展示了一辆面积较大、置信度为 0.98，却被判定为漏检的卡车。在此情况下，两种配置都检测到了这辆卡车，但边界框 IoU 低于 0.5，因为 YOLO32full（左图）将卡车后方的建筑物也包含在边界框中。因此，YOLO16full 的这一预测（右图）被同时视为假阳性和假阴性。在这个例子中，YOLO16full 对目标面积的预测比 YOLO32full 更准确，但也可能出现相反的情况。

如图 8 所示，正确检测（TP）的比例在目标面积小于 0.003 时开始下降，并在低于 0.0006 时下降得更加明显。请注意，观测到的最小面积对应的 TP 比例为 100%，但如图 6 所示，该尺寸的目标仅有 3 个。因此，可以忽略这一特定比例，因为如果该面积范围内的目标数量更多，预计其 TP 比例会低于面积较大的目标。

总体而言，这些结果表明，漏检主要来自面积较小的目标，因此在此情境下影响不大。大面积目标的漏检主要是因为两种配置都检测到了同一目标，但边界框之间的 IoU 低于设定的阈值，因此这些检测结果被视为假阳性和假阴性。

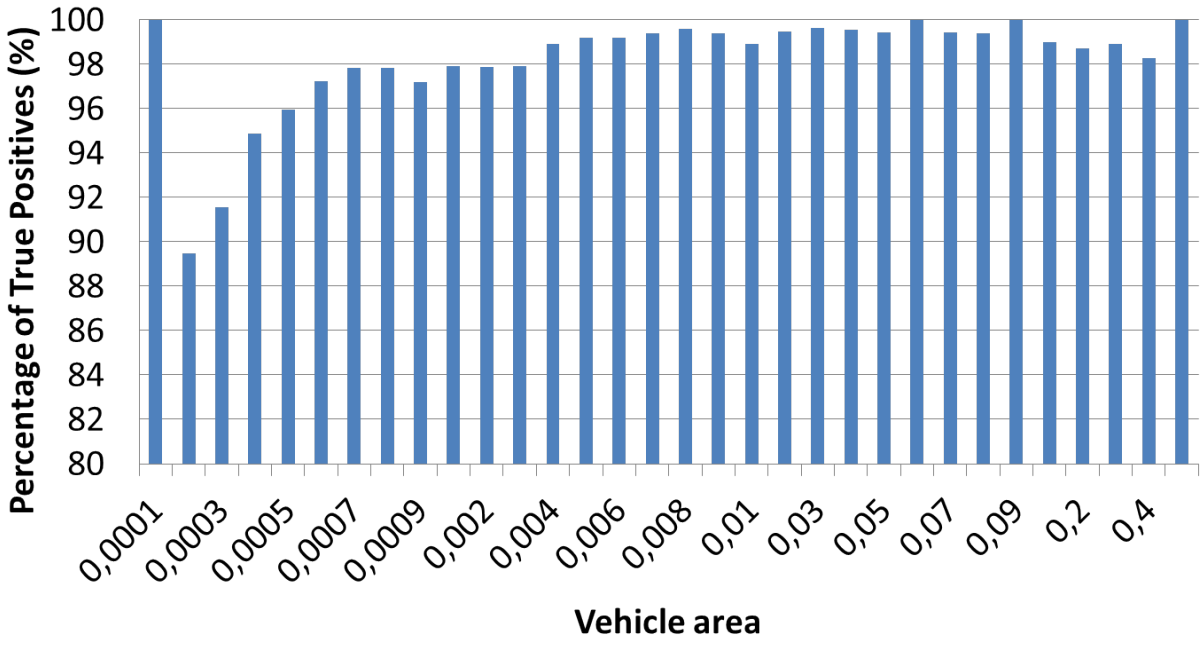


图8：YOLO16Approx T_i 视频用例的正确检测（真正例）百分比。

6.3.2 车辆类型的误分类

通过视觉检查，我们注意到，在某些情况下，不同配置会检测到同一目标，却将其分类为不同类别。例如，一种配置可能将某辆车辆分类为汽车，另一种配置则可能将其分类为卡车；面包车的情况尤其如此，因为该 YOLO 模型中没有面包车这一目标类别。目标误分类可能会对平均精度均值指标产生重大影响，因为在这种情况下，误分类的目标既被视为假阳性，也被视为假阴性：模型未能检测到（此处指未能正确分类）一个目标，却检测到了一个新目标（此处实际上是同一目标，但属于不同的目标类别）。

为了评估车辆类型误分类的影响，我们将基线 YOLO 模型中包含的不同车辆类别（汽车、摩托车、卡车和公交车）合并为一个名为 车辆 的类别。我们将此配置称为 泛化 配置。

图9显示了标注数据集中所有车辆类型的真正例和假阳性数量。在所有配置中，泛化模型的真正例数量略有增加，假阳性数量则相应减少。因此，从这些指标来看，目标检测器的精度有所提高。另一方面，图10中车辆的平均精度均值结果显示，在某些配置（YOLO16full 和 YOLO16Approx）下，泛化模型的车辆平均精度均值有所下降，这在一定程度上与之前观察到的精度提高相矛盾。这表明平均精度均值指标存在一个弱点。具体而言，基线的车辆平均精度均值是不同车辆类型平均精度的简单均值，没有考虑各类别的观测数量；而泛化车辆的平均精度则是作为单一数值计算的，自然地对所有目标进行了加权。这意味着，如果某种车辆类型的平均精度极高或极低，但与其他车辆类型的检测结果数量相比，该类型仅占全部目标的一小部分，那么

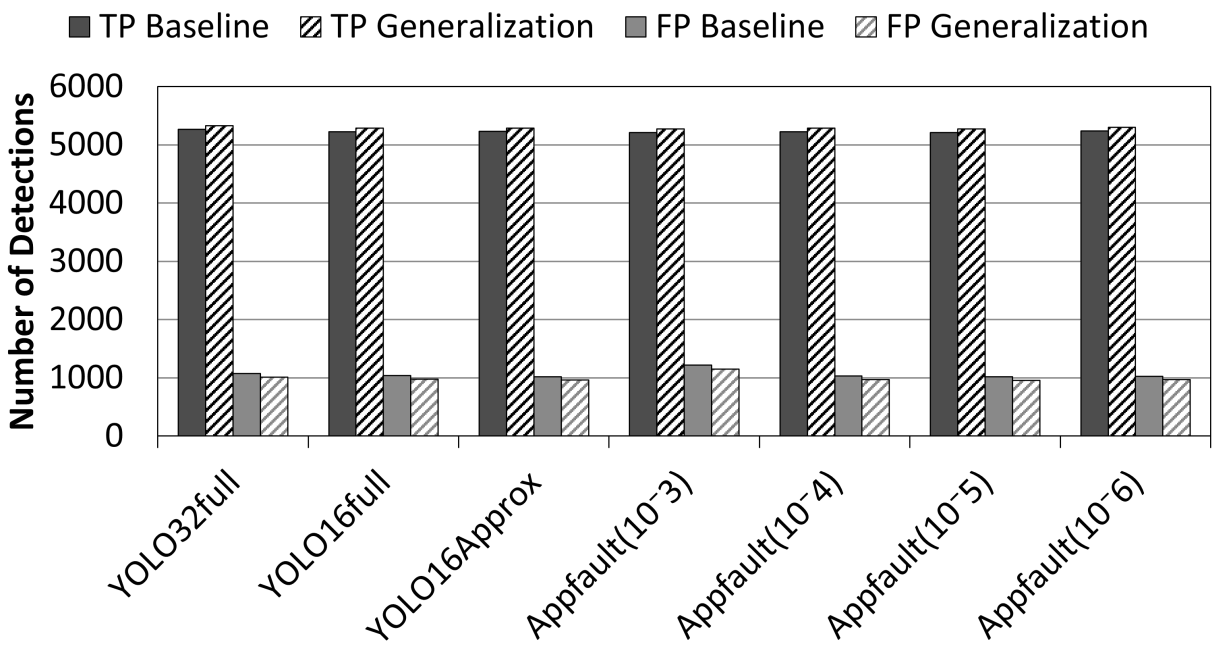


图9：标注数据集上基线配置与泛化配置的真阳性/预测和假阳性/预测总数对比。

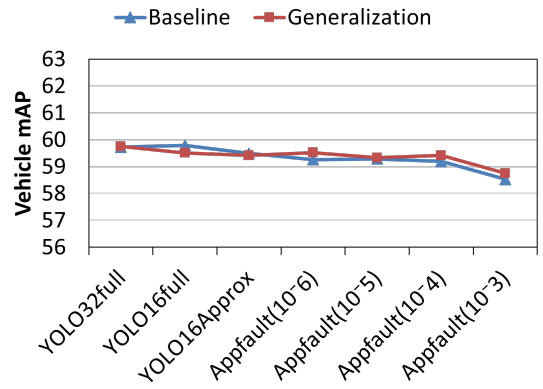


图10：标注数据集上基线配置与泛化配置的车辆平均精度均值对比。

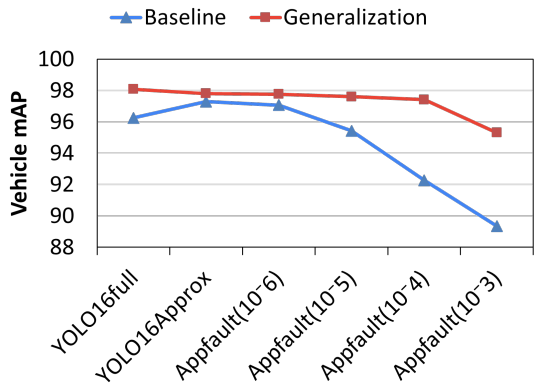


图11： T_i 视频用例中基线配置与泛化配置的车辆平均精度均值对比。

车辆的平均精度均值可能会受到显著影响。

对于视频用例，我们还发现，泛化后的真正例数量增加，而假阳性数量减少。另一方面，在本例中，车辆泛化的平均精度均值在所有配置下都更高，如图11中的 T_i 配置所示。 iT_i 配置也呈现类似趋势，但由于无法提供更多见解，因此未予展示。仔细分析这些视频可以发现，如果不进行泛化，而是将不同类型的车辆保留在不同类别中，YOLO32full会出现

相当多的误分类。因此，近似配置往往能够正确分类目标类别，而YOLO32full却无法做到。因此，假设近似配置一旦与YOLO32full的结果不一致就总是错误的，未免过于悲观。然而，由于视频没有标注，因此没有自动纠正这一问题的方法。将车辆泛化为单一类别，是缓解缺少已标注驾驶视频所带来问题的一种方法。

通过分析已标注和未标注数据，我们可以得出结论：在某些情况下，不同配置会检测到同一目标，却将其分类为不同类别；因此，如果仅分析平均精度均值，结果就会呈现更大的变异性。将所有车辆目标视为同一类别，可以避免基准配置对未标注数据进行误分类所带来的问题。此外，在许多情况下（但不一定总是如此），将车辆目标划分到错误的子类别中（例如，将汽车分类为货车）在语义上影响甚微。

7 结论

自动驾驶中的目标检测是一种基于深度学习的随机过程，因此会产生假阳性和假阴性。为容忍错误，目标检测采用传感器冗余（多个摄像头、激光雷达和雷达）以及时间冗余（利用一段时间内的检测结果），从而使偶发错误不会对输出产生可见影响。

本文分析了可用于降低高功耗基于摄像头的目标检测过程能耗和复杂度的不同近似域所带来的语义影响。具体而言，在汽车案例研究的背景下，我们展示了由较低精度和近似算术导致的大量轻微错误，以及由激进低电压运行引发的偶发任意错误，最终对目标检测的语义影响可以忽略不计，尽管它们会对检测结果的置信度产生一定影响，甚至偶尔造成漏检。针对标注数据集和真实驾驶视频的结果表明，即使同时考虑三个近似域——将浮点精度从 32 位降低至 16 位、采用近似算术，以及考虑数据尾数中高达 10^{-5} 的故障率——精度（以平均精度均值衡量）仍在原始值的 1% 以内。总的来说，我们的研究为开发利用多个近似域的专用加速器提供了有力依据，尤其适用于汽车领域等成本受限的领域。

致谢

本研究部分由 DRAC 项目资助。该项目在加泰罗尼亚 2014—2020 年 ERDF 运营计划框架内获得欧盟区域发展基金共同资助，资助金额为符合条件总成本的 50%。本研究还部分获得了西班牙科学与创新部的资助，项目编号为 PID2019-107255GB-C21/AEI/10.13039/501100011033。

参考文献

[1] J.Redmon和A. Farhadi。Yolo9000：更好、更快、更强。载于*IEEE*计算机视觉与模式识别会议（*CVPR*），第6517–6525页，2017年。[2]JosephRedmon和Ali Farhadi。Yolov3：渐进式改进。*ArXiv*，abs/1804.02767，2018年。[3] NVIDIA DRIVE -自动驾驶汽车开发平台。 https://developer.nvidia.com/drive。访问于2022年10月。[4] Apollo，一个开放式自动驾驶平台。 https://apollo.auto/，2018年。[5] Keiron O'Shea和Ryan Nash。卷积神经网络简介。*CoRR*，abs/1511.08458，2015年。[6] H. Wang、X. Jiang、H. Ren、Y. Hu和S. Bai。Swiftnet：实时视频目标分割。载于*2021*年*IEEE/CVF*计算机视觉与模式识别会议（*CVPR*），第1296–1305页，美国加利福尼亚州洛斯阿拉米托斯，2021年6月。IEEE计算机学会。[7] ChaoweiFang、HaibinTian、Dingwen Zhang、QiangZhang、JungongHan和Junwei Han。用于显著目标检测的密集嵌套自顶向下流，2021年。[8] Guobin Chen、Wongun Choi、XiangYu、Tony Han和Manmohan Chandraker。通过知识蒸馏学习高效的目标检测模型。载于第*31*届神经信息处理系统国际会议论文集，NIPS'17，第742–751页，美国纽约州雷德胡克，2017年。CurranAssociates Inc。[9] Tianqi Tang、Shen Li、YuanXie和NormJouppi。Mplat：用于机器学习加速器的功耗、面积和时序建模框架。载于*DOSSA*研讨会，美国纽约州纽约，2018年。[10] Norman P. Jouppi等。数据中心内张量处理单元的性能分析。载于第*44*届计算机体系结构年度国际研讨会论文集，ISCA '17，第1–12页，美国纽约州纽约，2017年。美国计算机协会。[11] Chen Wu、Mingyu Wang、XinyuanChu、KunWang和Lei He。用于高性能基于FPGA的CNN加速的低精度浮点算术。*ACM*可重构技术系统汇刊，15(1)，2021年11月。[12] Xiangyu Zhang、Jianhua Zou、Kaiming He和JianSun。加速用于分类和检测的超深卷积网络，2015年。[13] Joseph Redmon、Santosh Divvala、Ross Girshick和Ali Farhadi。You Only Look Once：统一的实时目标检测。载于*2016*年*IEEE*计算机视觉与模式识别会议（*CVPR*），第779–788页，2016年。[14] NaveenMellempudi、Sudarshan Srinivasan、DipankarDas和Bharat Kaul。使用8位浮点数进行混合精度训练，2019年。

[15] Yunji Chen、Tao Luo、Shaoli Liu、Shijin Zhang、Liqiang He、Jia Wang、Ling Li、Tianshi Chen、ZhiweiXu、NinghuiSun 和 Olivier Temam。Dadiannao：一台机器学习超级计算机。载于*2014*年第*47*届*IEEE/ACM*年度微体系结构国际研讨会，第609–622页，2014年。[16] Zachariah Carmichael、Hamed F. Langroudi、Char Khazanov、Jeffrey Lillie、John L. Gustafson和Dhireesha Kudithipudi。深度神经网络中低精度数值格式的性能-效率权衡。*2019*年下一代算术会议论文集，2019年3月。[17] Darryl D. Lin、Sachin S. Talathi 和V.Sreekanth Annapureddy。深度卷积网络的定点量化。载于 第*33*届国际机器学习会议论文集——国际机器学习会议第*48*卷，ICML'16，第2849–2858页。JMLR.org，2016年。[18] Suyog Gupta、Ankur Agrawal、Kailash Gopalakrishnan和 Pritish Narayanan。采用有限数值精度进行深度学习。载于 第*32*届国际机器学习会议论文集——国际机器学习会议第*37*卷，ICML'15，第1737–1746页。JMLR.org，2015年。[19] Issam Hammad、Ling Li、Kamal El-Sankary和 W. MartinSnelgrove。使用预处理精度控制器和具有各种精度的近似乘法器进行CNN推理。*IEEE Access*，9:7220–7232，2021年。[20] Karen Simonyan和 Andrew Zisserman。用于大规模图像识别的超深卷积网络。载于第*3*届国际学习表征会议（*ICLR2015*），2015年。[21] F.Chollet。Xception：采用深度可分离卷积的深度学习。载于 *2017*年*IEEE*计算机视觉与模式识别会议（*CVPR*），第1800–1807页，洛斯阿拉米托斯，加利福尼亚州，美国，2017年7月。IEEE计算机学会。[22] Shuihua Wang和Yu-Dong Zhang。采用复合学习因子和预计算进行多发性硬化症分类的基于DenseNet-201的深度神经网络。*ACM*多媒体计算、通信与应用汇刊，16:1–19，2020年6月。[23] Benjamin Recht、Rebecca Roelofs、Ludwig Schmidt和Vaishaal Shankar。ImageNet分类器能否泛化到ImageNet？，2019年。[24] Ali Ibrahim、Mario Osta、Mohamad Alameh、Moustafa Saleh、Hussein Chible 和Maurizio Valle。用于嵌入式机器学习的近似计算方法。载于 *2018*年第*25*届*IEEE*国际电子、电路与系统会议（*ICECS*），第845–848页，2018年。[25] N. Manikandan。加速基于CNN的图像处理应用的近似计算技术——硬件/软件视角下的综述。国际计算机科学与工程高级趋势期刊，9(3)，2020年。

[26] Doochul Shin 和 Sandeep K. Gupta。面向容错应用的数据通路模块重新设计技术。载于*2008*年 第*17*届亚洲测试研讨会，第431–437页，2008年。[27] Parag Kulkarni、Puneet Gupta 和 Milos Ercegovac。采用欠设计乘法器架构，以精度换取功耗。载于*2011*年 第*24*届国际*VLSI*设计会议，第346–351页，2011年。[28] Behzad Salami、Erhan Baturay Onural、Ismail Emir Yuksel、Fahrettin Koc、Oguz Ergin、Adrian Cristal Kestelman、Osman Unsal、Hamid Sarbazi-Azad 和 Onur Mutlu。现代现场可编程门阵列降压运行在神经网络加速中的实验研究。载于*2020*年 第*50*届*IEEE/IFIP*年度可靠系统与网络国际会议（*DSN*），第138–149页，2020年。[29] Nandhini Chandramoorthy、Karthik Swaminathan、Martin Cochet、Arun Paidimarri、Schuyler Eldridge、Rajiv V. Joshi、Matthew M. Ziegler、Alper Buyuktosunoglu 和 Pradip Bose。面向高能效的高可靠性低电压加速器。载于*2019*年*IEEE*高性能计算机体系结构国际研讨会（*HPCA*），第147–158页，2019年。[30] Annachiara Ruospo、Alberto Bosio、Alessandro Ianne 和 Ernesto Sanchez。评估卷积神经网络可靠性与其数据表示之间的关系。载于*2020*年 第*23*届*Euromicro*数字系统设计会议（*DSD*），第672–679页，2020年。[31] Y. Lecun、L. Bottou、Y. Bengio 和 P. Haffner。应用于文档识别的基于梯度的学习。《*IEEE*会刊》，86(11):2278–2324，1998年。[32] Li Deng。用于机器学习研究的手写数字图像MNIST数据库。《*IEEE*信号处理杂志》，29(6):141–142，2012年。[33] F. Libano、P. Rech、B. Neuman、J. Leavitt、M. Wirthlin 和 J. Brunhaver。数据精度与并行度如何影响现场可编程门阵列上卷积神经网络的可靠性。《*IEEE*核科学汇刊》，68(5):865–872，2021年。[34] Saurabh Jha、Shengkun Cui、Timothy Tsai、Siva Kumar Sastry Hari、Michael B. Sulli- van、Zbigniew T. Kalbarczyk、Stephen W. Keckler 和 Ravishankar K. Iyer。利用时间数据多样性检测自动驾驶计算系统中的安全关键故障。载于*2022*年 第*52*届*IEEE/IFIP*年度可靠系统与网络国际会议（*DSN*），第88–100页，2022年。[35] Lucas Klein Draghetti、Fernando Fernandes dos Santos、Luigi Carro 和 Paolo Rech。利用帧间时空相关性检测卷积神经网络中的错误。载于*2019*年 *IEEE*第*25*届在线测试与稳健系统设计国际研讨会（*IOLTS*），第310–315页，2019年。[36]Perception Apollo 3.0。`https://github.com/ApolloAuto/apollo/blob/master/docs/specs/perception_apollo_3.0.md`，2018年。[37] Autoware。开放式自动驾驶平台。 `https://github.com/Autoware-AI/autoware.ai`。访问于2022年10月。

[38] YunchaoGong 等。使用向量量化压缩深度卷积网络，2014年。[39] Honglan Jiang、Francisco Javier Hernandez Santiago、HaiMo、Leibo Liu 和 Jie Han。近似算术电路：综述、特征分析与近期应用。《*IEEE*会刊》，108(12):2108–2135，2020年。[40] Hamid Tabani、Roger Pujol、Jaume Abella和 FranciscoJCazorla。深度学习框架的跨层评述，以简化其优化与复用。载于*2020*年 *IEEE*第*23*届实时分布式计算国际研讨会（*ISORC*），第144–145页。IEEE，2020年。[41] Shekhar Borkar、Tanay Karnik、Siva Narendra、Jim Tschanz、AliKeshavarzi 和 Vivek De。参数变化及其对电路和微架构的影响。载于 第*40*届年度设计自动化会议论文集，DAC '03，第338–342页，纽约州纽约，美国，2003年。美国计算机协会。[42] Guanpeng Li、Siva Kumar Sastry Hari、Michael Sullivan、Timothy Tsai、Karthik Pat-tabiraman、Joel Emer和 Stephen W. Keckler。理解深度学习神经网络（DNN）加速器中的错误传播及其应用。载于高性能计算、网络、存储与分析国际会议论文集，SC'17，纽约州纽约，美国，2017年。美国计算机协会。[43] Alberto Bosio、Paolo Bernardi、AnnachiaraRuospo和 Ernesto Sanchez。深度神经网络的可靠性分析。载于 *2019*年*IEEE* 拉丁美洲测试研讨会（*LATS*），第1–6页，2019年。[44] D.Ernst、S.Das、S. Lee、D. Blaauw、T. Austin、T. Mudge、NamSung Kim和K. Flaut- ner。Razor：面向低功耗运行的电路级时序错误校正。*IEEE Micro*，24(6):10–20，2004年。[45] Joseph Redmon 和 AliFarhadi。用于 yolov3 的 Darknet 框架。 `https://github.com/pjreddie/darknet`，2018年。[46] Tsung-Yi Lin、MichaelMaire、Serge Belongie、James Hays、Pietro Perona、DevaRamanan、Piotr Doll´ar 和 C.Lawrence Zitnick。Microsoft coco：上下文中的常见对象。载于 David Fleet、Tomas Pajdla、Bernt Schiele 和 TinneTuytelaars 编，《计算机视觉 – *ECCV 2014*》，第740–755页，Cham，2014年。SpringerInternational Publishing。[47] J. Utah。Rio 4K- 科帕卡巴纳海滩- 清晨驾驶， [时间：2:50 -3:20], [帧数：5094- 5971]。 `https://www.youtube.com/watch?v=_hWCN1yV9TY`，2020年。[48] J. Utah。旧金山4K- 夜间驾驶， [时间：2:20 - 2:50], [帧数：4214 - 5091]。 `https://www.youtube.com/watch?v=jJ08h2cgWjI`，2020年。[49] J. Utah。旧金山 4K- 市中心驾驶， [[时间：2:00 - 2:30], [帧数：3639 - 4516]。 `https://www.youtube.com/watch?v=E7t3QyEfyLA`，2020年。

[50] J. Utah。洛杉矶 4K——摩天大楼都市——威尔希尔大道， [时间：18:30 - 19:00], [帧：33237 - 34114]。https://www.youtube.com/watch?v=zfblxgasy-0，2020年。[51] J. Utah。迈阿密 4K——黄金海岸——风景驾车之旅， [时间：13:25 - 13:55], [帧：24142 - 25019]。 https://www.youtube.com/watch?v=xj7abSp07w0，2020年。[52] J. Utah。拉斯维加斯 4K——日落驾车之旅， [时间：1:10 - 1:40], [帧：2097 - 2974]。 https://www.youtube.com/watch?v=X9U5DafT0d4，2020年。[53] Shanshan Zhang、Rodrigo Benenson 和 Bernt Schiele。Citypersons：用于行人检测的多样化数据集，2017年。[54] Andreas Geiger、Philip Lenz 和 Raquel Urtasun。我们准备好迎接自动驾驶了吗? KITTI 视觉基准套件。载于计算机视觉与模式识别会议（*CVPR*），2012年。[55] Xinyu Huang、Peng Wang、Xinjing Cheng、Dingfu Zhou、Qichuan Geng 和 Ruigang Yang。用于自动驾驶及其应用的 ApolloScape 开放数据集。*IEEE* 模式分析与机器智能汇刊，42(10):2702–2719，2020年10月。[56] John Hauser。Berkeley SoftFloat，2018年。[57] Rafael Padilla、Sergio L. Netto 和 Eduardo A. B. da Silva。目标检测算法性能指标综述。载于*2020*年国际系统、信号与图像处理会议（*IWSSIP*），第237–242页，2020年。[58] M. Everingham、L. Van Gool、C. K. I. Williams、J. Winn 和 A. Zisserman。Pascal 视觉目标类别（VOC）挑战赛。国际计算机视觉期刊，88(2)，2010年。