arXiv:2301.01758v1 [cs.DC] 4 Jan 2023

一种用于经济且准确的血糖仪读数的集成移动云计算方法

Navidreza Asadi 和 Maziar Goudarzi，*IEEE* 高级会员

摘要——尽管在研发用于定期监测血液指标的先进无线医疗设备方面付出了巨大努力，但许多国家仍有许多人无法获得这类设备，或无力承担其费用。若改用普通设备，患者就必须手动将数据记录到移动健康监测应用中。据医疗专家称，这会导致一些问题：(1) 由于需要人为直接介入，容易出错，而且据报道，用户往往会输入不切实际的数据；(2) 每天多次输入数值十分麻烦，会导致用户弃用移动应用。因此，亟需利用如今已无处不在的智能手机，通过拍摄医疗设备屏幕图像并自动提取有用信息来减少错误。然而，这一技术的开发面临若干挑战：(1) 数据稀缺导致现有方法不切实际，准确率极低：据我们所知，在这种情况下可用的数据集很少；(2) 准确性与可用性权衡：可以在手机上运行准确率较低的算法，以维持较高的可用性；也可以在云端部署准确率更高、计算开销更大的算法，但在连接性较差或没有连接的情况下，可用性会降低。我们提出了一种集成学习算法、一种移动云计算服务架构和一种简单的压缩技术，通过融合云端和移动端预测结果，在提高准确率的同时提升可用性并缩短响应时间。此外，我们还提出一种生成合成训练数据的算法，以便利用深度学习模型提高准确率。所提方法实现了三大目标：(1) 在两个不同的数据集上，准确率分别达到92.1% 和97.7%，较先前方法提升了 ∼40%；(2) 将所需带宽降低 45×，准确率仅下降 ∼1%；(3) 与仅移动端、仅云端、分割计算和早退服务模型相比，提供更高的可用性。

索引词—移动计算、集成学习、数据生成、深度学习、智慧健康

1 引言

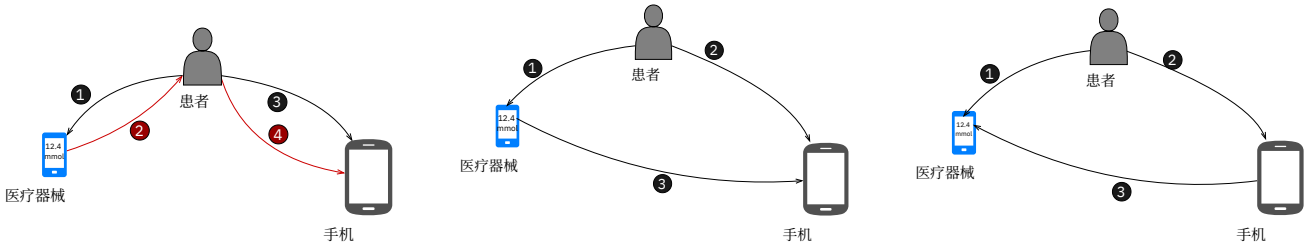
MANY 移动健康/泛在健康医疗设备，尤其是在中低收入国家价格实惠的设备，会在数字屏幕或七段数码管屏幕上显示测量值，以及日期、时间、图表和测量单位等其他附加信息。在大多数常用的移动健康服务中，尤其是面向糖尿病患者的服务，患者需要手动将感测值输入移动应用。如图1(a)所示，用户首先从医疗设备上读取数值，然后打开应用，进入记录界面，最后通过打字录入信息。每次记录测量结果时，都必须重复这些步骤。

1.1 研究动机

根据相关报告 [1], [2] 以及我们在糖尿病管理应用 iDia 中积累的经验 [3],，这一流程存在一些缺点：（1）每天多次手动记录令人望而却步；这一过程既麻烦又耗时。根据我们收到的反馈，用户最终会失去使用该应用的兴趣。（2）这一流程容易出现人为错误。更重要的是，观察结果表明，患者可能会倾向于输入虚假信息，使数值看起来更容易接受，也更接近正常数值。这会对预防、控制和治疗的整个过程产生相当大的负面影响。

目前有两种较好的方法：（1）一些医疗设备能够通过远距离通信技术（例如蓝牙）将数据传输到移动设备，从而简化记录流程（图 1(b)）。然而，这类设备价格高得多，而且在某些情况下准确率较低 [4]。更重要的是，它们大多只能与自带的软件应用程序交互，不允许第三方应用程序接收数据。此外，传输技术的不同版本会导致不同移动设备与医疗设备之间不兼容。尽管这类设备的前景可能十分光明，但目前其市场占有率不到百分之一 [5]。在发展中国家，这一比例甚至更低。（2）另一种方法近年来受到学术界关注，它利用移动设备（即智能手机）的图像采集和计算能力；这类设备几乎人人都有，也能够执行轻量级计算任务。图 1(c) 展示了这种方法：使用手机拍摄医疗设备的图像，然后通过图像处理自动识别感测值。在这种方法中（图 2），每个数字都被视为一个具有感兴趣区域（RoI）的对象；按顺序排列的数字（即感兴趣区域）构成一个读数字符串。正确读取不仅意味着所有数字都预测正确，还意味着顺序与设备上显示的顺序一致。本文采用这种方法，因为它具有更广泛的适用性，尤其适用于中低收入国家。

N. Asadi 现任德国慕尼黑工业大学计算机工程系（*navidreza.asadi@tum.de*）。在参与本项目期间，他曾就职于伊朗谢里夫理工大学计算机工程系。*M. Goudarzi* 任职于伊朗谢里夫理工大学计算机工程系（*goudarzi@sharif.edu*）。稿件已提交至《*IEEE*移动计算汇刊》。



(a) 手动 (b) 无线通信 (c) 基于图像

图 1. 记录方法。为避免错误记录，最好去除人工干预（红色箭头）。

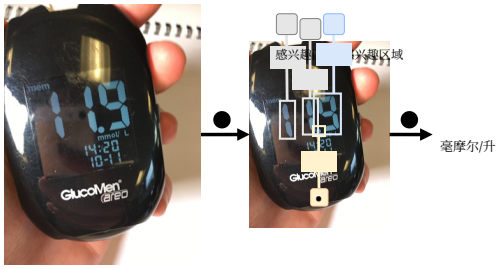


图 2. 基于图像的方法，展示了感兴趣区域（RoIs）和正确标签。预期的正确读数为“11.9 毫摩尔/升”。

1.2 挑战

准确率。虽然自动化图像读取看起来可能基础而简单，但有几点表明，在这一特定情形下它是一个颇具挑战性的问题：（1）当前最先进水平的研究 [6], 虽在很大程度上改进了以往算法，但仅使用两种医疗设备，准确率也只有**51.5%** ，这意味着近一半捕获的图像都被误读。（2）正如我们在第 2 节和第 3 节中所讨论的，医疗设备种类繁多，结构和视觉差异各异，屏幕上还存在噪声信息（例如日期和时间），这些因素使得使用常规图像处理技术读取感测值极为困难。

数据稀缺。深度学习算法的准确率取决于两种条件之一：使用大量标注数据以监督方式进行训练，或采用针对相关任务预训练的模型。据我们所知，在基于图像的读数识别这一特定任务中，既没有大型数据集，也没有相关的预训练模型。生成模型（例如生成对抗网络）也无济于事，因为它们需要使用类似的数据进行训练，而这不适用于我们的情况。此外，要为我们的使用场景生成自动标注可能非常困难（甚至不可能）。

资源限制。当前最先进水平的深度学习模型计算量大，需要部署在专用云基础设施上。因此，它们带来了新的挑战，包括网络状况较差时的可用性问题，而这确实是我们目标欠发达国家中预期会出现的问题。边缘计算 [7],是指靠近数据源且具备一定计算能力的每台设备（例如智能手机），被认为是一种很有前景的方法，可通过减少延迟来提高可用性和服务质量（QoS）。移动设备通常受到资源限制，因此只能运行更简单的深度神经网络（DNN）模型，其准确率也低得多。然而，云计算处于完全相反的一端。

1.3 主要贡献

我们提出了切实可行的解决方案，以应对上述挑战。我们提出了一种集成移动端-云端计算架构，以兼得两者的优势：通过集成云端和移动端模块（图 3，详见第 3 节），在移动端实现更高的可用性，同时获得更高的准确率和增强的性能表现（即衡量系统性能/服务质量水平的指标）。以下项目是我们的主要贡献：

(1) 一种混合移动云服务架构，以及与之兼容的集成深度学习算法。该方案可根据网络连接情况权衡准确性与可用性。我们解决了组合两个独立模型的预测结果所面临的挑战，例如每个数据元素对应的已识别边界框之间的差异和重叠。

(2) 一种简单而有效的压缩方法。结合我们的集成模型，即使传输的数据量较少，也能实现更高的准确率。

(3) 一种高保真数据合成器算法，使深度学习模型的应用成为可能。这基本上将挑战转化成了机遇：血糖仪型号、数据格式、单位、字体等的多样性对传统方法构成挑战，但我们将其用于数据合成机制，生成了足够且合理的数据，以训练准确率高的模型，涵盖多种情况，包括此前未见过的情况。

所提方法在两个真实世界数据集上分别达到了92.1% 和97.7% 的准确率，相较于此前发表的结果提升了40%以上。该方法将所需带宽降低了 45×，并且与仅移动端、仅云端、分割计算和早退服务模型相比，保持了更高的可用性。只需稍作修改，即可轻松将我们提出的方法扩展到其他用例。

本文其余部分安排如下：第 2 节回顾相关工作。第 3 节介绍所提方法。第 4 节阐述数据集生成算法。第 5 节评估我们的方法和算法，并在第 6 节总结全文。

2 相关工作

我们将相关工作分为两个部分。第一部分介绍用于读取感测值，或检测并识别数字屏幕或七段屏幕上数字的算法。第二部分总结了尝试在移动端部署部分或完整深度学习模型的研究。

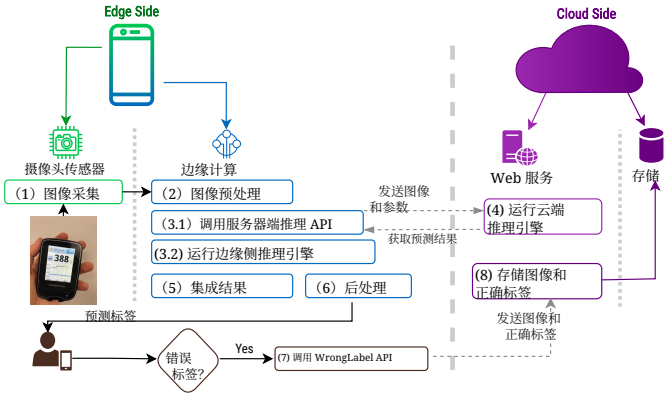


图 3. 我们的移动云架构。

2.1 基于图像的自动读数

大多数所提方法将问题分解为多个步骤，包括图像增强、感兴趣区域定位、检测与分类，最后进行排序。我们从五项标准对相关工作进行综述，如表 1 所示。

自动定位。 定位感兴趣区域是关键步骤，会影响最终准确率。许多研究试图简化这一问题，同时假设定位步骤已通过某种方式完成：由客户端手动定位，或通过固定设备或使用特殊标记来实现 [8], [9], [10], [11],[13], [15]。还有少数研究 [6], [12], [16], 以及我们的工作采取了更全面的方法，也应用自动定位。

准确率。 据我们所知，此前没有工作能达到合理的读数准确率。 [6] 以51.5% 的准确率迄今表现最佳。通用 OCR 引擎也无济于事；我们此前的实验以及 [15], [16],和 [6] 的报告均证实，最先进的 OCR 引擎 Tesseract 在这项特定任务中的表现很差（<10%）。另一方面，我们的方法在两个数据集上的准确率均可超过90% 。

鲁棒性。 不同的方法，尤其是采用传统算法的方法，通常容易受到倾斜度、各种噪声、相机透视与角度、曝光和照明等不同条件的影响。这里，我们将鲁棒性定义为在不同条件下始终保持稳定的表现。也就是说，只有少数研究 [6], [16] 尝试解决这一问题。我们在数据合成器中模拟了所有这些变化。

表 1 基于图像的自动读取文献综述汇总。

[8]	✗	✗	✗	✗	✓
[9]	✗	✗	✗	✗	?
[10]	✗	✗	✗	✗	?
[11]	✗	✗	✗	✗	?
[12]	✓	✗	✗	✗	✓
[13]	✓	✗	✗	✗	✓
[14]	✓	✗	✗	✗	✓
[15]	✗	✗	✗	✗	✗
[16]	✓	✗	✓	✓	✓
[6]	✓	✗	✓	✗	✓
Ours	✓	✓	✓	✓	✓

* 对不同环境条件的鲁棒性。? 信息不可用。

此外，集成两个不同的模型可以减轻此类错误。

泛化能力。 医疗设备种类繁多，各自具有独特特征，例如不同的字体样式（包括各种七段数码管样式和/或数字样式）、背景色和前景色、屏幕尺寸和形状、单位、背光等（图 4）。然而，除 [16], 外，先前的研究只使用一种或极少数特定设备，因此其提出的算法高度依赖所选设备的属性；因而可能无法被视为通用解决方案。例如， [13] 是为一种带蓝色背光屏幕的特定医疗器械而设计的， [6] 仅考虑七段数码管屏幕，而 [15] 则假设轮廓最大的区域就是屏幕，并将感兴趣区域的确切位置硬编码。相比之下，我们涵盖了广泛的市场，并使用来自牛津的一个附加公开数据集来说明这一点 [6]。

响应时间。 由于以往研究大多采用传统且轻量级的图像处理方法，因此能够实现合理的响应时间。例如， [13] 在 Samsung Galaxy S i9000 移动设备上实现，可处理每秒 20 帧；[12] 部署在 N95 移动设备上，可达到每秒五帧。唯一的例外是 [15] ，它在数字分类步骤中使用了深度卷积神经网络（CNN），耗时 10 秒，令人难以接受。尽管我们采用基于 CNN 的目标检测模型，但通过选择合适的参数，使响应时间不到半秒。此外，在网络状况不佳时使用云端引擎，我们设计了一种简单的压缩技术，因此缩短了端到端响应时间。

2.2 边缘端深度学习

在实际应用中，边缘设备可以是任何能够生成数据或靠近数据生成源的计算设备（包括智能手机）。边缘设备通常受到资源限制，因此在边缘端部署大型深度学习模型颇具挑战。表 2 对不同方法进行了简要比较。为克服这些局限所做的努力可分为三个主要方向。

第一个方向是设计轻量级深度神经网络模型，或优化现有模型。已有多项成功研究探索了轻量级模型，包括 [17], [18], [19],[20], [21], 和 [30]。总体而言，这一类别的相关工作结合采用参数较少的卷积块（例如，可分离卷积）、量化、剪枝和模型蒸馏等方法。这些技术

表 2 边缘深度学习相关工作：综述。

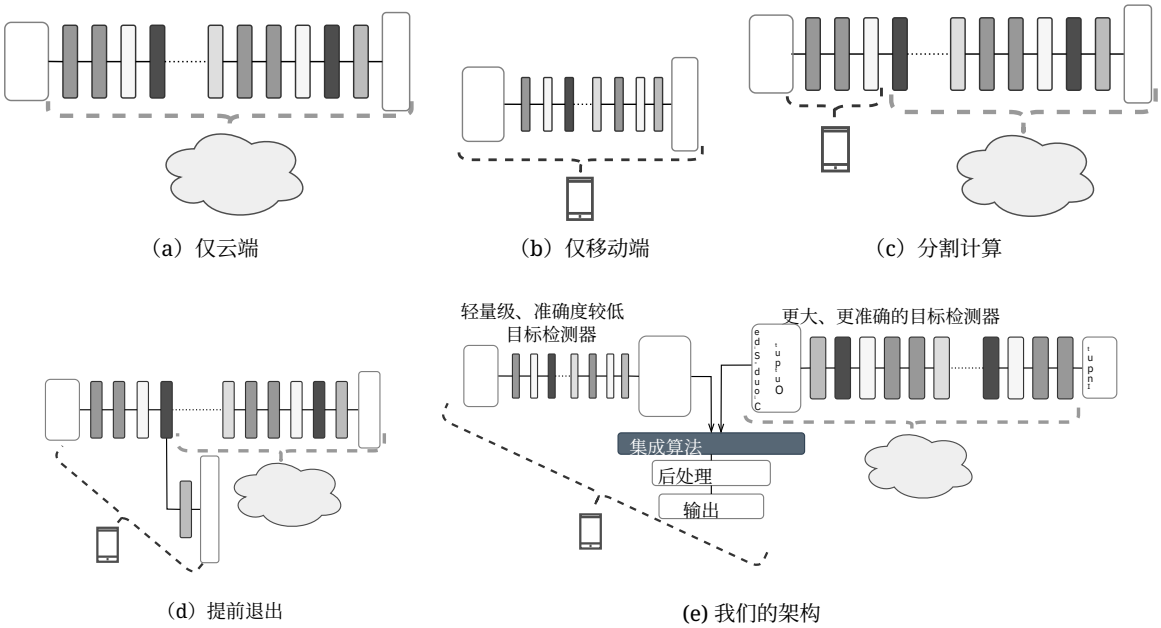
D1	[17], [18], [19], [20], [21]	✗	✓	✓	-	✓	E
D2 A1	[22], [23], [24]	✗	✓	✗	✓	✓	E,C
D2 A2	[25], [26], [27]	✓	✗	✗	✓	?	E,C
D3	[27], [28], [29]	✗	✓	✓	✗	✓	E 或 C
Ours		✓	✓	✓	✓	✓	E,C

*独立于中心云实体。**带宽使用优化。D：方向
A：方法 E：边缘 C：云端



(a) (b) (c) (d) (e) (f) (h) (i) (j)

图 4. 从医疗设备采集的图像示例。



（a）仅云端 （b）仅移动端 （c）分割计算

（d）提前退出 (e) 我们的架构

图 5. 深度学习服务架构。

主要专注于优化响应时间和内存占用，因此会牺牲准确率（图 5(b)）。

第二种方向是将计算垂直分布在边缘和中心云之间（图 5(c)）。一些技术旨在通过在将输入流发送到云端之前，先在边缘丢弃不重要的帧，来减少所需的计算量和带宽；根据任务的性质，这些技术可能采用不同的过滤策略 [22], [23], [24]。还有一些技术将推理模型分布在边缘和云端（分割计算） [25], [26], [27], ，通常以牺牲准确率为代价，换取更短的响应时间或更低的带宽使用量。这些方法依赖中心云，因此在网络中断时便无法使用。

另一种近期出现的方向是在神经网络模型（包括预处理步骤）中确定一个或多个退出点。通常会将退出点设计为至少有一个位于边缘。根据任务及其要求，模型可以提前退出，以牺牲准确率为代价满足延迟约束 [27], [28], [29]。尽管结果颇具吸引力，这类方法仍不够成熟，且仅在分类等简单任务上进行评估（图 5(d)）。

相比之下，我们提出的移动端-云端架构（图 5(e)）的优势在于能够兼取两者之长：得益于集成模型彼此独立的特性，该架构不仅提高了准确率，通常也提升了可执行性和可用性。

3 所提方法

我们的服务使用手机摄像头从医疗设备拍摄图像（图 3(1)），然后执行预处理步骤（图 3(2)）。移动设备在本地运行轻量级推理引擎的同时（图 3(3.2)），将处理好的图像发送到云端（图 3(3.1, 4)）。收到双方的预测结果后，移动设备运行我们的集成算法（图 3(5)）。此外，如有需要，移动设备还会执行后处理步骤，以修正其生成的初始答案（图 3(6)）。如果用户发现预测有误，可以将图像及其对应的真实读数（图 3(7)）发送到云存储，以便进行后续分析和训练迭代（图 3(8)）。移动端模型提供可用性，而云端模型则提升可执行性（参见 §5.5）。两种模型协同工作，可提高准确率。

我们将问题转化为目标检测问题，使感测值中的数字（而非温度、时间和日期等用于表示噪声信息的其他数字）成为感兴趣目标，并通过后处理步骤对目标重新排序，生成最终响应。基于深度神经网络的方法已取得显著成果，但要充分发挥其优势，我们需要一个标注完善的训练数据集，而这在我们的问题中并不可用。为此，我们设计了更完善的工作流程和一种数据生成算法，以自动合成数千张标注对齐良好的训练图像。我们的数据生成工作流程将在第 4 节中详细介绍。

我们使用移动智能手机从医疗设备采集图像并提取有用信息。虽然在移动设备上进行推理能为用户提供低延迟和高可用性，但由于资源限制，通常准确率较低。另一方面，在云端进行推理能够提供准确率和可执行性，但

在网络连接较差时，延迟较高且不可用，这是其主要弱点。如图 3 所示，我们提出的混合移动云服务架构兼具移动端和云计算的优势。此外，我们还设计了一种专用集成模型，以进一步提升所需的性能指标。

3.1 深度学习模型

我们将医疗设备屏幕的基于图像的读数识别问题转化为目标检测和后处理任务。感测值中的每个数字和小数点都占据一个感兴趣区域（RoI），并对应一个类别（图2-1）。在后处理步骤中，我们将一组对象转换为有意义的字符串（图2-2）。对于目标检测部分，我们设计并训练了两个基于单次检测器（SSD）架构的卷积模型（卷积神经网络） [31]。

通常，我们在云端拥有强大的计算资源，但移动端资源有限。因此，我们为SSD架构考虑了两种适配的骨干网络。一种针对智能手机进行了高度优化，参数较少，因此准确率较低。另一种准确率更高，但消耗的资源也更多，无法部署在移动端。我们倾向于两端都采用基于SSD的网络，因为它们能实现更好的端到端延迟 [32]。

3.1.1 云端模型

我们在云端采用 ResNet-50 [33] 作为 SSD 模型的骨干网络。它包含 16 个带有捷径连接的卷积块和一个全连接层，参数超过 $25M$，每个样本需要执行4 0 亿次乘加运算（MAC）。我们移除其最后几层（包括分类头），将其余部分用作 SSD 架构的特征提取器骨干网络。由于输入图像中的感兴趣区域大小不一，我们采用特征金字塔网络（FPN） [34]。它利用现代卷积神经网络（即 ResNet）固有的金字塔结构来生成多尺度特征图。FPN 能够提高模型的准确率，而且与多次输入不同尺寸的图像相比，计算效率更高。

3.1.2 移动端模型

与云端类似，我们将基于 CNN 的分类模型堆叠起来，作为移动端 SSD 架构的特征提取骨干网络。不过，我们在骨干网络和检测头中都采用了适合边缘设备的架构。我们的移动端模型直接受到当前最先进的移动设备检测架构 MobileDets [35] 的启发。之所以使用深度可分离卷积层而非常规卷积层，是因为前者的参数量和 MAC 数更少，同时我们希望这些指标能直接对应更短的训练和推理时间。然而，近期研究 [36], [37] 表明，网络参数量和 MAC 数可能并不能很好地作为模型推理吞吐量和延迟的衡量指标，因为它们并非唯一影响因素。因此，MobileDets 通过加入常规卷积来扩大神经架构搜索空间。经过对不同骨干网络架构进行多次实验后，我们发现， [35]提出的架构最适用于我们的移动端模型。由于移动设备资源受限，我们没有采用特征金字塔网络（FPN）。

3.2 集成算法

如图 3 所示，我们在移动端设计并部署了一个轻量级引擎，同时在云端部署了一个计算量更大、准确率更高的引擎。我们通过集成算法（算法 1）整合两个深度学习模型的预测结果。获得两个模型检测到的物体的感兴趣区域、标签和置信度分数后，集成算法首先在移动端和云端的预测结果中查找对应的感兴趣区域（第 3 行）。这些区域必须具有相同的标签（例如数字“5”），且距离不得超过可容忍值 ϵ。其原因在于，不同目标检测模型预测的感兴趣区域坐标可能并不完全一致。ϵ 控制两个不同模型预测结果之间的最大对应距离。可能存在多个具有相同标签的物体，因此，如果不进行检查，就会导致感兴趣区域误识别。不过，严格比较并不可取，因为两个差异较大的深度学习模型在训练后，其感兴趣区域细化结果也可能存在细微差异。因此，我们加入可容忍距离，使其更为适用。

找到后，如果两个分数都高于指定阈值，集成算法就将它们相加，作为最终置信度分数（第 7 行）；否则，取较高的分数（第 9 行）。如果某个

算法 1集成学习算法

输入： ① 感兴趣区域的最大数量（N）。② 感兴趣区域矩阵（R），以及其对应的置信度向量（ ③）和
④ 标签（L）。 ⑤ 置信度阈值（T）。 ⑥ 可容忍距离（ϵ）。
▷ 每个 R_l^K 由四个顶点组成：左侧（x_{min}），右（x_{max}）、上（y_{min}）、下（y_{max}）。
▷ 上标 C、 M 和 E 分别代表云端、移动端（边缘）和集成。

输出： 与最终预测结果等价的字符串（例如，“10.6”）。

1: $R^E \leftarrow \emptyset,\ \rho^E \leftarrow \emptyset,\ L^E \leftarrow \emptyset$
2: **对于** $i \leftarrow 1$ **到** N **执行**
3: 求 j，使得：
$R_i^C[x_{min}] \approx R_j^M[x_{min}] \pm \epsilon$ **且**
$R_i^C[x_{max}] \approx R_j^M[x_{max}] \pm \epsilon$ **且**
$L_i^C = L_j^M$
4: **若** $j \neq \emptyset$ **则**
5: $L^E \leftarrow L_i^C \cup L^E,\quad R^E \leftarrow R_j^M \cup R^E$
6: **若** $\rho_i^C \geq T_i^C$ **且** $\rho_j^M \geq T_i^M$，**则**
7: $\rho^E \leftarrow (\rho_j^M + \rho_i^C) \cup \rho^E$
8: **else**
9: $\rho^E \leftarrow$ 最大值 $(\rho_j^M, \rho_i^C) \cup \rho^E$
10: **结束如果**
11: $R^M \leftarrow R^M - R_j^M, \rho^M \leftarrow \rho^M - \rho_j^M, L^M \leftarrow L^M - L_j^M$
12: **否则**
13: $L^E \leftarrow L_i^C \cup L^E,\ \rho^E \leftarrow \rho_i^C \cup \rho^E,\ R^E \leftarrow R_i^C \cup R^E$
14: **结束条件判断**
15: **结束循环**
16 $R^E \leftarrow R^M \cup R^E\ \rho^E \leftarrow \rho^M \cup \rho^E\ L^E \leftarrow L^M \cup L^E$:,,:
17 $I \leftarrow \{i \in \mathbb{N} \mid i \leq |\rho^E|,\ \rho_i^E \geq T^E\}$
18 i : $L^E \leftarrow [L_{i \in I}^E \mid \forall (,j)$:
$i < j \Leftrightarrow R_i^E[x_{min}] \leq R_j^E[x_{min}]]$
19: **返回** $\|_{i=1}^{n} L_i^E$ ▷ (‖ 连接 L^E中的每个元素)。

特定对象仅在移动端或云端中的一方有预测结果，则直接将其加入最终结果（第 13、16 行）。第二次筛选发生在第 17 行：所有剩余对象都将与一个更高的通用阈值进行比较。最后，算法根据剩余对象在图像中的位置对其重新排序（第 18 行），并将相应标签串联起来，生成输出字符串（第 19 行）。

3.3 后处理算法

为了进一步移除冗余物体并提升准确率，我们在算法 1 第 18 行之前添加了一个附加后处理算法。我们对非极大值抑制（NMS） [38] 进行了修改，使其适用于我们的问题。在此，我们的目标是找出并移除彼此显著重叠的感兴趣区域。这意味着某个区域内存在多个感兴趣物体，而实际上不应如此。采用 NMS 可以减少假阳性（FP）。修改后的 NMS 定义于算法 2 中，并以迭代方式执行。给定感兴趣区域及其对应的置信度分数和标签后，该算法首先按置信度降序排列感兴趣区域（第 7 行）。然后，它会移除与另一个物体的占用区域重叠程度足够高（$> T_{nms}$）且置信度同时较低的物体（第 13、14 行）。两个物体之间的重叠程度通过其对应感兴趣区域的交并比（IoU）计算。尽管这种方法效果良好，但对所有标签使用同一个阈值会在我们的特定任务中引发问题。表示感测数值小数点的单位标签会与其他物体大幅重叠，但不能将其移除。类别**“1”**和**“7”**也有同样的问题。为避免这一情况，我们在 NMS 过程中单独处理单位标签（第 2–6 行）。

算法 2后处理算法

输入： ① 重叠阈值（T_{nms}）。② 感兴趣区域矩阵（R），以及其 ③ 置信度（ρ）和 ④ 标签对应的向量 (L).

输出： 缩减后的感兴趣区域矩阵，以及其对应的置信度和标签向量。

1: $R^M \leftarrow \emptyset, \quad \rho^M \leftarrow \emptyset, \quad L^M \leftarrow \emptyset$
2: $i \leftarrow j \in \mathbb{N} \mid j \geq |L|, \quad L_j =$'.' ▷('.'≡ 小数点)
3: **如果** $i \neq \emptyset$ **那么**
4: $L^M \leftarrow L_i, \ \rho^M \leftarrow \rho_i, \ R^M \leftarrow R_i$
5: $L \leftarrow L - L_i, \rho \leftarrow \rho - \rho_i, R \leftarrow R - R_i$
6: **结束如果**
7: $(L, \rho, R) \leftarrow [(L_i, \rho_i, R_i) \mid \forall (i, j) : i < j \Leftrightarrow \rho_i \geq \rho_j]$
8: **当** $R \neq \emptyset$ **时执行**
9: $R^M \leftarrow R_1 \cup R^M, \ R \leftarrow R - R_1$
10: $L^M \leftarrow L_1 \cup L^M, \ L \leftarrow L - L_1$
11: $\rho^M \leftarrow \rho_1 \cup \rho^M, \quad \rho \leftarrow \rho - \rho_1$
12: **对于** $i \leftarrow 1$ **到** $|R|$ **执行**
13: **如果** 交并比$(R_i, \ R_1^M) \geq T_{nms}$ **则**
14: $R \leftarrow R - R_i, \quad L \leftarrow L - L_i$
15: **结束如果**
16: **结束循环**
17: **结束 while 循环**
18: **返回** $R^M \cup R, \ L^M \cup L, \ \rho^M \cup \rho$

3.4 带宽优化

我们的目标使用场景是在欠发达国家，因此用户往往可能位于某些区域或处于无法访问互联网，或连接质量较差的情形，例如可用带宽有限或存在网络拥塞问题。因此，我们更倾向于以较低的响应准确率为代价，优先提供最高的可用性，从而总体上提高可执行性。在无连接的情况下，用户仍然可以获得移动端的答案。不过，用户会经历准确度下降。对于网络状况较差的情况，我们提出了一种简单的图像压缩技术，在发送捕获的图像时减少带宽使用量（算法3）。我们会缩小图像尺寸，将其从 RGB 变换为 HSV色彩空间，然后仅将图像的数值（V）通道发送到云端。在云端，使用预定义常数填充 H 和 S，并在执行推理之前将其放大至原始尺寸。

算法3简单有损压缩

输入： ① Img ② 输出大小（$\mathcal{H}\nu, \mathcal{W}_\nu$） ③ 填充值（$\mathcal{K}$）

输出： 在移动端：$Img_{comp.}$ 在云端：$Img_{decomp.}$

在移动端

1: $Img_{comp.} \leftarrow$ 调整大小 Img 至（$3 \times \mathcal{H}\nu \times \mathcal{W}\nu$）。2: 变换 $Img_{comp.}$ 从 RGB 色彩空间至 HSV。3: $Img_{comp.} \leftarrow$ 删除通道 H （色调）和 S （饱和度）。4:**返回** $Img_{comp.}$

在云端

5: $Img_{decomp.} \leftarrow$ 添加通道 H 和 S，并将像素填充为$\mathcal{K}$。6: 调整大小 $Img_{decomp.}$ 至 Img的原始尺寸。7:**返回** $Img_{decomp.}$

3.5 所提算法的复杂度

对于每个检测对象，我们都会遍历其他对象。同样，在后处理算法中也会再次执行此操作，但对象数量更少。因此，我们算法的时间复杂度为 $O(N^2+N'^2)$，其中 N 是预测的感兴趣区域的数量，N' 是需要后处理的对象数量。由于$N'<N$，复杂度可写为 $O(N^2)$。我们使用一维数组存储感兴趣区域框、置信度向量和标签向量，其大小取决于 N。因此，该算法的空间复杂度为 $O(N)$。

4 数据生成

基于图像的读数识别中的数据稀缺，是以往研究结果不佳的一个主要原因。深度学习模型（例如卷积神经网络）的强大之处，在于能够直接从数据中学习表征。深度学习算法要么依赖大量标注数据以监督方式进行训练，要么依赖在相关任务上预训练的模型；而我们的任务两者皆无。

相关工作通常采用传统的计算机视觉流程，首先进行数据分析，随后依次进行数据清洗、预处理、特征提取、特征选择，最后设计并部署合适的模型（图 6(a)）。相比之下，我们提出了一种不同的流程（图 6(b)），使我们能够利用深度学习模型，其性能大幅超过以往的算法。

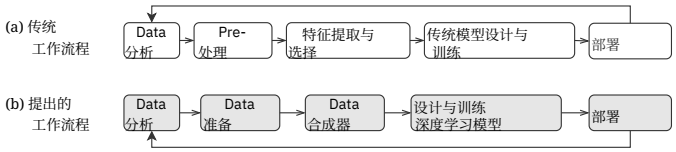


图 6。（a）传统的计算机视觉工作流程。（b）提出的工作流程。请注意，尽管两者中的某些模块相同，但其内部流程可能完全不同。

4.1 数据准备

分析从各种血糖仪显示屏拍摄的图像后，我们发现了一些常见的差异特征，汇总于表 3。我们发现，许多问题都源于环境条件。例如，如图 4 所示，有些图像存在旋转（a、b、d、f、g）、拍摄视角不同（a、b、d、e、f）、测量值与背景之间对比度较低（a、e、f）、闪光灯反射（h），以及失真和／或模糊（e、h）等情况。表 3 中的一些模式也可以观察到。例如，图像中可能包含图表（d）、存在背光（c、d、g），或屏幕形状不规则（g、h）。此外，几乎所有图像的背景颜色、显示字体样式和类型等都各不相同。

医疗设备的多样性以及环境变化给以特征工程为基础的现有方法带来了诸多困难。相反，我们利用这些特性，合成带有标注的大型训练数据集。我们通过互联网收集了 100 张不同医疗设备的图像和 150 种开源字体（包括七段码、点阵、液晶显示器和发光二极管样式）。随后，我们手动变换所有图像，使其具有标准视角。我们还确定了每张图像中对应于显示屏角点以及屏幕上可能出现的不同项目（例如感测值、测量单位、日期等）的~20 个不同点坐标。最后，我们将所有这些初步信息输入数据合成器 算法，以合成几乎无限多张具有不同感测值的医疗设备图像。

4.2 数据合成器

我们的数据合成器（算法4）接收上一步准备好的数据（第1行），即若干张转换后的图像（$\mathcal{I}mages$）及其对应的点坐标（$\mathcal{A}$）、一组字体（$\mathcal{F}$）、一个自由度（$\mathcal{D}$），用于显示屏上的每个项目及其属性，以及要合成的最大图像数量（$\mathcal{N}$）。随后，它按照算法4生成新图像（第4行）和严格的标注（第5行）。从较小的样本量生成数千张图像，尤其是为了训练深度学习模型，是一项重要的流程，因为训练数据中的任何细微噪声或分布偏移都会导致过拟合问题：我们的模型在合成训练数据上可能达到合理的准确率，但在真实世界的测试数据上表现不佳。因此，所有人工生成的图像都必须与真实图像十分相似。另一方面，为了训练深度学习模型，训练集应包含足够多的区别性特征。此外，我们的算法还必须生成对齐良好的标注。

表3 医疗设备基于图像的读数中常见的差异属性和干扰因素。

#	属性	模式
1	显示屏	不同的宽高比
2		不同的屏幕形状
3		液晶显示器、发光二极管、七段显示、点阵
4		带背光或不带背光
5		规则或不规则屏幕
6		圆角或尖角
7	数值	不同的测量单位（$mg/dL, mmol/L$）
8	（标签）	通常是显示屏上最大的项目
9		在不同显示屏上没有特定位置
10	附加信息 $Items$	包括日期、时间、各种标志和符号、测量单位、图表等。
11		在不同显示屏上的位置不固定
12		在某些情况下，字体、样式和颜色与数值

算法4数据合成器

输入： ① 源图像 $\mathcal{I}mages$, ② 对应坐标-定的 $\mathcal{A}$, ③ 字体集合 $\mathcal{F}$, ④ 自由度 $\mathcal{D}$，以及 ⑤ 要生成的最大样本数 $\mathcal{N}$。

输出： 由 $\mathcal{N}$ 个元组组成的集合，每个元组包含一张合成图像，其标签（数值）以及对应的感兴趣区域（R^{Value},R^{Items}）。

1: **过程** 合成器($\mathcal{I}mages^{\mathcal{S}et}, \mathcal{A}, \mathcal{D}, \mathcal{F}, \mathcal{N}$)
2: **对** $\iota \leftarrow 1$**到** $\mathcal{N}$ **执行**
3: **重复**
4: (Img, R^{Screen}) ← 样本 $_{Rand}$(($\mathcal{I}mages^{\mathcal{S}et}$, $\mathcal{A}$))
5: (数值 $Items$) ←生成$_{Rand}$(Img, $\mathcal{A}_{Img}$, $\mathcal{D}_{Img}$, $\mathcal{F}$)
6: ∀项目 $j \in Items$ ∀对象 $k \in$ 项目 j：计算($R^{Items}_{j\ k\ ,}$)。
7: ∀ 对象 $i \in$ 数值：计算(R_i^{Value})。
8: **直到** $R_i^{\text{Value}} \cap R^{Items}_{j\ ,\ k} = \emptyset$ **并且** 交并比(R_i^{Items}, R_j^{Items}) $< \epsilon,\ i \neq j$
9: **重复**
10: Img ←变换(Img, $\mathcal{S}et$)
11: $\forall R_i \in R^{\text{数值}}$ ： 计算(R_i) ▷（重新计算感兴趣区域。）
12: **直到** $\forall R_i \in R^{\text{Value}}$： $R_i \cap (\mathbb{U} - R^{Screen}) = \emptyset$
13: **存储** $\{Img, R^{\text{Value}}$, 数值 $R^{Items}\}$ ▷数值 ≡标签 L_{Img}
14: **结束循环**
15: **结束过程**

我们使用 $\mathcal{D}$，确保合成图像及其对应的标注既与真实捕获的图像相似，又符合表 3 中描述的模式。我们在合成过程中引入受控随机性。随机性是增加训练样本之间差异、防止过拟合并提高最终准确率的关键。此外，我们的算法会随机选择或生成不同的字体样式、字体大小、颜色、背景光照、时间、日期以及不同的指示符（例如温度或血滴符号）（第 5 行）。附加信息的字体样式、字体大小、位置和颜色也会从一定范围内随机选取。随后会验证每个项目的位置（第 8 行）。屏幕上的感测数值与其他附加信息（例如时间、日期、温度等）之间不得有任何重叠（$Items$）。此外，两个不同的 $Items$之间也不应有明显的（$> \epsilon$）交集。

因此，它会调用一个变换过程（第10行），对每张生成的图像及其标注进行修改。我们的变换过程主要分为两个部分：几何和视觉。在几何部分，我们进行缩放、裁剪、旋转、错切、透视变换和平移。在视觉部分，我们会修改颜色、对比度、亮度和锐度。我们还会添加噪声，并随机移除一些像素，以模拟

显示屏上的光线反射或划痕。每种变换的范围和发生概率取决于生成集合（$\mathcal{S}et$）。通常，我们在训练集上采用更大的范围和更高的发生概率，而在验证集上则相反。其背后的考虑是，我们希望模型在训练时具备更好的泛化能力，而验证时则需要使用一组与真实世界图像更为相似的数据。请注意，我们绝不会在测试集上使用我们的数据合成器。此外，为防止任何有意义的信息泄漏，我们确保测试集中的医疗设备与训练集/验证集中的医疗设备不相似。应用变换后，数值必须仍处于$Screen$ 感兴趣区域（RoI）（R^{Screen}）内。否则，我们会再次应用变换，直至满足条件。图7展示了仅使用来自 $\mathcal{I}mages$的一张图像生成的几个合成样本。

数值 在合成图像中的范围为 0 至 1000，且服从离散伪独立同分布。其中一半为整数，35% 为一位小数（例如 12.5），其余为两位小数（例如 1.25）。生成的测量单位必须与合成数值相兼容。背光颜色和显示背景颜色并非均匀随机生成；相反，我们的算法会优先选择在真实设备上更常见的颜色，但也会以一定概率生成全新的颜色，使模型能够适应新设备或离群值。每个项目 $\in Items$ 可能会在 $Screen$上随机出现或消失。 $\mathcal{I}mages$ 分为训练 ($\mathcal{I}mages^{train}$) 和验证 ($\mathcal{I}mages^{val}$) 两部分，比例分别为 90% 和 10%。每张图像都包含一台不同的医疗器械。

5 评估

5.1 实验设置

训练数据集。我们使用数据合成器生成了 $1M$ 个各不相同且标注完善的训练样本。图像尺寸会直接影响训练吞吐量、推理延迟、检测模型的大小以及存储数据集所需的空间。因此，我们将每张图像缩小至 3×320^2，并以JPEG格式保存。请注意，我们的卷积神经网络模型输入尺寸各不相同。生成并存储 $1M$ 个样本需要占用 $45GB$ 磁盘空间，耗时 ∼11小时。

测试数据集。我们从客户使用的各种血糖仪设备直接采集了300张图像，除缩小至所需尺寸外，未进行任何修改。如图8所示，类别“1”出现频率最高，而类别“.”的出现频率最低。此外，我们还在牛津大学CameraLab公开提供的另一数据集上评估了我们的方法（以下简称牛津数据集） [6]。

训练。我们在一台服务器上训练了移动端和云端的神经网络模型。我们使用自建的GPU服务器（表6）训练模型，并通过TensorFlow框架设计和训练模型。通过迁移学习，利用在COCO数据集上预训练的权重[39]，我们减少了训练模型所需的时间和资源。这有助于模型提取更优的低级特征。我们对每个模型训练了50万至60万轮（∼3.5 天），并在训练期间采用简单的数据增强以防止过拟合。

推理设置。对于概念验证，我们使用相同的服务器作为云端。移动端使用 Galaxy Tab A（2019），这是一款性能一般的平板电脑（表 5）。我们借助 Android 版 TensorFlow Lite 转换并优化移动端模型，并测量其性能。推理过程中，输入图像会分别调整为 3×416^2 和 3×350^2，以供云端模型和移动端模型使用。

5.2 准确率指标

我们按照公式 (1) 报告算法的准确率。与此前报告的分类和定位指标（即精确率、召回率和 F1分数）相比，该指标更具说明性。不过，它也是最严格的指标；鉴于所读取数值的医疗属性，这种严格性是必要的。

给定预测结果 Y及其真实标注 Λ。令:

$$\hat{Y} := \hat{y_1}\hat{y_2}\cdots\hat{y_m}$$
$$\Lambda := \lambda_1\lambda_2\cdots\lambda_n$$

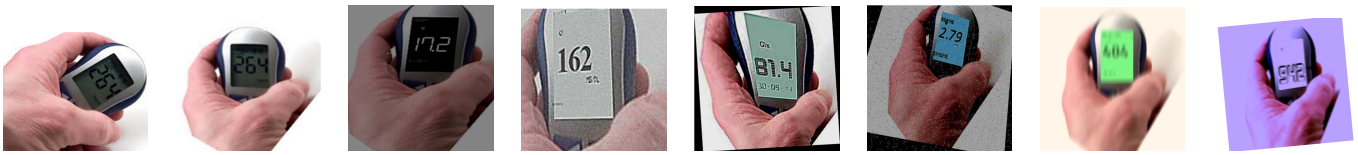
Y在满足以下条件时被视为正确预测:

$$m = n \quad \text{and} \quad \forall i : \hat{y_i} = \lambda_i,\ i \in \mathbb{N},\ i \leq n \tag{1}$$

只有当 ① 所有真实标注的感兴趣区域均被正确检测， ② 标签预测正确， ③ 对象的顺序与真实标注中的顺序一致，且 ④ 未检测到任何附加对象（例如来自 $Items$的对象）时，预测才是正确的。例如，假设真实标注的标签为 *11.9* （图 2），那么 119、1.19、111、1149、1109、11、19、1.9 等都不是正确的预测，因此该指标比精确率或召回率更难达到。

5.3 我们测试集上的准确率

我们首先使用客户从各种血糖仪设备直接拍摄的300张图像评估算法，除缩小到所需尺寸外，未对图像进行任何修改。如图8所示，类别‘1’出现频率最高，而类别‘.’的出现频率最低。尽管云端和仅移动端的预测准确率均在 $89-90\%$ 附近波动（图9），我们的集成模型仍将准确率提高了7.3 个百分点。其原因如图10所示。在每个混淆矩阵中，每列表示预测为某一类别的物体，每行则表示实际属于某一类别的物体。为了使矩阵包含更多信息，我们增加了一行和一列。最后一列表示属于某个真实类别但被忽略的物体，最后一行表示检测到但不属于任何类别的物体。为便于观察，每个矩阵都按行归一化。除‘.’和‘1’外，我们的云端模型在大多数类别上表现良好。另一方面，我们的移动端模型在‘0’、‘4’和‘9’上的表现较差，但对‘.’和‘1’的预测远优于云端模型。最后一个矩阵证实，我们的集成算法优于单个模型，也减少了假阳性和假阴性。误预测图像中通常会出现屏幕上的一些特殊物体或符号。例如，图12(f)所示的设备屏幕上显示了一个与‘7’十分相似的箭头，有时会误导模型。



(a) (b) (c) (d) (e) (f) (g) (h)

图7. 算法4合成并变换的图像示例。(a)为原始图像，(b)为其手动变换后的图像。请注意，为便于理解，此处仅展示由一张源图像生成的图像。

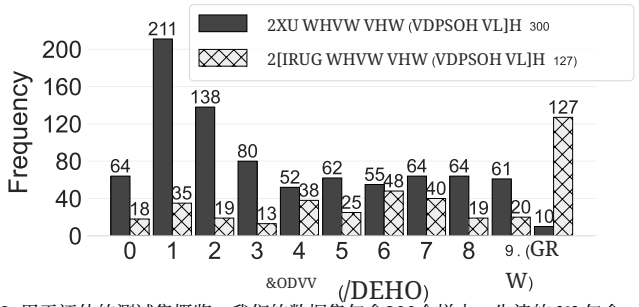


图8. 用于评估的测试集概览。我们的数据集包含300个样本，牛津的 [6] 包含127个。

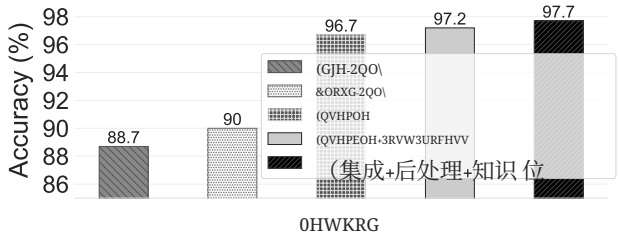


图9. 不同设置下我们测试集上的准确率。

5.3.1 后处理的影响

我们的后处理算法对测试样本的预测产生了积极效果。在测试集上，它使准确率提高了0.5 至1 （图9）。假阳性错误减少的主要原因是在集成步骤之后进行了后处理（图10）。

5.3.2 设备感知预测（知识注入）

在我们所提方法中，准确率最低的预测属于“.”（小数点）类别，准确率为90% 。当前最先进水平 [6] 仅检测物体，并手动添加小数点。这是因为它们的数据集包含两种设备，二者均使用 $mmol/L$ 作为测量单位，也就是说，所有数值均为浮点数。相比之下，我们的数据集更具通用性，也包含 mg/DL。通过注入关于医疗器械测量单位（mg/DL 或 $mmol/L$）的一位先验知识，可以提高算法的准确率。如图9所示，准确率提高了0.5。不过，在未来的工作中，可以将屏幕上显示的测量单位视为另一个感兴趣物体。

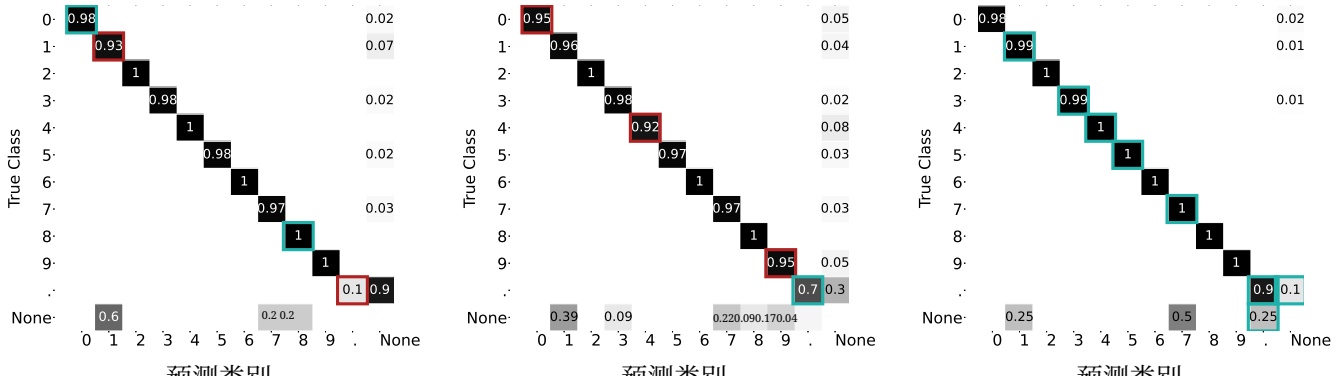
5.4 泛化能力：牛津数据集上的准确率

牛津数据集 [6]为我们提供了评估所提算法鲁棒性和泛化能力的机会。我们还可以直接与当前最先进的方法 [6]进行比较。该数据集包含两种不同的血糖仪设备，每种设备的数据均平均划分为训练集和测试集。两个数据集相当相似（包括图像采集设备），除了屏幕上显示的数值不同。每个数据集包含127个样本，所有样本的七段显示器均以毫摩尔/升为测量单位（图8）。每张图像的尺寸为 $\sim 3\times 2600\times 4600$ 像素。我们首先报告模型在测试集上的准确率，且不使用训练集进行学习，以衡量我们方法的鲁棒性。在这种设置下，我们取得了**89.8%** 的准确率。接下来，为了进行公平比较，我们使用训练集对模型进行了微调。在这两种情况下，均未采用知识注入。除 T^E 外，两个数据集评估期间的所有参数均保持不变；牛津数据集使用0.72 ，我们的数据集使用0.82。为避免类别不平衡问题，我们将牛津训练集与我们自己训练集中的一小部分数据合并。微调后，准确率提升至**92.1%**，而之前的方法达到 51.5% [6]。结果表明，无论是（+40.6%）还是不（+38.3%）使用训练集， 我们的算法都优于最先进的方法。详细评估显示，在大多数误读图像中，设备离摄像头较远，因此有时难以正确读取。主要原因是我们的模型分辨率较低。输入分辨率会显著影响感兴趣区域检测的准确率[32]。一般而言，将图像尺寸缩小4倍会使准确率降低近16% ，并将响应时间缩短 $\sim$27%。我们的模型输入分辨率为 <1牛津数据集中图像分辨率的% 。因此，增大神经网络模型的输入尺寸或许能进一步降低错误率，但代价是训练时间和响应时间更长。移动端所需的内存和存储空间也是必须纳入考虑的限制因素。

5.5 可用性与可执行性

我们的移动设备能够分别在启用图形处理器和仅使用中央处理器的设置下，于 $260ms$和 $360ms$ 内完成预测。即使没有互联网连接，用户仍可依靠智能手机上的结果。虽然准确率会有所下降，但可用性更高。连接质量较差时，用户可以启用我们简单的压缩算法。该算法可将带宽使用量减少 **45**× ，准确率仅下降% ，或者将带宽使用量减少 90× ，准确率损失为 $\sim$2% 。这是通过我们的数据合成器 算法实现的。该算法促使我们的深度学习模型学习信道不变特征。此外，集成方法本身也能提升鲁棒性。图 11 所示的响应时间分解表明，应用我们的压缩技术后，对总延迟贡献最大的传输延迟大幅缩短。因此，即使带宽仅为 $512Kbps$ 、往返时间为 $100ms$，用户仍能在 $<500ms$ 内获得合理的响应。

为比较相关服务架构（图 5），我们假设一个服务级别协议，其中 可执行性被描述为达到



(a) 云端模型 (b) 移动端模型 (c) 集成模型

图 10. 不同模型的混淆矩阵。矩阵按行归一化。

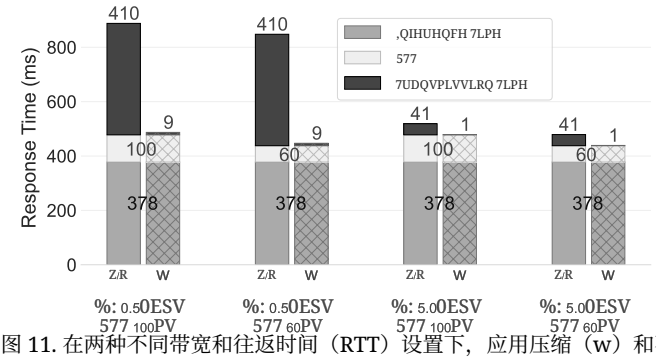


图 11. 在两种不同带宽和往返时间（RTT）设置下，应用压缩（w）和不应用压缩（w/o）时的云端响应时间分解。RTT 是往返时间，可通过命令 *ping* 测量。此处假设 RTT 值为 100ms 和 60ms 以作说明。

L_k 准确率，同时能够在 $5s$ 内准备好响应。可用性可通过在 $<500ms$ 内响应来实现，将其视为软截止时间。这里，L_1 (✓✓) 和 L_2 (✓) 分别代表 $>90\%$和 $>85\%$ 的准确率。我们在三种不同的连接质量下评估每项服务：极佳、较差（例如，由于拥塞或带宽受限）以及无连接（没有网络连接）。如表 4 所示，我们的模型在这三种情况下的表现都更好。据我们所知，目前没有可比的分割计算或提前退出方法能够在几乎不损失准确率的情况下，显著压缩其中间数据（第 4 行和第 6 行）。

6 结论

我们提出了一种基于移动云的自动化图像识别血糖仪读数系统，可广泛应用于各种医疗设备；我们的方法利用手机的摄像头、无线通信和计算能力，为在发达国家更容易获得的昂贵设备提供了一种低成本替代方案

表 4 不同方法和连接（Conn.）下可用性（Avail.）和可执行性（Perf.）概览。

服务	连接良好		连接较差		无连接	
	可用性	可执行性	可用性		可用性	
仅移动端	✓✓	✓	✓✓	✓	✓✓	✓
仅云端	✓✓	✓✓*	⇓	✓✓*	✗	✗
分割计算	✓✓	✓✓*	✗	✗	✗	✗
分割计算+	✓✓	✓✓*	⇓	✓✓*	✗	✗
提前退出	✓✓	✓✓*	✓✓	✗或✓	✓✓	✗或✓
提前退出+	✓✓	✓✓*	✓✓	✓✓*	✓✓	✗或✓
Ours	✓✓	✓✓	✓✓	✓✓	✓✓	✓

+ 假设中间数据采用无损方法进行压缩。⇓ 明显的性能下降。* 会产生额外开销。

。我们的深度学习集成算法与移动云服务架构相结合，与仅移动端、仅云端、分割计算和提前退出等竞争模型相比，具有更高的可用性和性能。我们提出了一种数据生成算法来解决数据稀缺问题，并合成了一百万个标注完善的样本。值得注意的是，血糖仪设备及其屏幕输出种类繁多，通常会给现有技术的适用性带来挑战；但我们反而利用这些多样性，通过数据生成技术从现有照片中生成更多训练样本，从而获得更稳健的模型。

我们的方法即使在昏暗环境、屏幕反光、照片模糊或失焦等不完美条件下，也能够正确读数。我们的集成算法能够高效地融合两个独立深度神经网络模型得出的结果。具体而言，我们会考虑这两个模型识别出的边界框之间的细微偏移，以及“.”符号这一特殊情况：与其他符号不同，它可能与其他检测对象大幅重叠。

我们在两个不同的真实场景测试集上评估了方法的准确率，一个由我们的用户收集，另一个来自 Camera Lab [6]。我们的方法在自己的测试集上取得了97.7% 的准确率，在牛津测试集上取得了92.1% 的准确率，表现优于当前最先进水平。结果表明，我们的算法具有鲁棒性，并且能够很好地泛化到其他数据集。

参考文献

[1] J. E. Given、M. J. O’Kane、B. P. Bunting 和 V. E. Coates，《糖尿病患者自填血糖日记记录与仪表存储器的比较：系统综述》，《糖尿病医学》，第 30 卷，第 8 期，第 901–913 页，2013 年。[2] D. Salvi、C. Velardo、L. Mackillop 和 L. Tarassenko，《妊娠糖尿病患者报告的血糖日记记录与仪表存储器的算法比较》，《医学信息学解锁》，第 20 卷，文章编号 100397，2020 年。[3] “idia。” [在线]。网址：https://idia.app/[4] “连续血糖监测：权衡利弊。” [在线]。网址：https://www.verywellhealth.com/continuous-glucose-monitoring-the-arrival-of-dexcom-5 3289566 - “血[5] [ ] 糖仪指南。”在线。网址：https://diabetes.co.uk/diabetes care/blood glucose monitor guide.html [6] E. Finnegan、M. Villarroel、C. Velardo 和 L. Tarassenko，《一种从血糖仪和血压监测仪的图像中检测并读取七段数字的自动化方法》，《医学工程与技术杂志》，第 43 卷，第 6 期，第 341–355 页，2019 年。

[7] W. Shi、J. Cao、Q. Zhang、Y. Li 和 L. Xu，《边缘计算：愿景与挑战》，《*IEEE*物联网期刊》，第 3 卷，第 5 期，第 637–646 页，2016 年。[8] T. Morris、P. Blenkhorn、L. Crossey、Q. Ngo、M. Ross、D. Werner 和 C. Wong，《ClearSpeech：面向视障人士的显示屏阅读器》，《*IEEE*神经系统与康复工程汇刊》，第 14 卷，第 4 期，第 492–500 页，2006 年。[9] R. P. Ghugardare、S. P. Narote、P. Mukherji 和 P. M. Kulkarni，《测量仪器七段显示图像的光学字符识别系统》，载于《*TENCON 2009——2009*年*IEEE*第十区会议》。IEEE，2009 年，第 1–6 页。[10] M. Mariappan、V. Ramu、T. Ganesan、B. Khoo 和 K. Vellian，《基于 LabView 的 OTOROB 虚拟医疗仪器：利用光学字符识别获取多种医疗仪器的液晶读数》，载于《国际生物医学工程与技术会议论文集（*IPCBEE*）》，第 11 卷，2011 年，第 70–74 页。[11] S. Ghosh 和 S. Shit，《一种利用图像处理技术从数字显示仪器采集数据的低成本系统》，载于《*2014*年国际计算、通信与信息学进展会议（*ICACCI*）》。IEEE，2014 年，第 1065–1068 页。[12] E. Tekin、J. M. Coughlan 和 H. Shen，《面向视障人士的 LED/LCD显示屏实时检测与读取》，载于《*2011*年*IEEE*计算机视觉应用研讨会（*WACV*）》。IEEE，2011 年，第 491–496 页。[13] I. Rasines、P. Iriondo 和 I. D´ıez，《面向视障人士的实时显示屏识别系统》，载于《残障人士计算机国际会议》。施普林格，2012 年，第 623–629 页。[14] “Tesseract开源光学字符识别引擎。” [在线]。网址：https://github.com/tesseract-ocr/tesseract [15] M. K. Prakruthi、V. Kalyan、V. Suhas、M. G. Katti 和 S. Pai，《卷积神经网络在移动设备中的应用：推断医疗器械读数》，《国际研究与科学创新期刊》，第 IV 卷，第 VI 期，第 2321–2705 页，2017 年。[16] C. Liu，《医疗设备数字识别》，学位论文，西安大略大学，2016 年。[17] M. Sandler、A. Howard、M. Zhu、A. Zhmoginov 和 L.-C. C. Chen，《MobileNetV2：倒置残差与线性瓶颈》，《*IEEE*计算机学会计算机视觉与模式识别会议论文集》，第 4510–4520 页，2018 年 1 月。[18] X. Dai、P. Zhang、B. Wu、H. Yin、F. Sun、Y. Wang、M. Dukhan、Y. Hu、Y. Wu 和 Y. Jia，《ChamNet：通过平台感知模型适配实现高效网络设计》，载于《*IEEE*计算机视觉与模式识别会议论文集》，2019 年，第 11398–11407 页。[19] A. Howard、M. Sandler、G. Chu、L.-c. Chen、B. Chen、M. Tan、W. Wang、Y. Zhu、R. Pang、V. Vasudevan、G. Chu、L.-c. Chen、B. Chen 和 M. Tan，《搜索 MobileNetV3：准确率、乘加运算次数与模型大小的权衡》，载于《国际计算机视觉会议》，2019 年，第 1314–1324 页。[20] M. Tan、B. Chen、R. Pang、V. Vasudevan、M. Sandler、A. Howard 和 Q. V. Le，《MnasNet：面向移动端的平台感知神经架构搜索》，载于《*IEEE*计算机视觉与模式识别会议论文集》，2019 年，第 2820–2828 页。[21] M. Rastegari、V. Ordonez、J. Redmon 和 A. Farhadi，《借助 XNOR网络在边缘端实现人工智能》，《*ACM*通讯》，第 63 卷，第 12 期，第 83–90 页，2020 年。[22] Y. Li、A. Padmanabhan、P. Zhao、Y. Wang、G. H. Xu 和 R. Netravali，《Reducto：面向资源高效实时视频分析的摄像机端过滤》，载于《*ACM*数据通信特别兴趣小组关于计算机通信的应用、技术、架构与协议的年会论文集》，2020 年，第 359–376 页。[23] K. Hsieh、G. Ananthanarayanan、P. Bodik、S. Venkataraman、P. Bahl、M. Philipose、P. B. Gibbons 和 O. Mutlu，《Focus：以低延迟和低成本查询大型视频数据集》，载于《第 *13* 届 *USENIX* 操作系统设计与实现研讨会（*OSDI 18*）》，2018 年，第 269–286 页。[24] J. Wang、Z. Feng、Z. Chen、S. George、M. Bala、P. Pillai、S.-W. W. Yang 和 M. Satyanarayanan，《借助边缘计算实现无人机带宽高效的实时视频分析》，载于《*2018*年*IEEE/ACM*边缘计算研讨会（*SEC*）》。IEEE，2018 年，第 159–173 页。[25] Y. Kang、J. Hauswald、C. Gao、A. Rovinski、T. Mudge、J. Mars 和 L. Tang，《Neurosurgeon：……之间的协同智能

云端与移动边缘，” *ACM SIGARCH*《计算机体系结构新闻》，第45卷，第1期，第615–629页，2017年。[26] D. Hu 和 B. Krishnamachari，“借助边缘通信压缩实现快速、准确的流式卷积神经网络推理”，载于*2020*年 *IEEE/ACM*第五届物联网设计与实现国际会议（*IoTDI*）。IEEE，2020年，第157–163页。[27] S. Laskaridis、S. I. Venieris、M. Almeida、I. Leontiadis 和 N. D. Lane，“SPINN：设备与云端协同的神经网络渐进式推理”，载于第*26*届移动计算与网络年度国际会议论文集，2020年，第1–15页。[28] S. Teerapittayanon、B. McDanel 和 H.-T. T. Kung，“云端、边缘和终端设备上的分布式深度神经网络”，载于*2017*年 *IEEE*第*37*届分布式计算系统国际会议（*ICDCS*）。IEEE，2017年，第328–339页。[29] L. Zhang、Z. Tan、J. Song、J. Chen、C. Bao 和 K. Ma，“SCAN：面向紧凑高效模型的可扩展神经网络框架”，载于神经信息处理系统进展，2019年，第4027–4036页。[30] 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论文集，2017年，第7263–7271页。[31] W. Liu、D. Anguelov、D. Erhan、C. Szegedy、S. Reed、C.-Y. Fu 和 A. C. Berg，“SSD：单发多框检测器”，载于欧洲计算机视觉会议。施普林格，2016年，第21–37页。[32] J. Huang、V. Rathod、C. Sun、M. Zhu、A. Korattikara、A. Fathi、I. Fischer、Z. Wojna、Y. Song、S. Guadarrama 和 K. Murphy，“现代卷积目标检测器的速度/准确率权衡”，*IEEE*计算机视觉与模式识别会议，*2017*年计算机视觉与模式识别会议，第3296–3305页，2017年。[33] K. He、X. Zhang、S. Ren 和 J. Sun，“用于图像识别的深度残差学习”，载于*IEEE*计算机视觉与模式识别会议论文集，第2016年12月卷，2016年，第770–778页。[34] T.-Y. Lin、P. Doll´ar、R. Girshick、K. He、B. Hariharan 和 S. Be-longie，“用于目标检测的特征金字塔网络”，载于*IEEE*计算机视觉与模式识别会议论文集，2017年，第2117–2125页。[35] Y. Xiong、H. Liu、S. Gupta、B. Akin、G. Bender、Y. Wang、P.-J. Kindermans、M. Tan、V. Singh 和 B. Chen，“MobileDets：为移动加速器搜索目标检测架构”，载于*IEEE/CVF*计算机视觉与模式识别会议论文集，2021年，第3825–3834页。[36] V. Sze、Y.-H. Chen、T.-J. Yang 和 J. S. Emer，“深度神经网络的高效处理”，计算机体系结构综合讲座，第15卷，第2期，第1–341页，2020年。[37] D. Stamoulis，“面向高效深度学习应用的硬件感知自动机器学习”，2020年。[38] A. Neubeck 和 L. Van Gool，“高效的非极大值抑制”，载于第*18*届国际模式识别会议（*ICPR'06*），第3卷。IEEE，2006年，第850–855页。[39] T.-Y. Lin、M. Maire、S. Belongie、J. Hays、P. Perona、D. Ramanan、P. Doll´ar 和 C. L. Zitnick，“Microsoft COCO：上下文中的常见物体”，载于欧洲计算机视觉会议。施普林格，2014年，第740–755页。

附录 A 移动设备和云规格

表5 移动设备规格

三星 Galaxy Tab A 8.0 with S Pen（2019年，SM-P205）	
CPU	八核 big.LITTLE （2×1.8GHz Cortex-A73 和 6×1.6GHz Cortex-A53）
芯片组	Exynos 7 Octa 7904（14 nm）– 64 位
GPU	ARM Mali-G71 MP2
内存	3GB LPDDR4X
OS	Android 10
网络	2G/3G/4G（LTE）和 Wi-Fi 802.11

表 6
云服务器规格

CPU	Intel Xeon E5-2630-v3（x86 64） 频率：2.4GHz (1.2-3.2GHz) 缓存：32K-32K-256K-20480K #核心：24（双线程）
GPU	NVidia GeForce GTX 1080 Ti 总内存：11GB CUDA V9.0.176，计算能力 6.1
内存	32GB (1600MHz)
OS	Gnu/Linux Ubuntu 18.04

附录 B 响应时间

表 7 显示了在不同设置下，移动端和云端的总响应时间。对于移动端推理，使用了适用于 Android 的 TensorFlow 基准测试工具；对于云端，我们测量了从发送 REST 请求到服务器并收到响应所需的时间。表 4 中报告的云端响应时间，是推理时间、典型传输时间（按 5Mbps 带宽计算）和典型往返时间（RTT 等于60ms）之和。

表 7 不同设置下的响应时间。

引擎	平台	响应时间（毫秒）
移动端	中央处理器（1 个线程）	494 ± 1
	中央处理器（4 个线程）	360 ± 25
	图形处理器（1 个中央处理器线程）	$\mathbf{260 \pm 1}$
云端	CPU	$\mathbf{479 \pm 20}$

附录 C 带宽优化

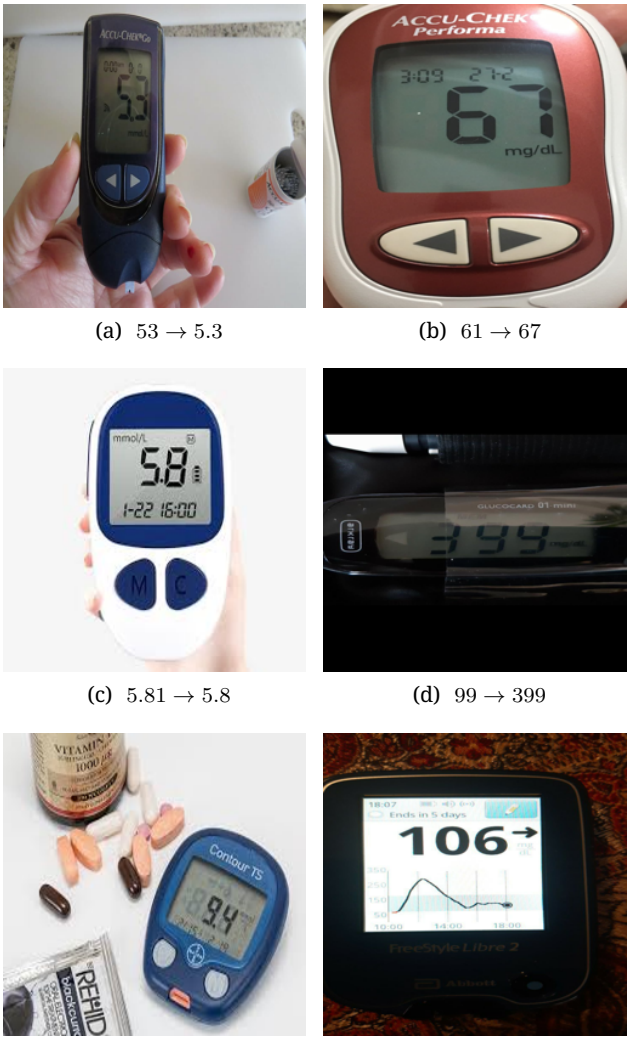
表 8 展示了我们的压缩方法对不同输入分辨率下带宽和准确率的影响。

表 8 带宽使用量降低与准确率（%）

方法	原始 RGB		我们的压缩	
64×64	93.0%	30×	95。3%	90×
64×128	95.7%	15×	96.7%	45×
128×128	95.7%	7.5×	97%	22.5×
256×256	97.2%	2.5×	97.2%	5.5×
350×350	97。2%	1.0×	97。2%	3.0×

附录 D 误识别示例

我们提供了一些算法未能正确识别的样本。如图 12 所示，在某些情况下，屏幕上的附加元素会误导算法。例如，在图 12(f) 和图 12(c) 中，箭头和电池信息分别被错误地识别为“7”和“1”。在后续迭代中，将类似样本添加到训练集即可纠正这一问题。



(a) $53 \rightarrow 5.3$ (b) $61 \rightarrow 67$

(c) $5.81 \rightarrow 5.8$ (d) $99 \rightarrow 399$

(e) $94 \rightarrow 9.4$ (f) $1067 \rightarrow 106$

图12. 误预测图像示例。（预测 → 真实标注）