arXiv:2302.05662v1 [cs.DC] 11 Feb 2023

AUTO-SPMV：图形处理器上稀疏矩阵-向量乘法内核的自动优化

Mina Ashoury设拉子大学计算机科学与工程系，伊朗设拉子
minaashoury@gmail.com

Mohammad Loni梅拉达伦大学创新、设计与工程学院，瑞典韦斯特罗斯
mohammad.loni@mdu.se

Farshad Khunjush设拉子大学计算机科学与工程系，伊朗设拉子
khunjush@shirazu.ac.ir

Masoud Daneshtalab梅拉达伦大学创新、设计与工程学院，瑞典韦斯特罗斯
masoud.daneshtalab@mdu.se

摘要

稀疏矩阵-向量乘法（SpMV）是一种重要的线性代数运算，在许多科学应用中占据了主要的计算开销。由于图形处理器能够提供大规模并行度和高内存带宽，因此常用于加速稀疏矩阵-向量乘法内核。以往研究主要致力于降低图形处理器上稀疏矩阵-向量乘法内核的延迟。然而，鲜有研究尝试提升其能效，导致图形处理器未能应用于低功耗应用。此外，先前工作主要关注优化稀疏矩阵-向量乘法内核的稀疏格式，相关文献却忽略了评估调整编译参数所带来的影响。最后，研究者很少关注构建全面的稀疏矩阵-向量乘法内核运行训练数据集，以及微调学习超参数。为解决这些局限，我们提出了一种名为 Auto-SpMV 的新框架，可在图形处理器上实现高能效、低延迟的稀疏矩阵-向量乘法内核。为实现最佳运行时性能，Auto-SpMV 提出了两种优化模式：编译时 和运行时。在编译时 模式下，Auto-SpMV 会调整编译参数；而在运行时 模式下，Auto-SpMV 会为输入矩阵选择最佳稀疏格式。为获得最佳分类结果，我们：1）收集了迄今规模最大的此类数据集，其中包含 30 个不同的稀疏矩阵，涵盖 1.5 万余种不同配置；2）通过自动微调学习超参数来提升分类模型。实验结果表明，与默认设置相比，Auto-SpMV 在编译时 模式下分别将延迟、能耗、平均功率和能效优化了最高 51.9%、52%、33.2% 和 53%。在运行时 模式下，Auto-SpMV 将平均功率和能效分别优化了最高 34.6% 和 99.7%，同样优于默认设置。

关键词 稀疏矩阵-向量乘法、图形处理器、性能建模、能效

1 引言

稀疏矩阵-向量乘法（SpMV）是许多科学应用中的关键操作，例如大型仿真系统 [35], 图学习 [39], 经济建模 [38], 等。优化

现代硬件设备上的稀疏矩阵-向量乘法内核至关重要，因为在特征值计算 [36] 和 Krylov 子空间 [37]等迭代求解方法中，SpMV 占据了主要的计算开销。GPU 是高吞吐量设备，因其易于编程且能够提供强大的处理能力和高内存带宽，已广泛用于加速各种科学应用。

以往研究探讨了如何提升图形处理器上稀疏矩阵-向量乘法内核的性能。 [71,79,81] 提出根据输入矩阵的稀疏模式，自动调整不同参数，例如每个线程可使用的最大寄存器数或线程块大小。另一研究方向是通过提出多种高效稀疏格式（如 COO [88], 压缩稀疏行 [88], ELL格式 [88], BELL格式 [90],和 SELL格式 [90]），减少稀疏矩阵-向量乘法内核的内存占用。然而，许多研究表明 [23,73,78], , 采用不恰当的稀疏格式可能导致性能显著下降（最高可达 1.6× [41]）。为解决这一局限，近期研究提出自动选择最优稀疏格式，以尽量缩短图形处理器上稀疏矩阵-向量乘法内核的执行时间和/或降低其能耗 [31,32,42,78]。所提出的稀疏格式选择方法虽已取得成功，却忽视了优化功耗（每瓦百万次浮点运算）的重要性，因此 GPU 无法在许多低功耗应用中处理稀疏矩阵-向量乘法内核 [26]。此外，稀疏矩阵-向量乘法内核在 GPU 上的性能对编译器配置参数（线程块大小、每个线程可使用的最大寄存器数以及存储层次结构配置）高度敏感 [81]；然而，文献尚未探讨这些参数的合理选择。最后，以往工作的结果是否可复现尚无定论，主要原因是其报告中缺乏全面评估 [33]。

为解决最先进方法的不足，我们提出了一种机器学习框架，可自动调整稀疏格式和编译参数，以针对不同的优化目标，包括：1）稀疏矩阵-向量乘法内核在图形处理器上的延迟（秒）；2）能耗（焦耳）；3）平均功耗（瓦特）；以及 4）能效（每瓦百万次浮点运算）。我们将这种自动调整稀疏矩阵-向量乘法内核配置参数的方法称为 Auto-SpMV。Auto-SpMV 支持两种优化模式：编译时和运行时。在编译时模式下，Auto-SpMV 调整编译器参数；而在运行时模式下，Auto-SpMV 根据稀疏输入矩阵的结构特征，自动选择效率最高的稀疏格式和编程内核。

本文考虑 CSR、ELL格式、BELL格式和 SELL格式，这些稀疏格式实现简单，并且在许多应用中的性能优于其他混合格式 [73]。为表征稀疏矩阵的稀疏矩阵-向量乘法内核性能，我们考虑稀疏输入矩阵的简单稀疏性特征。在学习过程中，我们使用一个由稀疏矩阵数据集训练多分类器，这些矩阵根据其稀疏性特征进行表征。随后，我们使用该模型为未见过的输入预测最优稀疏格式、每线程最大寄存器数（每线程最大寄存器数）、存储层次结构配置和线程块大小（TB 大小）。

为了实现最佳学习性能，分类器应在全面的数据集上进行训练。为满足这一要求，我们从 SuiteSparse 矩阵集合中收集了一个包含 30 个稀疏矩阵的数据集 [61]，其中记录了使用不同编译器参数和稀疏格式、在 GPU 上运行稀疏矩阵-向量乘法内核的结果。据我们所知，与所有现有的自动化稀疏矩阵-向量乘法优化研究相比，Auto-SpMV 提供了规模最大的训练数据集，包含 15520 条记录，涵盖在两种 GPU（Turing 和 Pascal 架构）上的 ≈70 次运行。为了提高分类准确率，我们需要微调学习超参数，例如神经网络的学习率。为满足这一要求，我们使用自动机器学习（AutoML）工具高效优化学习流程（第 5.4 节）。随后，我们报告最佳分类结果。本文还设计了回归模型，用于估算稀疏输入矩阵上的各种优化目标。

我们的主要贡献总结如下：

1. 通过使用高效的机器学习分类器来提升图形处理器上稀疏矩阵-向量乘法内核的性能。这使我们能够根据不同的优化目标，选择最优的编译参数和稀疏格式。2. 随着绿色图形处理器计算的普遍需求不断增长， [30], Auto-SpMV 考虑了新的优化目标：能效和平均功耗。除了延迟和能耗之外，评估这两个优化目标极其繁琐，因为必须在图形处理器上多次运行稀疏矩阵-向量乘法内核。3. 使用高度优化的分类器，而不是依赖采用默认学习超参数训练的常用模型。4. 收集迄今为止规模最大的 数据集，其中记录了在两种不同图形处理器架构上，对 30 个不同稀疏矩阵进行超过 ≈70M 次运行、涉及超过 15k 种配置设置的情况。论文被接收后将公开数据集、代码和配置参数。

结果显示，与最佳默认配置设置相比，对于所有选定的稀疏矩阵，Auto-SpMV 在编译时 模式下分别将延迟、能耗、平均功率和能效最多优化了 51.9%、52%、33.2% 和 53%。在运行时模式下，Auto-SpMV 将平均功率和能效分别最多优化了 34.6% 和 99.7%。我们分析了运行时内核选择以及不同稀疏格式间转换所带来的开销。最后，我们在两种不同的英伟达®GPU 设备（GTX 1650-mobile，采用 Turing 架构；以及 GTX 1080，采用 Pascal 架构）上进行的实验结果表明，Auto-SpMV 的预测结果不受底层 GPU 架构影响。

本文其余部分安排如下：第 2 节介绍本文考虑的稀疏格式，以及稀疏矩阵-向量乘法内核在图形处理器上的执行方式。第 3 节介绍稀疏格式和编译器参数的选择问题。第 5 节介绍 Auto-SpMV优化框架。第 4 节介绍评估所提方法的配置设置。第 7 节评估 Auto-SpMV 框架，并将其与其他基线方法进行比较。第 8 节深入讨论所提结果，并分析 Auto-SpMV 的处理开销。第 9 节回顾相关研究。最后，第 10 节总结全文。

2 背景

2.1 GPU 架构

图形处理器通过执行并行工作负载来最大限度地提高计算吞吐量，即使这会牺牲单项任务的串行性能。图形处理器配备了大量流处理器（SP），每个流处理器都具有全流水线逻辑单元。SP 被划分为多个流式多处理器（SM），以单指令多线程（SIMT）模式执行指令。内核是应用程序中与图形处理器相关的部分，由程序员定义的网格中的数千个线程执行，这些线程被组织成线程块。线程块是一组线程，会在 SM 上同时执行。SM 以每组 32 个并行线程的方式运行线程，这样的一组线程称为线程束。为了达到最大吞吐量，每个 SM 单元都应有足够多的活跃线程束，以隐藏内存和指令流水线的延迟。

图形处理器具有丰富的存储层次结构配置，包括 L1 和 L2 缓存能力、纹理内存、共享内存和全局内存。存储层次结构可由程序员定义，允许手动配置 L1 缓存和共享内存。线程块使用共享内存，块内所有线程均可访问和共享该内存。程序员可以在编译时使用 每线程最大寄存器数 变量，指定每个线程使用的寄存器数量上限。根据默认设置，CUDA® 编译器会尽量减少寄存器使用量，以最大限度地增加流式多处理器（SM）中可同时活动的线程块数量，从而提高 SM 的占用率。图 1 展示了 GPU 架构概览。

2.2 图形处理器上的稀疏矩阵-向量乘法处理

稀疏矩阵-向量乘法（SpMV）是一种复杂的二级基本线性代数子程序运算，用于计算稀疏矩阵 A 与稠密向量 X 的乘积，得到稠密向量 Y，即 $Y = A \times X$。图 2.a 展示了矩阵 A 与稠密向量 X 进行稀疏矩阵-向量乘法运算的示例。由于稠密格式需要存储零值，而零值乘法会分别导致内存使用和计算效率低下，因此稠密格式不适合在图形处理器上实现稀疏矩阵-向量乘法内核。为解决这一问题，人们提出了多种稀疏格式（第 2.3 节）。

2.3 常见的稀疏矩阵-向量乘法格式

为了提高在图形处理器上运行稀疏矩阵-向量乘法内核的效率，人们提出了多种稀疏格式，以避免不必要地：1）存储零元素；2）对零元素执行计算。需要注意的是，对于所有稀疏矩阵-向量乘法矩阵而言，并不存在一种适用于所有情况的最佳稀疏格式 [23, 73, 78]。换句话说，如果未谨慎选择稀疏格式，很可能会导致性能显著下降 [78]。由于矩阵具有稀疏性特征，稀疏格式可以采用扁平类型、块类型和复合类型。下面简要介绍本文所使用的几种最常见的稀疏格式。

压缩稀疏行格式 [88]。 根据压缩稀疏行格式，非零元素和列索引分别显式存储在数据和列索引 数组中。每一行的边界保存在第三个名为行索引的数组中。图 2.b 展示了矩阵 A 的压缩稀疏行格式。该格式需要扭曲内的线程相互协调，以汇总各线程的结果。尽管该格式无需零填充，但对数据 和列索引 的访问可能未对齐。由于对 X 的访问是随机的，且每行的非零元素数各不相同，该内核可能会受到负载不均衡问题的影响。

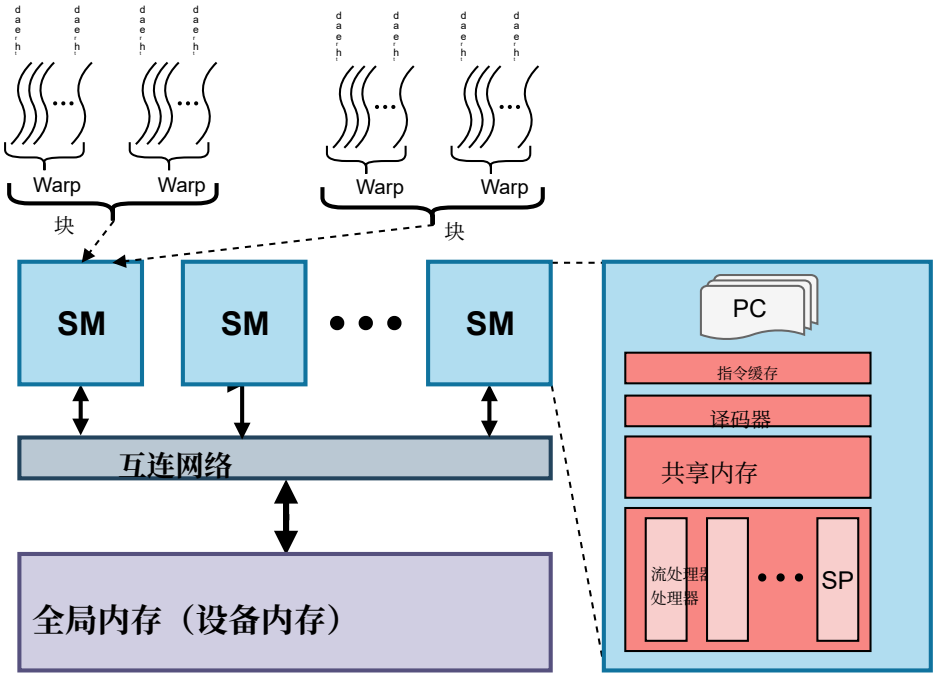


图 1：图形处理器架构概览。

ELL格式 [88]。 在 ELL格式中，非零元素被分组存储在一个稠密的数据 矩阵中，而矩阵 A 的列索引则存储在一个列索引 矩阵中。数据 矩阵的维度为$m \times max_nnz$，其中max_nnz表示一行中非零元素数的最大值。图 2.c 展示了矩阵 A 的 ELL格式。尽管 ELL格式具有较高的效率，但它并不一定能保证对输入向量 X 进行连续访问。

分块ELL（BELL）格式 [90]。 这是 ELL格式 的一种变体，其中一个非零元素块被视为 ELL格式 的一个元素。图 2.d 展示了矩阵 A 在默认块大小为 2×2 时的 BELL格式。BELL格式 由两个矩阵组成：1）数据 矩阵，用于存储非零元素块；2）列索引 矩阵，用于存储块索引。一般而言，这种格式适用于非零块呈均匀分布的矩阵。

切片ELL（SELL）格式 [90]。 该格式的每个切片都由矩阵 A 中固定数量的行组成。每个切片的长度等于该切片中每行非零元素数量的最大值。要将矩阵压缩为 SELL格式，必须将矩阵的每个切片视为一个采用 ELL格式的矩阵打包，并通过存储每个切片的指针来实现。SELL格式由三个矩阵组成：1）切片索引 用于保存指向各切片的指针；2）数据 矩阵用于存储各切片中的非零元素；3）列索引 矩阵用于存储数据 元素的列索引。图 2.e 展示了矩阵 A 在切片高度设为 2 时的 SELL格式。对于每行非零元素数量各不相同的稀疏矩阵，这种格式非常适用。

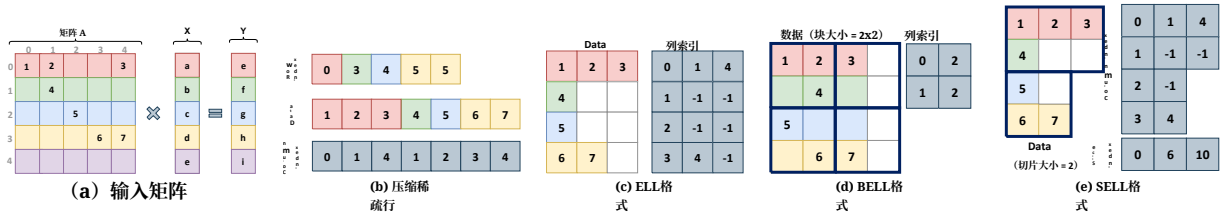


图 2：图示 (a) 稀疏矩阵-向量乘法、(b) 压缩稀疏行、(c) ELL格式、(d) BELL格式和 (e) SELL格式。彩色空单元格表示稀疏输入矩阵中的零值，白色单元格表示零填充。

3 研究动机

本节首先说明调整编译时参数的重要性，以及默认优化方案的低效之处。然后，我们讨论以往学习模型的局限性；这些模型试图对图形处理器上稀疏矩阵-向量乘法内核的延迟和能耗进行建模。

3.1 默认优化参数的低效性

图3比较了Auto-SpMV对稀疏矩阵-向量乘法内核的存储格式和编译参数进行优化的结果与默认 CUDA® 编译器配置参数的结果，所使用的稀疏矩阵为作为示例基准测试给出的*consph*稀疏矩阵。结果已按照Auto-SpMV进行归一化（数值越高越好）。由于压缩稀疏行 [88] 格式应用广泛，我们选择它作为CUDA的默认稀疏格式。结果表明，与默认 CUDA® 编译参数相比，对于所选稀疏矩阵，Auto-SpMV至少带来了2.04× 倍更低的延迟、2.07× 倍更低的能耗、1.08× 倍更低的平均功耗，以及高出2.086%的能效。显然，默认配置参数并非最优，导致所有优化目标均出现显著下降。因此，优化稀疏矩阵-向量乘法内核的编译器配置参数和稀疏格式至关重要。

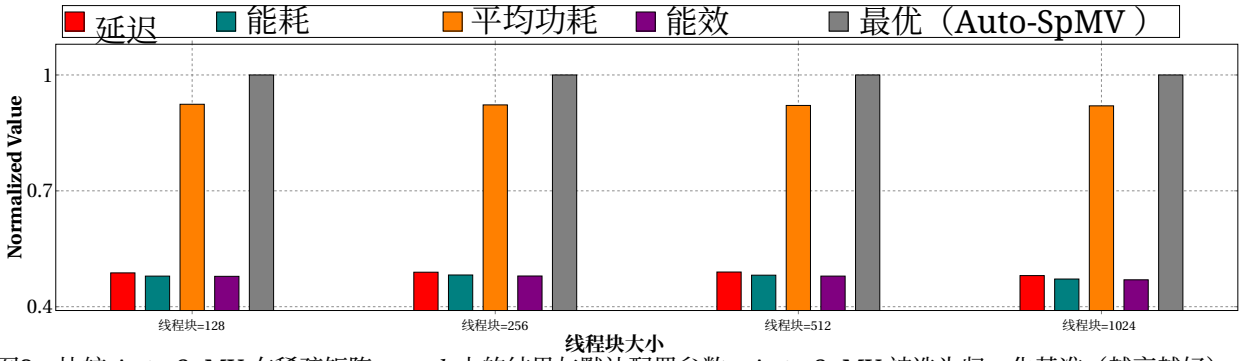


图3：比较 Auto-SpMV 在稀疏矩阵*consph* 上的结果与默认配置参数。Auto-SpMV 被选为归一化基准（越高越好）。

3.2 现有学习模型的局限性

由于 GPU 架构复杂，且稀疏性特征种类繁多，因此很难描述每种稀疏性特征对图形处理器上稀疏矩阵-向量乘法内核性能的影响 [78]。缺乏这种全面的理解，正是采用机器学习方法的动机。要在图形处理器上高效执行稀疏矩阵-向量乘法内核，需要优化多个软件和硬件参数，从而形成庞大的优化空间，以供选择最佳参数值。因此，使用穷举优化算法寻找最优解在实践中并不可行，因为这一过程极其耗时 [76]。已有多种解析模型被提出，用于估算稀疏矩阵-向量乘法内核的性能并选择最佳稀疏格式 [24]。然而，由于硬件的复杂性，且需要通过假设简化问题，解析模型无法应对现代应用的复杂性 [23]。

为应对现有挑战，先前研究 [21–23, 31, 32, 67] 提出设计机器学习模型，以 1）估算图形处理器上稀疏矩阵-向量乘法内核的性能和能耗，以及 2）预测近乎最优的配置参数。采用机器学习模型有诸多优势，其中包括不依赖硬件规格（例如每个流式多处理器的核心数）。此外，无须对问题作简化假设。尽管先前研究在构建机器学习模型方面取得了成功，但由于 1）收集的训练数据集相对较小，以及 2）评估的机器学习模型数量有限，其预测仍不准确（准确率最高损失 16% [78]）。为应对这些挑战，Auto-SpMV 构建了准确度极高的机器学习模型，并使用由 15520 个样本组成的大型数据集进行训练。为了构建这一全面的训练数据集，研究人员在两种不同的英伟达® 图形处理器架构 Pascal 和 Turing 上进行了约 7000 万次运行。

4 配置参数

本节通过消融研究，分别考察每个编译参数和稀疏格式对优化目标的影响。图4展示了 Auto-SpMV 针对作为示例基准测试的 *eu-2005* 稀疏矩阵，对稀疏矩阵-向量乘法内核的各项配置分别进行优化时，在不同优化目标上产生的影响。

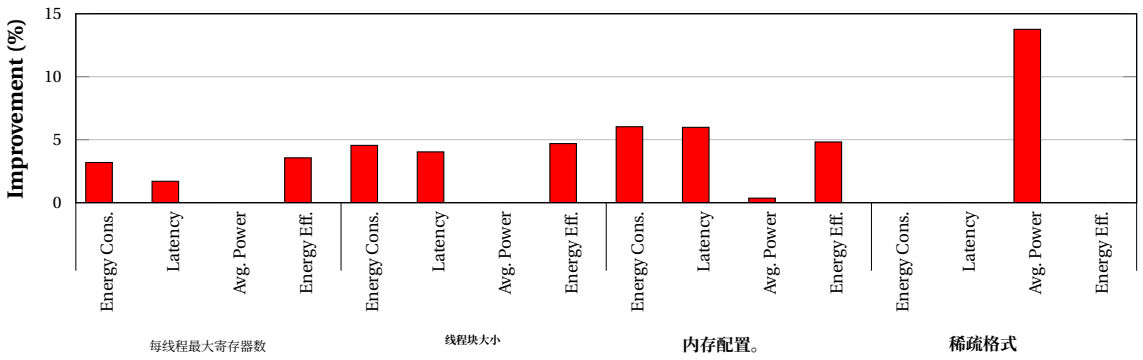


图4：在不同优化目标下，Auto-SpMV 各配置参数带来的单项改进。我们选择 *eu-2005* 作为示例基准测试。

图4所示的实验结果带来了以下观察：

1. **每线程最大寄存器数**（`maxrregcount`）。寄存器内存是线程私有内存，由某个流式多处理器上的所有驻留线程共享。所有局部变量都存放在寄存器或局部内存中。由于局部内存位于设备动态随机存取存储器中（尽管可以缓存在片上），而寄存器位于片上，因此最好将线程私有变量存放在寄存器中。内核中每个线程可使用的最大寄存器数由编译器控制。不过，程序员可以为编译单元中的每个内核声明所使用的寄存器数。限制每个线程的寄存器数可能会增加可同时驻留在一个流式多处理器上的线程块数量，从而提高延迟隐藏能力。然而，限制寄存器数也可能导致寄存器溢出，即可用寄存器不足以完成给定任务。因此，寄存器可能会溢出到全局内存中。由于寄存器溢出与更高占用率之间存在相互制约，通常需要进行一些实验才能获得最佳配置 [58]。

2. **线程块大小（TB 大小）**。稀疏矩阵-向量乘法内核是内存受限应用，占用率更高的内核具有更强的延迟隐藏能力。为了提高内核占用率，我们需要增大线程块的大小。然而，较大的线程块大小可能导致每个流式多处理器中的资源使用不均衡。在所有线程之间均衡分配工作负载的同时确定合适的线程块大小，可以减少空闲线程，从而提升性能。另一方面，占用率是 CUDA® 程序中线程并行度的一个指标，并且可以通过增大线程块大小来提高。但是，将线程块大小最大化可能导致可实现的并行度被浪费；例如，当线程块被挂起时，可处于活动状态的其他线程块会更少。因此，每个块的最优线程数是占用率与性能权衡的结果，需要经过一定的实验才能确定。

3. **存储层次结构配置**。对于稀疏矩阵-向量乘法等内存受限内核，优化的挑战在于利用图形处理器的高速存储资源和缓存能力来减少内存延迟时间。利用存储层次结构以及寄存器等其他硬件资源的可重配置性，可能实现图形处理器设备的能效优化。理想情况下，每个线程首次加载后，下一个元素将位于 L1 缓存中；但一个流式多处理器上最多可运行 2048 个线程，它们共享 L1 缓存，并且很可能会相互替换对方的数据。尽管对于数据密集型应用而言，增大图形处理器上的 L1 缓存大小是有益的，但由于需要将所访问的全局地址转换为缓存位置，使用 L1 缓存时可能会产生一些额外延迟。因此，行中非零元素数量不同的各种输入矩阵，对缓存和共享内存的需求也各不相同。

4. **稀疏格式**。 SpMV 所涉及的不规则计算使其性能优化颇具挑战。因此，人们投入了大量精力设计数据格式，以存储稀疏矩阵并最大限度地提升性能。由于矩阵具有稀疏性特征，稀疏格式可以

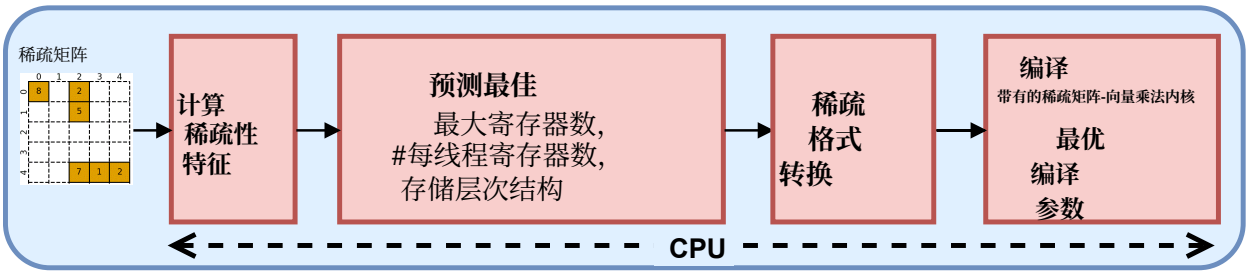
采用扁平、块状和复合类型。因此，我们选取了四种应用广泛的稀疏格式，它们对稀疏性特征的规格要求较少，且具有较强的通用性。

尽管提高图形处理器上稀疏矩阵-向量乘法内核效率的主要方法是选择最佳稀疏格式 [29]，但内核优化并不止于此。如本节所示，优化编译器参数对提升图形处理器上稀疏矩阵-向量乘法内核效率的影响更大。因此，Auto-SpMV 通过设计高精度的机器学习模型，实现对编译时 参数（线程块大小、每线程最大寄存器数和存储层次结构配置）以及运行时 参数（稀疏格式）的优化。此外，Auto-SpMV 能够针对更广泛的优化目标寻找最佳配置设置，包括延迟、能耗、能效和平均功耗。

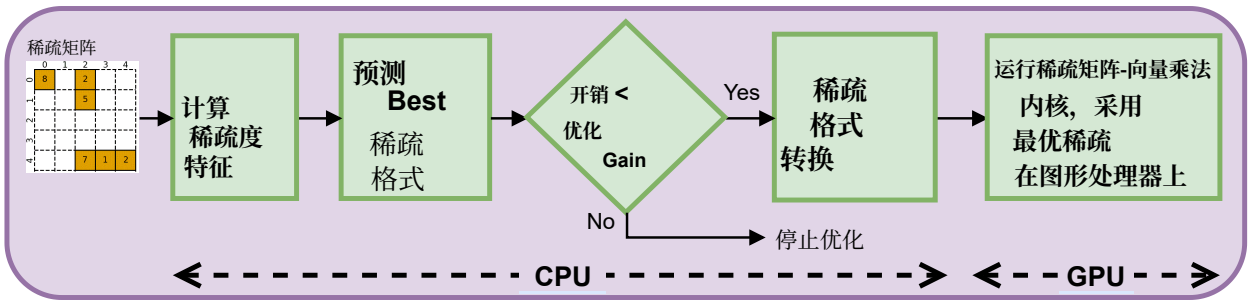
5 Auto-SpMV 优化方法

5.1 方法概述

如图 5 所示，Auto-SpMV 针对各种优化目标，在编译时 和运行时 两种模式下优化稀疏矩阵-向量乘法内核。在编译时 模式下（第 5.2 节），优化编译器配置参数。在运行时 模式下（第 5.3 节），Auto-SpMV 优化稀疏输入矩阵的稀疏格式。两种优化模式均在 CPU 上执行。本节其余部分将介绍优化模式的详细信息、从输入矩阵中提取的稀疏性特征，以及 Auto-SpMV 性能预测流水线。



(a) 编译时优化模式



(b) 运行时优化模式

图 5：所提出的 Auto-SpMV优化框架概览。(a) 编译时 优化众数。(b)运行时 优化众数。

5.2 编译时 优化众数

编译时 优化众数旨在通过调整 CUDA®编译参数，实现图形处理器效率最大化。如图 5 所示，编译时 优化众数包括以下步骤：

1. 计算稀疏性特征。2. 预测最佳图形处理器编译参数。3. 将稀疏输入矩阵转换为 CSR 存储格式（作为默认格式），并使用最佳 CUDA® 编译参数编译 CSR 稀疏矩阵-向量乘法内核。

我们不报告编译时优化模式的优化开销，因为整个过程都是在编译时执行的。

5.3 运行时 优化众数

运行时优化模式旨在为稀疏输入矩阵找出性能最佳的稀疏矩阵-向量乘法内核及其稀疏格式。给定一个采用默认稀疏格式并使用默认编译参数的稀疏矩阵，执行以下步骤以预测其最佳稀疏格式：

1. 计算稀疏性特征。2. 为给定的稀疏矩阵预测最佳稀疏格式。3. 估算优化开销。4. 如果优化收益高于估算的优化开销，则将稀疏输入矩阵转换为预测的最佳稀疏格式。

为避免对并非由迭代求解器构成的应用进行不必要的转换，Auto-SpMV 利用基于机器学习的模型估算特征提取和转换的开销，从而判断是否值得进行优化，并通过从预测收益中减去估算开销（第 7.5 节），评估是否可以接受由此产生的开销。图6展示了运行时优化模式下特征提取和转换开销的估算性能。结果表明，Auto-SpMV 能够在运行时准确估算特征提取和转换的开销。

在我们的工作中，我们使用了适用于压缩稀疏行、ELL格式、BELL格式和SELL格式的稀疏矩阵-向量乘法内核实现。所用数据集包含来自 SuiteSparse 矩阵集合的 30 个稀疏矩阵 [61]（第 6.1 节）。稀疏性特征的计算、开销/稀疏格式预测，以及转换为预测的最佳稀疏格式，均在应用程序运行时由 CPU 执行。在线稀疏格式优化的开销将在第 7.5 节讨论。

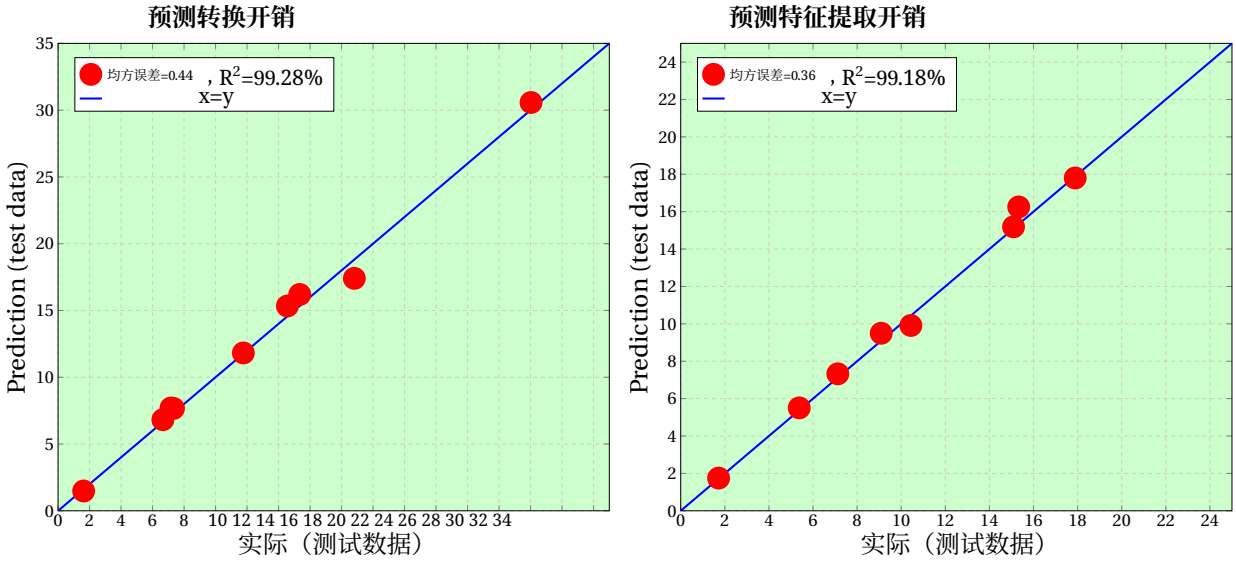


图6：在线优化模式下估算特征提取和转换开销的性能。

5.4 预测最佳配置参数

通常，我们有两种方法来学习最佳配置参数：

1. 多类分类：训练分类器，使其根据提取的稀疏性特征学习最佳配置参数。请注意，针对不同的优化目标，应训练不同的分类器。2. 回归：首先，训练不同的回归模型，用于估计稀疏矩阵-向量乘法内核的不同优化目标。然后，根据最佳估计结果选择最佳配置参数。

Auto-SpMV 依赖于多类分类方法。不过，我们还提供了用于估计稀疏矩阵-向量乘法内核不同优化目标的回归模型结果（第7.4节）。通过使用机器学习算法，我们可以通过以下步骤预测稀疏矩阵-向量乘法内核的最佳配置参数和稀疏格式：

1. **步骤1**。 提取稀疏矩阵的稀疏性特征，以表征在图形处理器上运行稀疏矩阵-向量乘法内核的特性。2. **步骤2**。 构建涵盖多种稀疏矩阵的学习数据集。3. **步骤 3**。 微调机器学习算法，以提供最准确的预测。

本文在分类和回归任务中均采用监督学习算法。此外，我们使用 Optuna（一种最先进水平的自动机器学习超参数优化库） [75]。Optuna 采用贝叶斯优化 [63] 作为搜索方法，可针对给定数据集自动选择合适的超参数。

以往研究的主要缺点有：1）依赖单一学习模型（例如，BestSF 仅使用支持向量机模型 [78]）；2）构建的优化空间规模相对较小（例如， [80] 仅包含两个优化候选项）。相比之下，Auto-SpMV 构建了一个大规模数据集，其中包含四个优化目标，并使用 Optuna 微调六种不同的学习模型（最近质心、决策树、非线性支持向量机、梯度提升、随机森林和多层感知机），以提供最准确的结果。表1列出了用于微调学习模型的超参数及其对应取值范围。第6节介绍了微调后学习模型的规格。

表1：为微调而选取的各学习模型的超参数范围。

机器学习模型	超参数
MLP	隐藏层大小:{20, 50, 100, 150, 200}, #层数:{1, 2, 3, 4, 5, 10}, 激活函数:{"恒等函数", "逻辑函数", "双曲正切函数", "ReLU"}
决策树	criterion:{"基尼", “熵", "log_loss"}, splitter:{"最佳", "随机"}
最近质心分类器	metric:{"曼哈顿距离", "欧几里得距离", "闵可夫斯基距离"}
非线性支持向量机	内核: {"线性", "多项式", "径向基函数", "Sigmoid", "预计算"}
梯度提升	#估计器：50、100、150、200，学习率：0.1、0.01、0.001
随机森林	准则：{"基尼", "熵", "log_loss"}

5.5 稀疏性特征

为了确定两种优化模式下的最优配置参数，Auto-SpMV 首先需要从稀疏输入矩阵中提取八种不同的稀疏性特征。这些稀疏性特征简要列于表2中。

表2：Auto-SpMV 选用的稀疏性特征。

特征	定义
n	行数
nnz	非零元素数
平均_非零元素数	每行非零元素的平均数量
Var_nnz	每行非零元素的方差
ELL_比率	ELL格式中非零元素与矩阵大小的比率
中位数	每行非零元素的中位数
Mode	每行非零元素的众数
Std_nnz	各行非零元素的标准差

我们根据以下两点选择特征：1）运行时计算开销最小；2）相关论文报告的性能影响 [78]。图形处理器的复杂性以及稀疏模式的多样性，使得我们无法准确说明每种稀疏性特征会如何影响图形处理器上稀疏矩阵-向量乘法内核的性能。尽管如此，关于不同稀疏格式的性能，我们总结了以下一般观察结果：

1. 行数（n）和非零元素数（**nnz**）刻画了在图形处理器上运行的稀疏矩阵-向量乘法内核的计算量。2. 图形处理器上的稀疏矩阵-向量乘法性能取决于所用的稀疏格式，并且会受到稀疏矩阵各行中非零元素分散程度的影响。为刻画这种敏感性，我们纳入了平均值_nz、方差_nnz和标准差_nnz等稀疏性特征。3. 对于非零元素分布规则的矩阵，压缩稀疏行稀疏矩阵-向量乘法内核性能良好。这类矩阵的平均值_nz 较大，而标准差_nnz相对较小，因此这些内核不会受到线程间负载分配不均的影响。4. 较小的ELL格式_比率 是矩阵中存在许多非零元素块的良好指标，这有利于采用 ELL格式，因为在这种情况下，ELL格式较少受到过多零填充问题的影响；过多零填充会带来不必要的额外计算。

6 实验设置

6.1 基准测试

SuiteSparse [61] 提供了丰富的稀疏矩阵集合。SuiteSparse 源自现实应用，例如数值线性代数、计算流体动力学、计算机图形学和金融建模。我们根据三个标准从 SuiteSparse 库中选取了 30 个稀疏矩阵：1）矩阵维度范围广（14,340<n<1,489,752）；2）非零元素数范围广（800,800<(非零元素数)<19,235,140）；3）矩阵的稀疏性特征相似度最低，以涵盖更多样的基准测试。图7展示了所选矩阵的稀疏性特征分布，并按非零元素数的升序排列。图8展示了稀疏性特征的皮尔逊相关。结果表明，所选矩阵的稀疏性特征之间的相关程度较低。

6.2 硬件规格

本文使用两种不同的英伟达® 图形处理器架构——Pascal 和 Turing，来评估学习结果对硬件设备变化的敏感性。表3列出了所用硬件设备的规格。我们使用英伟达® CUDA® v. 11 在图形处理器上编译稀疏矩阵-向量乘法内核。

6.3 图形处理器性能监控

本文考虑了四个优化目标，包括：1）延迟；2）能耗；3）平均功耗；以及 4）能效（每瓦百万次浮点运算）。执行时间和能耗采用多线程技术进行测量：除负责分配内存并在图形处理器上启动内核的线程外，另由一个独立线程在主机 CPU 上运行。为计算执行时间，我们使用英伟达® 管理库（NVML）的函数 `QueryPerformanceCounter()`。该函数会获取性能计数器的当前值，并提供高分辨率时间戳（< 1微秒）。根据内核的复杂程度，为满足图形处理器片上传感器返回瞬时功耗测量值所需的时间间隔，我们会在图形处理器上运行每个内核 500–200000 次 [85]。随后报告这些运行的平均执行时间，作为该内核的执行时间。

我们使用 GPU功率传感器获取能量和功耗。在 CPU 上运行的线程通过 NVML 接口查询内部传感器，传感器会返回以毫瓦为单位的功率读数。计算能量和平均功耗时，不考虑空闲功率样本。功率样本的算术平均值作为平均功耗，而功率样本对时间样本的积分则视为稀疏矩阵-向量乘法内核的能耗。最后，能效（每瓦百万次浮点运算）通过将实际达到的每秒百万次浮点运算数（MFLOPS）除以平均功耗计算得出。

6.4 学习算法

表 4 展示了微调后最佳学习模型的规格。其余学习模型的规格采用 Scikit-learn 机器学习库的默认超参数 [62]。我们使用数据集的 80% 进行训练，20% 用于验证预测结果。

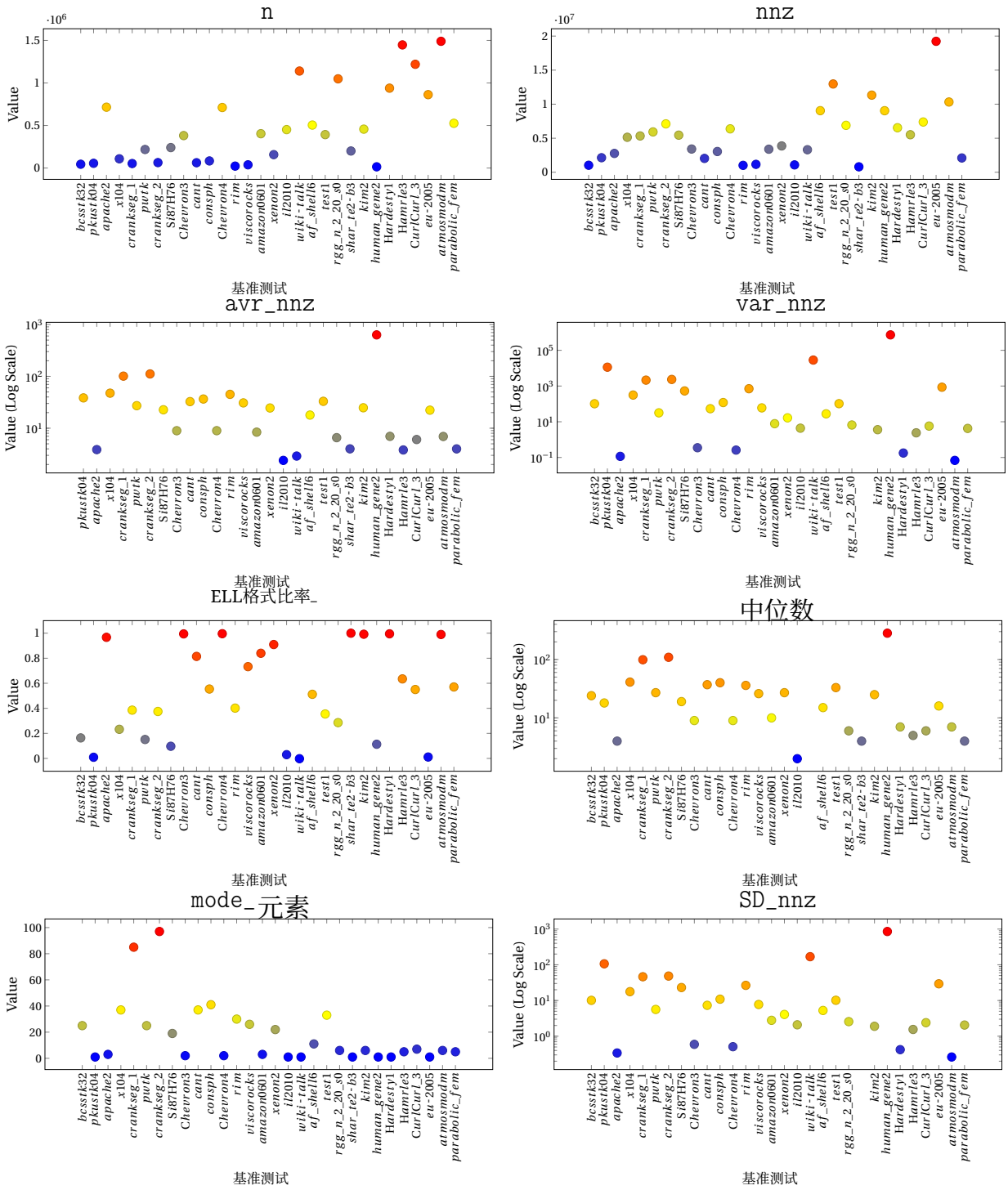


图7：所研究矩阵的结构特征分布，按 非零元素数 递增排序。可以看到，所选矩阵涵盖了广泛的稀疏性特征。

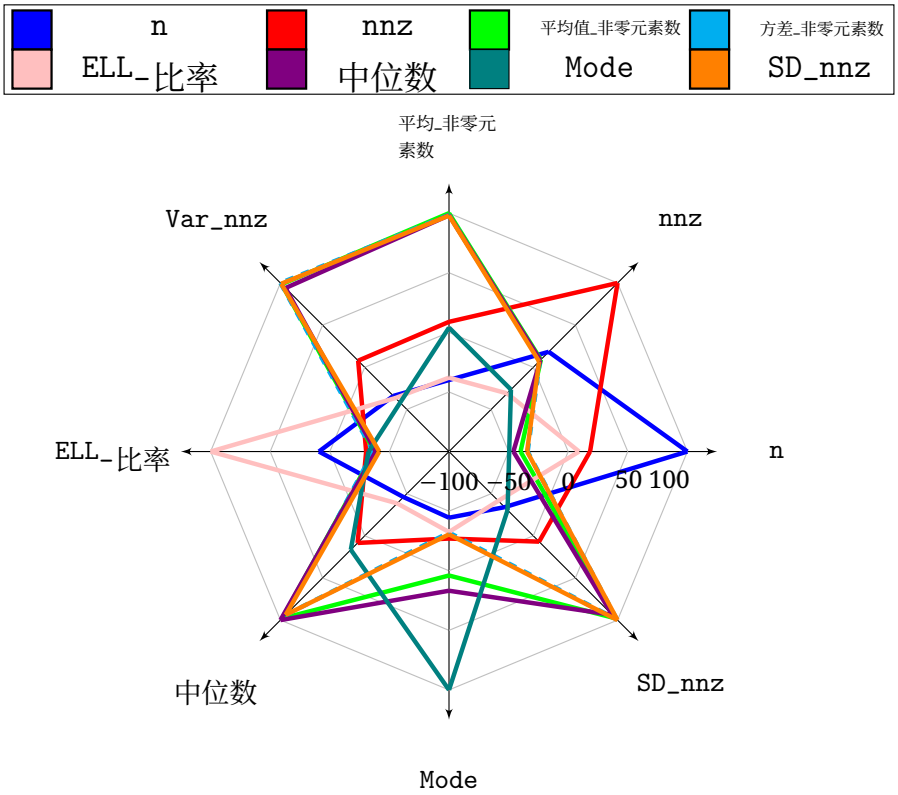


图8：所选基准测试的稀疏性特征之间的皮尔逊相关（%）。我们可以看到，它们之间不存在相关性。

表3：硬件规格。

GPU 设备	**规格**	**数值**
GTX 1650-mobile	时钟频率	1.6 GHz
	片上内存	4GB GDDR5X
	# CUDA® 核心	896
GTX 1080	基准时钟频率	1.6 GHz
	片上内存	8GB GDDR5X
	# CUDA® 核心	2560
CPU 设备	**规格**	**值**
AMD® Ryzen 5	时钟频率	3.6 GHz
	内存	16 GB

表 4：总结微调后用于预测图形处理器上稀疏矩阵-向量乘法内核最佳超参数（分类）以及估计各种目标函数（回归）的学习模型规格。

	学习算法	超参数与取值
Classification	最近质心	度量=曼哈顿距离，#邻居=5
	决策树	深度 = 13
	非线性支持向量机	内核=径向基函数，C=1.0，degree=3，gamma=scale
	梯度提升	损失函数=合页损失，阿尔法=0.0001，训练轮数=1000
	随机森林	估计器=100 最大深度=15
	多层感知机	层=5，每层节点数=100 激活函数=ReLU，#训练轮数= 200 优化器=Adam，学习率=0.001
Regression	贝叶斯岭回归	#迭代次数=300，容差=0.001
	套索回归	阿尔法=1.0，#训练轮数=1000
	随机森林	#估计器=100，深度=None
	决策树	深度=None
	Lars	#非零系数=500，eps=2.22045e-16，

	多层感知机	#层数=5，每层节点数=200
		激活函数=ReLU，训练轮数= 200
		优化器=Adam，学习率=0.0001

7 实验结果

7.1 编译时 优化众数的结果

图 9 比较了对于不同优化目标，在 CSR稀疏格式下，Auto-SpMV 在<strong?编译时 优化模式中相较于默认参数的性能提升。每个柱状图的上限和下限分别表示调整线程块大小时的最佳结果和最差结果，因为该参数需要由程序员负责选择。结果表明，对于所有选定的稀疏矩阵，如果分别以延迟、能耗、平均功率和能效为目标优化编译器参数，Auto-SpMV 的性能最高可分别提升 51.9%、52%、33.2% 和 53%。

7.2 运行时优化模式的结果

图 10 比较了针对不同优化目标，Auto-SpMV 在运行时优化模式下相较于默认稀疏格式压缩稀疏行的性能提升。为确保公平比较，我们将编译参数设为最佳值。通过在运行时优化稀疏格式，Auto-SpMV 对所有选定的稀疏矩阵，平均功率最多提升 34.6%，能效最多提升 99.7%。由于压缩稀疏行是延迟和能耗优化目标下的最佳稀疏格式，因此 Auto-SpMV 的性能与默认设置相同。

7.3 分类结果

表 5 展示了 Auto-SpMV 在预测最佳配置设置时取得的最佳分类结果。我们采用准确率和 F1 分数作为评估指标。结果表明，Auto-SpMV 成功构建了高度准确的学习模型，在预测线程块大小、每线程最大寄存器数和存储层次结构配置的最佳设置时，准确率达到 100%。请注意，通过调整决策树分类器的超参数得出了最佳结果 [77]。

表 6 比较了 Auto-SpMV 与最先进水平方法的分类结果。与最先进水平方法相比，Auto-SpMV 的分类准确率至少提高了 10%。通过评估不同的学习模型并构建大规模优化空间，Auto-SpMV 能够高效地进行学习。

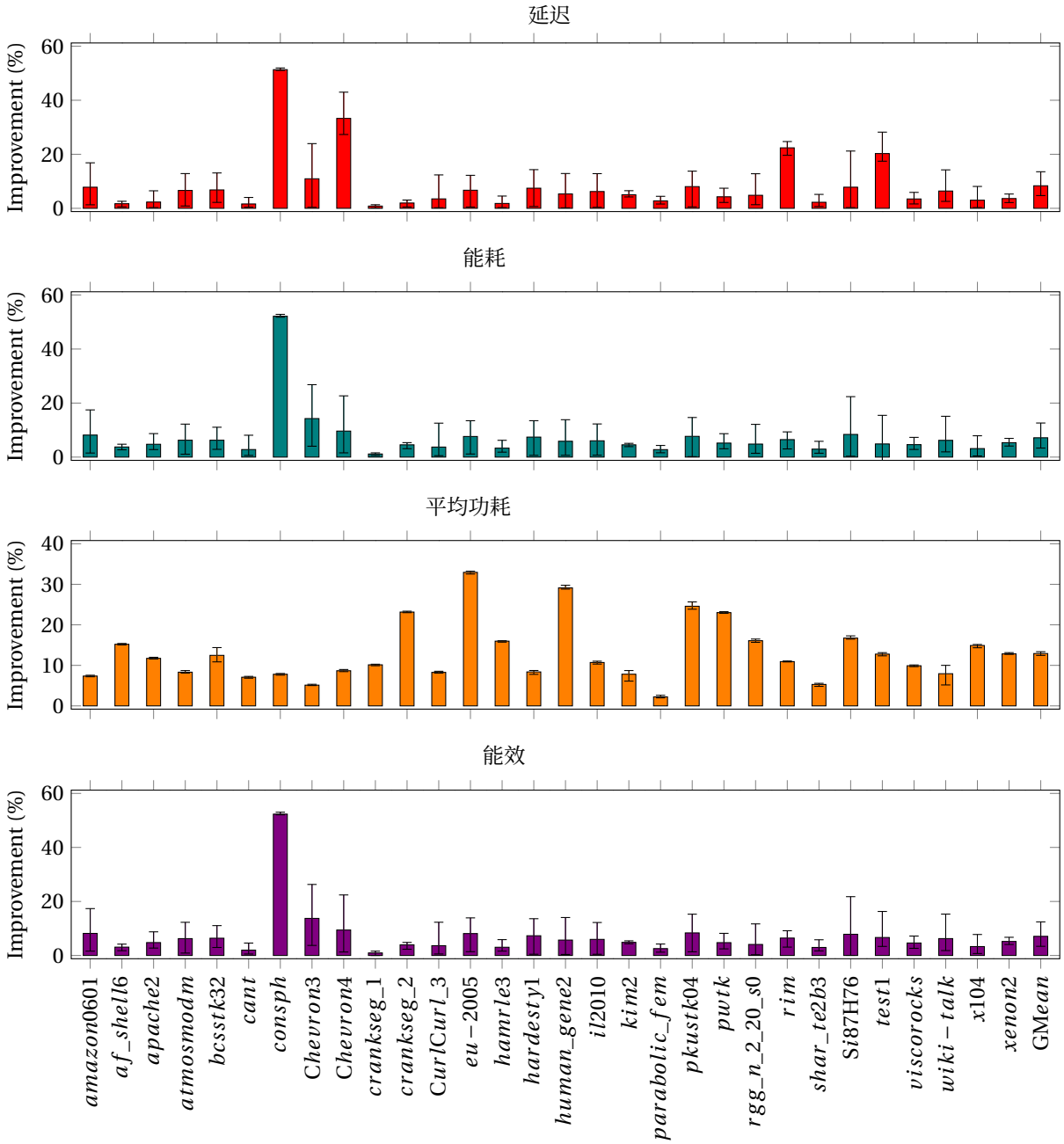


图 9：比较　　在不同优化目标下，将 Auto-SpMV 与默认配置设置进行比较　　s.

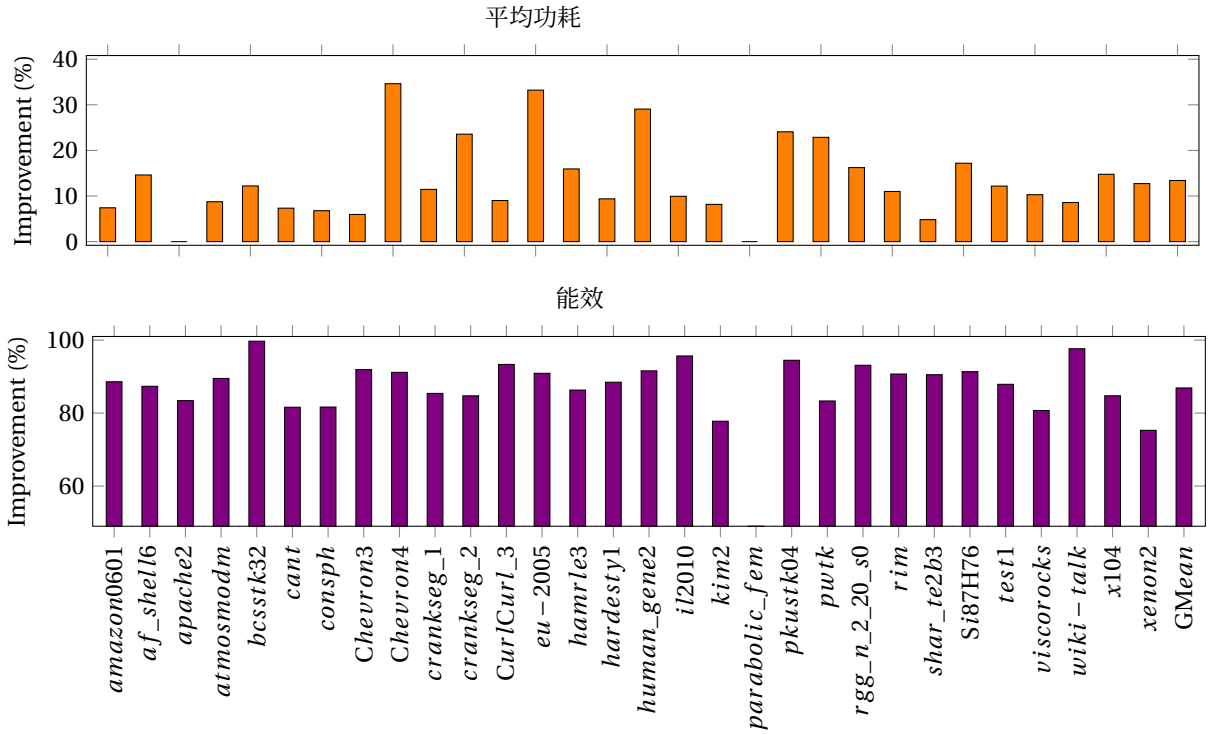


图 10：Auto-SpMV 针对不同优化目标，Auto-SpMV 在默认稀疏格式上的运行时间优化 .

表 5：用于预测在图形处理器上运行稀疏矩阵-向量乘法内核的最佳配置设置的最优学习模型（决策树）的分类准确率。

分类 (%)	**延迟**		**能量**		**平均功率**		**能效**	
	Acc.	F1 分数	Acc.	F1 分数	Acc.	F1 分数	Acc.	F1 分数
线程块大小	100	100	100	70	100	87.5	100	100
最大寄存器数	100	100	100	88.9	100	92.5	100	92.9
内存	100	91.7	100	81.8	100	50	100	91.7

表6：Auto-SpMV 与最先进水平的分类结果比较。

	学习模型	**硬件架构**	**准确率（执行时间）**	**准确率（能耗）**
BestSF [78]	SVM	麦克斯韦	82%	-
[74]	装袋树分类器	Pascal	89%	84%
[32]	CNN	Pascal	90%	-
Auto-SpMV（我们的方法）	决策树	Turing	**100%**	**100%**

7.4 回归结果

图 11 展示了对不同优化目标进行估计的回归结果。我们报告 R^2 相关性和均方误差估计指标，以确定每个优化目标的最佳学习模型。结果表明，随机森林是估计稀疏矩阵-向量乘法内核能耗和能效的最佳学习模型，其 R^2 相关性分别为 99.11% 和 99.94%，均方误差分别为 0.0086 和 0.00063。决策树是估计平均功率的最佳学习模型，其 R^2 = 99相关性为 .99%，均方误差为 $= 2.82\times10^{-5}$。多层感知机是估计延迟的最佳学习模型，其均方误差为 $= 1.9\times10^{-2}$。我们成功使用机器学习算法估计不同目标函数，主要得益于：1）使用了大型训练数据集，涵盖了具有不同规模和稀疏性特征的广泛稀疏矩阵；2）调整了学习模型的超参数。

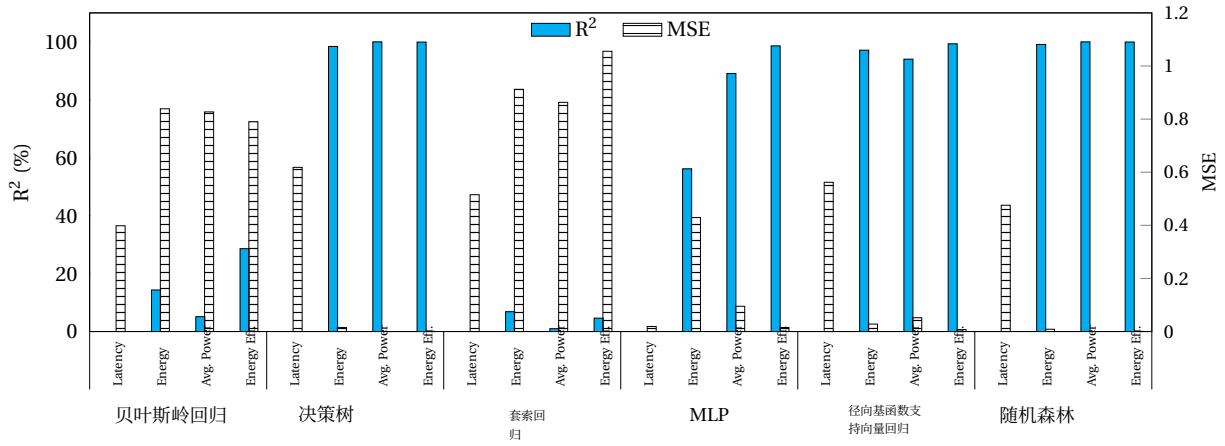


图 11: 评估用于估算不同优化目标的回归模型 s.

7.5 运行时预测开销

本节分析优化模式的开销。借助开销预测机制（第 5.3 节），当转换开销高于优化收益时，Auto-SpMV 不会执行不必要的转换。优化模式的开销根据 f_latency +o_latency + p_latency + c_latency 计算。f_latency 是提取特征所用的时间，o_latency 是预测转换开销所需的时间，p_latency 是预测最优稀疏格式所需的时间，而 c_latency 是转换为预测稀疏格式的延迟。输入矩阵通常以默认稀疏格式保存，在 SuiteSparse [61] 中，该格式为 COO 稀疏格式。请注意，运行时决策阶段的所有步骤都在 CPU 端执行，并使用 NumPy Python 数组与矩阵编程库的并行实现。

我们的测量结果表明，总运行时间开销主要由 f_latency 和 c_latency 之和构成，因为 o_latency 和 p_latency 是介于 ≈20 毫秒之间的常数值，而另外两种延迟则取决于给定输入矩阵的特征。表 7 展示了在以较低延迟为优化目标时，选定矩阵的开销。对于该数据集，我们的数据集的平均 f_latency+c_latency 为 24.2 秒，最大值为 87.8 秒，对应一个非零元素数接近 1.9×10^7 的稀疏矩阵（非零元素数）。如 [23,78]，所示，对于运行大规模稀疏矩阵的应用而言，这一开销可以忽略不计，例如使用预条件共轭梯度法这一迭代求解器的应用。因此，对于在迭代方法（例如特征值问题和线性系统 [69,78]）中对同一输入矩阵多次执行稀疏矩阵-向量乘法内核的应用，总预测开销微不足道。

7.6 学习模型对图形处理器的敏感性

我们必须验证分类结果是否独立于底层 GPU 架构。为此，我们在采用 Pascal 架构的英伟达® 1080 GPU 设备上，对 *amazon*0601、*crankseg*_2、*bcsstk*32、*x*104、*il*2010 和 C*hevron*3 这些稀疏矩阵重复进行了一部分评估。随后，我们将 Auto-SpMV 分类器提供的预测配置设置的性能，与在英伟达® 1080 GPU 设备上的实际测量结果进行比较。请注意，Auto-SpMV 的预测结果基于在采用 Turing 架构的英伟达® GTX 1650-mobile 设备上运行稀疏矩阵-向量乘法内核所得。图 12 比较了运行稀疏矩阵-向量乘法的最优测量结果

表 7：Auto-SpMV 各优化模式的优化开销（秒）。按非零元素数升序排列。

矩阵名称	nnz	f_延迟	c_延迟	f_延迟+c_延迟
shar_te2-b3	800800	1.71875	1.625	3.34375
rim	1014951	1.578125	2.046875	3.625
bcsstk32	1029655	1.71	2.125	3.835
il2010	1082232	3.625	2.5	6.125
viscorocks	1162244	1.90625	2.4375	4.34375
cant	2034917	3.4531	4.59	8.0431
抛物线_有限元	2100225	5.46875	4.984375	10.45313
pkustk04	2137125	3.78	4.53125	8.31125
apache2	2766523	7.84	6.06	13.9
consph	3046907	5.375	6.65625	12.03125
维基讨论时序网络	3309592	10.4375	7.3281	17.7656
amazon0601	3387388	7.125	7.171875	14.29688
Chevron3	3413113	7.0625	7.328125	14.39063
xenon2	3866688	6.75	9.375	16.125
x104	5138004	9.093	11.76563	20.85863
曲轴段_1	5333507	9.78025	11.75	21.53025
Si87H76	5451000	9.828125	11.90625	21.73438
Hamrle3	5514242	15.09375	12.89063	27.98438
pwtk	5926171	11.39	13.8593	25.2493
雪佛龙4	6376412	13.76563	14.71875	28.48438
Hardesty1	6539157	15.09375	14.5625	29.65625
rgg_n_2_20_s0	6891620	15.32813	15.34375	30.67188
crankseg_2	7106348	13.03125	15.25	28.28125
CurlCurl _3	7382096	17.89063	18.8125	36.70313
human_gene2	9041364	17.01563	21.70313	38.71875
af_shell6	9046865	18.78125	21.4687	40.24995
atmosmodm	10319760	24.14063	23.90625	48.04688
kim2	11330020	22.70313	27.10938	49.8125
test1	12968200	23.78125	30.03125	53.8125
eu-2005	19235140	39.82813	47.98438	87.8125

英伟达® 图形处理器上的内核与 Auto-SpMV 预测的结果。结果已归一化至实际测量值。结果表明，Auto-SpMV 能够预测最优配置设置（线程块大小和每线程最大寄存器数），误差很小（性能损失最高达 2%）。因此，Auto-SpMV 无需依赖底层 GPU 架构即可预测最优配置设置。

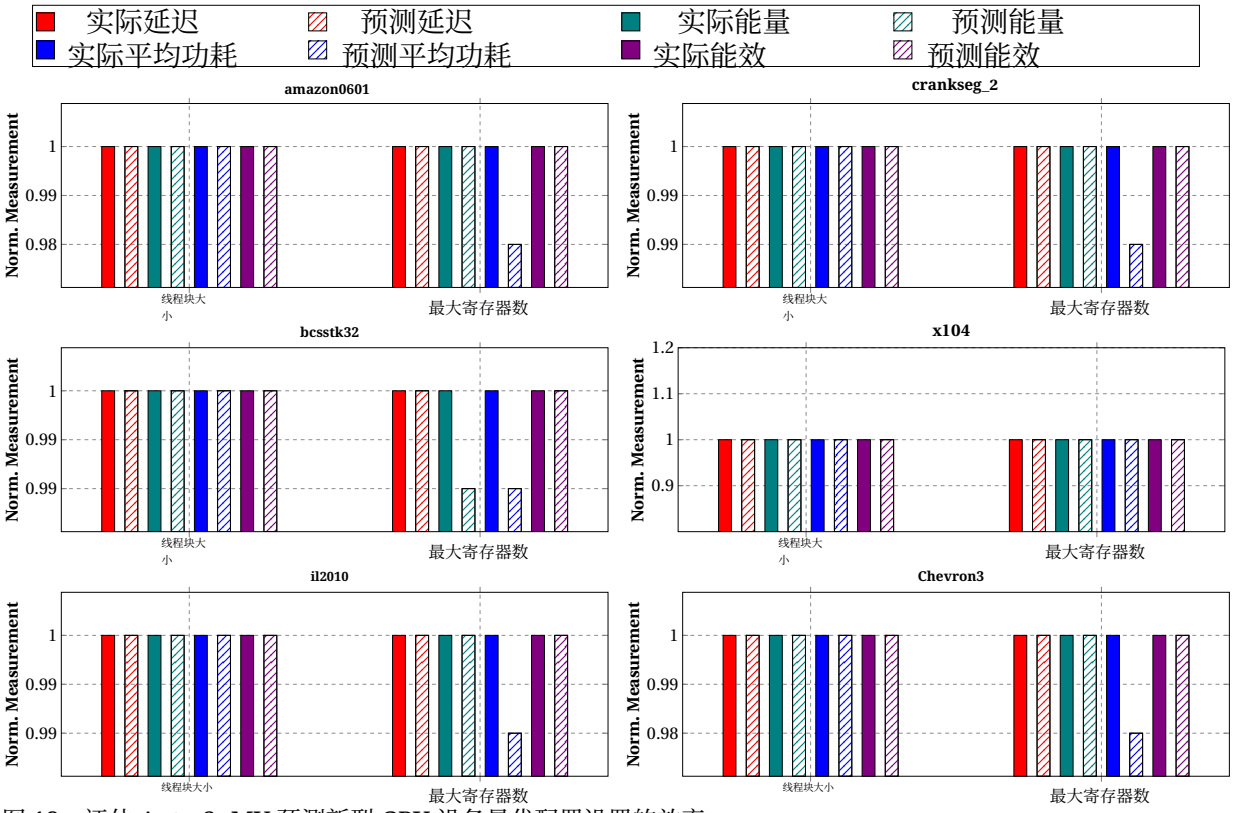


图 12：评估 Auto-SpMV 预测新型 GPU 设备最优配置设置的效率。

8 讨论

我们的研究结果总结如下：

1. 即使采用相同的稀疏格式，也不存在适用于所有稀疏输入矩阵的最佳配置设置。2. 不谨慎地选择编译器参数和存储格式，会导致延迟、能耗、平均功率和能效优化目标的性能平均分别损失 13.3%、5.4%、2.0% 和 13.9 %（图 9）。3. 将压缩稀疏行（CSR）视为默认稀疏格式时，不谨慎地选择稀疏格式（编译时 众数），会导致平均功率和能效优化目标的性能平均分别损失 13.4% 和 86.89%（图 10）。4. 即使程序员尝试选择最佳线程块大小，也无法实现最大改进，因为 CUDA® 编译器会选择每线程最大寄存器数 和存储层次结构配置。5. 结果表明，对于平均功耗和能效优化目标，压缩稀疏行（CSR）并非最佳格式。6. 增大每线程最大寄存器数 可以降低内存延迟。相反，减小每线程最大寄存器数 可以让更多线程处于活动状态，从而提高占用率并运行更多线程束。因此，低效的每线程最大寄存器数 会导致：1）全局内存中的寄存器溢出；2）占用率降低，进而导致延迟增加和能耗显著上升 [81]。

7. 增大线程块大小可以使每个流式多处理器（SM）上的线程数增加，从而提高线程占用率。另一方面，当线程块暂停时，活动线程块数量会减少，并导致并行度降低。因此，程序员应仔细权衡提高 SM 占用率与减少线程块数量之间的取舍，以实现更高的并行度，从而在能耗和性能方面提高效率。

8. 为了充分利用 GPU 计算能力，内存访问效率至关重要。根据稀疏输入矩阵的特征确定 L1 缓存和共享内存的最佳大小，可以确保更高的内存访问效率，从而提升性能和能效。

9. 稀疏格式的选择会深刻影响数据局部性、缓存性能和内存带宽利用率。结果表明，自动选择图形处理器上稀疏矩阵-向量乘法内核所用的稀疏格式，可以节省大量处理时间和能耗。

9 相关工作

据我们所知，Auto-SpMV 是首个同时考虑硬件和软件优化、旨在优化图形处理器上稀疏矩阵-向量乘法内核性能与能效的自动化框架。它也是首项证明编译器参数对 GPU 利用率、运行时性能和能耗具有重要作用的研究。过去，人们已对图形处理器上稀疏矩阵-向量乘法内核的高效运行进行了广泛研究。先前的研究可分为以下几类：1）提出新型稀疏矩阵-向量乘法稀疏格式；2）针对稀疏矩阵-向量乘法内核的软件优化技术；3）对图形处理器上稀疏矩阵-向量乘法内核的性能或能耗进行估算。本节将简要讨论这些方法，并将其与本文进行比较。

SpMV 稀疏格式。 [2, 4] 介绍了最早一批关于在图形处理器上加速稀疏矩阵-向量乘法内核的研究。这些研究详细分析了 CUDA® 实现的经典稀疏格式（包括 ELL格式、压缩稀疏行和 COO）的内存访问模式。在过去十年中，研究者提出了多种稀疏矩阵-向量乘法稀疏格式 [3, 12, 13, 15–18]。总体而言，这些格式显著节省了内存，且适用于任何稀疏矩阵。然而，正如我们在第 3 节所展示的，若不谨慎地选择稀疏格式和编译器参数，由于图形处理器利用率低，性能会显著下降。为最大限度地提升图形处理器上稀疏矩阵-向量乘法内核的性能，本文使用在大规模数据集上训练的高度优化机器学习模型，预测最佳配置设置。

一些稀疏矩阵-向量乘法稀疏格式利用稀疏矩阵的特定特征，以提高存储效率和索引效率 [5, 6, 8]。例如，CSR-VI 和 CSR-DU 利用非零值相似性，实现更高效的稀疏矩阵压缩 [8]。CSX 首先分析稀疏矩阵以寻找重复值或稠密对角线等模式，然后使用适合已发现模式的稀疏格式对矩阵进行编码 [5]。然而，这些方法仅适用于具有特定稀疏性特征的矩阵，而且识别稀疏性特征还需要耗费大量资源进行预处理。因此，我们的工作不考虑这些特定格式。

稀疏矩阵-向量乘法内核的软件优化技术。许多现有研究探讨了各种性能提升技术的潜力，例如自动调优 [19, 20], 对稀疏矩阵进行分区并组合稀疏格式 [43, 45], 提升缓存和寄存器的复用率 [47, 48] 以及软硬件协同设计优化 [49, 51]。尽管这些技术提升了性能，但在文献中，尚未解决如何在能耗较高的 GPU 上运行稀疏矩阵-向量乘法时，平衡能耗与性能之间的权衡。

GPU 上稀疏矩阵-向量乘法内核的性能或能耗估算。许多研究通过提出性能估算模型来选择最佳配置设置 [21–23, 31, 32, 34, 42, 78]，以加速稀疏矩阵-向量乘法内核。这些技术主要利用机器学习技术来学习稀疏矩阵的特征，并提供高度准确的预测。与最先进水平的工作相比 [34], , 我们考虑了更多稀疏矩阵特征，以提高预测准确率。除了延迟和能耗 [53, 57, 67], , 我们还对稀疏矩阵-向量乘法内核的能效和平均功耗进行建模，从而为低功耗应用提出最佳配置设置，例如在嵌入式设备上运行几何流任务 [72]。我们还使用自动机器学习工具优化不同的学习模型，为给定的稀疏矩阵找到性能最佳的预测模型。

10 结论

本文提出 Auto-SpMV，这是一种自动化框架，通过预测性能最佳的稀疏格式和编译器参数，优化图形处理器上的稀疏矩阵-向量乘法内核。Auto-SpMV 支持针对不同目标进行优化

，包括延迟、能耗、平均功耗和能效。我们使用真实世界基准测试进行的实验结果表明，与最佳默认编译器参数和稀疏格式相比，Auto-SpMV 实现了显著的性能和能效提升。我们希望关于图形处理器上稀疏矩阵-向量乘法内核性能特性的新发现，能够启发改进未来性能建模算法的新技术。

参考文献

[1] Im, E. & Yelick, K. 在对称多处理器（SMP）上优化稀疏矩阵-向量乘法。《PPSC》。(1999)[2] Bell, N. & Garland, M. 在吞吐量导向处理器上实现稀疏矩阵-向量乘法。高性能计算、网络、存储与分析会议论文集。第 1-11 页 (2009)[3] Langr, D. & Tvrdik, P. 稀疏矩阵存储格式的评估标准。《IEEE 并行与分布式系统汇刊》。27，428-440 (2015)[4] Bell, N. & Garland, M. 基于 CUDA 的高效稀疏矩阵-向量乘法。(英伟达技术报告 NVR-2008-004，英伟达公司，2008)[5] Kourtis, K., Karakasis, V., Goumas, G. & Koziris, N. CSX：用于共享内存系统中 SpMV 的扩展压缩格式。《ACM SIGPLAN 通知》。46，247-256 (2011)[6] Li, J., Tan, G., Chen, M. & Sun, N. SMAT：用于稀疏矩阵-向量乘法的输入自适应自动调优器。第 34 届 ACM SIGPLAN 编程语言设计与实现会议论文集。第 117-126 页 (2013)[7] Su, B. & Keutzer, K. clSpMV：面向 GPU 的跨平台 OpenCL SpMV 框架。第 26 届 ACM 超级计算国际会议论文集。第 353-364 页 (2012)[8] Kourtis, K., Goumas, G. & Koziris, N. 使用索引和数值压缩优化稀疏矩阵-向量乘法。第 5 届计算前沿会议论文集。第 87-96 页 (2008)[9] Buluç, A., Fineman, J., Frigo, M., Gilbert, J. & Leiserson, C. 使用压缩稀疏块并行执行稀疏矩阵-向量和矩阵转置-向量乘法。第二十一届算法与体系结构并行化研讨会论文集。第 233-244 页 (2009)[10] Willcock, J. & Lumsdaine, A. 通过数据压缩加速稀疏矩阵计算。第 20 届年度国际超级计算会议论文集。第 307-316 页 (2006)[11] Maggioni, M. & Berger-Wolf, T. AdELL：一种用于在 GPU 上高效执行稀疏矩阵-向量乘法的自适应线程束均衡 ELL 格式。2013 年第 42 届并行处理国际会议。第 11-20 页 (2013)[12] Maggioni, M. & Berger-Wolf, T. CoAdELL：通过自适应性和压缩提升 GPU 上的稀疏矩阵-向量乘法性能。2014 年 IEEE 并行与分布式处理国际研讨会专题研讨会。第 933-940 页 (2014)[13] Liu, W. & Vinter, B. CSR5：一种用于跨平台稀疏矩阵-向量乘法的高效存储格式。第 29 届 ACM 超级计算国际会议论文集。第 339-350 页 (2015)[14] Kreutzer, M., Hager, G., Wellein, G., Fehske, H. & Bishop, A. 一种统一的稀疏矩阵数据格式，可在配备宽 SIMD 单元的现代处理器上高效执行通用稀疏矩阵-向量乘法。《SIAM 科学计算杂志》。36，C401-C423 (2014)[15] Guo, D., Gropp, W. & Olson, L. 一种可提升 GPU 上稀疏矩阵-向量乘法性能的混合格式。《国际高性能计算应用杂志》。30，103-120 (2016)[16] Chen, X., Xie, P., Chi, L., Liu, J. & Gong, C. 一种用于稀疏矩阵-向量乘法的高效 SIMD 压缩格式。《并发与计算：实践与经验》。30，e4800 (2018)[17] Mofrad, M., Melhem, R., Ahmad, Y. & Hammoud, M. 使用三重压缩稀疏格式进行高效的分布式图分析。2019 年 IEEE 集群计算国际会议 (CLUSTER)。第 1-11 页 (2019)[18] Li, Y., Xie, P., Chen, X., Liu, J., Yang, B., Li, S., Gong, C., Gan, X. & Xu, H. VBSF：一种面向现代处理器上 SIMD 稀疏矩阵-向量乘法的新型存储格式。《超级计算杂志》。76，2063-2081 (2020)[19] Yan, S., Li, C., Zhang, Y. & Zhou, H. yaSpMV：又一个 GPU 上的 SpMV 框架。《ACM SIGPLAN 通知》。49，107-118 (2014)

[20] Hou, K., Feng, W. & Che, S. 面向多核和众核处理器并行化稀疏矩阵-向量（SpMV）乘法的自动调优策略。2017 年 IEEE 并行与分布式处理国际研讨会专题研讨会 (IPDPSW)。第 713-722 页 (2017)[21] Tan, G., Liu, J. & Li, J. 面向多核和众核架构的自适应 SpMV 库的设计与实现。《ACM 数学软件汇刊》(TOMS)。44，1-25 (2018)[22] Guo, P. & Zhang, C. 使用深度学习为基于 CSR 的 SpMV 选择稀疏矩阵。2019 年 IEEE 第五届计算与通信国际会议 (ICCC)。第 2097-2101 页 (2019)[23] Benatia, A., Ji, W., Wang, Y. & Shi, F. 使用机器学习方法预测 GPU 上 SpMV 的性能。2016 年 IEEE 第二十二届并行与分布式系统国际会议 (ICPADS)。第 894-901 页 (2016)[24] Guo, P., Wang, L. & Chen, P. 一种用于 GPU 上稀疏矩阵-向量乘法的性能建模与优化分析工具。《IEEE 并行与分布式系统汇刊》。25，1112-1123 (2013)[25] Li, K., Yang, W. & Li, K. 使用概率建模分析并优化 GPU 上的 SpMV。《IEEE 并行与分布式系统汇刊》。26，196-205 (2014)[26] Mach, P. & Becvar, Z. 移动边缘计算：架构与计算卸载综述。《IEEE 通信综述与教程》。19，1628-1656 (2017)[27] Cass, S. 英伟达让嵌入人工智能变得轻而易举：Jetson Nano 将强大的机器学习能力装进 DIY 项目中-[动手实践]。《IEEE Spectrum》。57，14-16 (2020)[28] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. & Others Scikit-learn：Python 机器学习。《机器学习研究期刊》。12，第 2825-2830 页 (2011)[29] Kanellopoulos, K., Vijaykumar, N., Giannoula, C., Azizi, R., Koppula, S., Ghiasi, N., Shahroodi, T., Luna, J. & Mutlu, O. Smash：协同设计软件压缩与硬件加速索引，以实现高效的稀疏矩阵运算。第 52 届 IEEE/ACM 微体系结构国际研讨会论文集。第 600-614 页 (2019)[30] Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J. & Jones, P. 比较 CPU、GPU 和现场可编程门阵列实现视觉内核的能效。2019 年 IEEE 嵌入式软件与系统国际会议 (ICESS)。第 1-8 页 (2019)[31] Nisa, I., Siegel, C., Rajam, A., Vishnu, A. & Sadayappan, P. 基于机器学习的有效格式选择与 GPU 上 SpMV 性能建模。2018 年 IEEE 并行与分布式处理国际研讨会专题研讨会 (IPDPSW)。第 1056-1065 页 (2018)[32] Zhao, Y., Li, J., Liao, C. & Shen, X. 弥合深度学习与稀疏矩阵格式选择之间的差距。第 23 届 ACM SIGPLAN 并行编程原理与实践研讨会论文集。第 94-108 页 (2018)[33] Lindauer, M. & Hutter, F. 神经架构搜索科学研究的最佳实践。《机器学习研究期刊》。21，1-18 (2020)[34] Li, M., Ao, Y. & Yang, C. 面向稀疏度各异的输入向量的 GPU 自适应 SpMV/稀疏矩阵-稀疏向量乘法。arXiv 预印本 arXiv:2006.16767。(2020)[35] Oyarzun, G., Peyrolon, D., Alvarez, C. & Martorell, X. 一种采用 FPGA 缓存的稀疏矩阵-向量乘积，用于非结构化计算流体动力学模拟。arXiv 预印本 arXiv:2107.12371。(2021)[36] Sgherzi, F., Parravicini, A. & S antambrogio, M. 面向大规模 Top-K 稀疏特征值问题的混合精度多 GPU 设计。arXiv 预印本 arXiv:2201.07498。(2022)[37] Ahamed, A. & Magoulès, F. 在 GPU 上使用迭代 Krylov 方法加速求解亥姆霍兹方程。2014 年 IEEE 高性能计算与通信国际会议、2014 年 IEEE 第六届网络空间安全国际研讨会、2014 年 IEEE 第十一届嵌入式软件与系统国际会议 (HPCC、CSS、ICESS)。第 54-61 页 (2014)[38] Schiesser, W. 工程与应用科学中的计算数学：常微分方程、微分代数方程和偏微分方程。(CRC 出版社，2014)[39] Qiu, S., Liang, Y. & Wang, Z. 优化图神经网络的稀疏矩阵乘法。arXiv 预印本 arXiv:2111.00352。(2021)[40] Krawezik, G., Kuntz, S. & Kogge, P. 在 Lucata Pathfinder-A 计算机上实现稀疏线性代数内核。2020 年 IEEE 高性能极限计算会议 (HPEC)。第 1-6 页 (2020)

[41] Lehnert, C., Berrendorf, R., Ecker, J. & Mannuss, F. 图形处理器架构上稀疏矩阵-向量乘法内核的性能预测与排名。欧洲并行处理会议。第 90-102 页（2016年）[42] Sedaghati, N., Mu, T., Pouchet, L., Parthasarathy, S. & Sadayappan, P. GPU 上稀疏矩阵表示的自动选择。第 29 届 ACM 超级计算国际会议论文集。第 99-108 页（2015年）[43] Yang, W., Li, K. & Li, K. 一种在 GPU 上采用分块格式和最优划分实现 SpMV 的并行计算方法。《计算机与系统科学杂志》。92，第 152-170 页（2018年）[44] Su, B. & Keutzer, K. clSpMV：一个跨平台的 GPU OpenCL SpMV 框架。第 26 届 ACM 超级计算国际会议论文集。第 353-364 页（2012年）[45] Muhammed, T., Mehmood, R., Albeshri, A. & Katib, I. Suraa：一种用于 GPU 上负载均衡与合并式 SpMV 计算的新方法和工具。《应用科学》。9，947（2019年）[46] Naumov, M., Chien, L., Vandermersch, P. & Kapasi, U. cuSPARSE 库。GPU技术大会。（2010年）[47] Xu, W., Zhang, H., Jiao, S., Wang, D., Song, F. & Liu, Z. 在 Fermi图形处理器上使用缓存分块方法优化稀疏矩阵-向量乘法。2012 年第 13 届 ACIS 软件工程、人工智能、网络与并行/分布式计算国际会议。第 231-235 页（2012年）[48] Nagasaka, Y., Nukada, A. & Matsuoka, S. 面向 Kepler图形处理器的缓存感知型稀疏矩阵格式。2014 年第 20 届 IEEE 并行与分布式系统国际会议（ICPADS）。第 281-288 页（2014年）[49] Kanellopoulos, K., Vijaykumar, N., Giannoula, C., Azizi, R., Koppula, S., Ghiasi, N., Shahroodi, T., Luna, J. & Mutlu, O. Smash：协同设计软件压缩与硬件加速索引，实现高效的稀疏矩阵运算。第 52 届 IEEE/ACM 微体系结构国际研讨会论文集。第 600-614 页（2019年）[50] Sadi, F., Fileggi, L. & Franchetti, F. 面向大规模SpMV问题的算法与硬件协同优化解决方案。2017 年 IEEE 高性能极限计算会议（HPEC）。第 1-7 页（2017年）[51] Animesh, S. 稠密与稀疏矩阵计算的算法架构协同设计。（亚利桑那州立大学，2018年）[52] Mullen, J., Wolf, M. & Klein, A. Pakck：CPU 和 GPU 上关键计算内核的性能与功耗分析。2013 年 IEEE 高性能极限计算会议（HPEC）。第 1-6 页（2013年）[53] Giefers, H., Staar, P., Bekas, C. & Hagleitner, C. 异构系统上稀疏矩阵乘法能效分析：GPU、Xeon Phi 和现场可编程门阵列的比较研究。2016 年 IEEE 系统与软件性能分析国际研讨会（ISPASS）。第 46-56 页（2016年）[54] Subramaniam, B., Saunders, W., Scogland, T. & Feng, W. 能效计算趋势：来自 Green500 的观点。2013 年国际绿色计算会议论文集。第 1-8 页（2013年）[55] Woo, D. & Lee, H. 扩展阿姆达尔定律以适应众核时代的能效计算。《计算机》。41，24-31（2008年）[56] Mukunoki, D. & Ogita, T. GPU 上精确与混合精度线性代数内核的性能和能耗。《计算与应用数学杂志》。372，第 112701 页（2020年）[57] Anzt, H., Tomov, S. & Dongarra, J. GPU超级计算机上稀疏计算的能效与性能前沿。第六届多核与众核编程模型及应用国际研讨会论文集。第 1-10 页（2015年）[58] Fatica, M. & Ruetsch, G. 面向科学家与工程师的 CUDA Fortran。（爱思唯尔，2014年）[59] Li, Z., Wang, Y., Zhi, T. & Chen, T. 神经网络加速器综述。《计算机科学前沿》。11，746-761（2017年）[60] Moons, B., Bankman, D. & Verhelst, M. 嵌入式深度学习。（施普林格，2019年）[61] Davis, T. & Hu, Y. 佛罗里达大学稀疏矩阵集合。《ACM 数学软件汇刊》（TOMS）。38，1-25（2011年）[62] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. & Others Scikit-learn：Python 中的机器学习。《机器学习研究期刊》。12，第 2825-2830 页（2011年）[63] Shahriari, B., Swersky, K., Wang, Z., Adams, R. & De Freitas, N. 将人类从循环中移除：贝叶斯优化综述。《IEEE 会刊》。104，148-175（2015年）

[64] Ryoo, S., Rodrigues, C., Baghsorkhi, S., Stone, S., Kirk, D. & Hwu, W. 使用 CUDA 的多线程 GPU 优化原则与应用性能评估。第 13 届 ACM SIGPLAN 并行编程原理与实践研讨会论文集。第 73-82 页（2008年）[65] Documentation, C. v10. 0，2018年。[66] Barreda, M., Dolz, M. & Castano, M. 用于估算稀疏矩阵-向量乘法运行时间和能耗的卷积神经网络。《国际高性能计算应用杂志》。35，268-281（2021年）[67] Benatia, A., Ji, W., Wang, Y. & Shi, F. GPU 上稀疏矩阵-向量乘法的能耗评估。2016 年第七届国际绿色与可持续计算会议（IGSC）。第 1-6 页（ 2016 年）[68] Braun, L., Nikas, S., Song, C., Heuveline, V. & Fröning, H. 一种用于便携式快速预测 GPU 内核运行时间和功耗的简单模型。《ACM 架构与代码优化汇刊》（TACO）。18，1-25（2020年）[69] Li, R., Xi, Y., Erlandson, L. & Saad, Y. 特征值切片库（EVSL）：算法、实现与软件。《SIAM 科学计算杂志》。41，C393-C415（2019年）[70] Kurzak, J., Bader, D. & Dongarra, J. 使用多核与加速器进行科学计算。（CRC 出版社，2010年）[71] Guo, P., Huang, H., Chen, Q., Wang, L., Lee, E. & Chen, P. 一个面向 GPU 稀疏矩阵-向量乘法的模型驱动分区与自动调优集成框架。2011 年 TeraGrid 会议：极限数字发现论文集。第 1-8 页（2011年）[72] Wang, Y., Feng, B. & Ding, Y. TC-GNN：利用 GPU 上的稠密张量核心加速稀疏图神经网络计算。arXiv 预印本 arXiv:2112.02052。（2021年）[73] Filippone, S., Cardellini, V., Barbieri, D. & Fanfarillo, A. 通用图形处理器上的稀疏矩阵-向量乘法。《ACM 数学软件汇刊》（TOMS）。43，1-49（2017年）[74] Dufrechou, E., Ezzatti, P. & Quintana-Orti, E. 通过机器学习为 GPU 选择最优 SpMV 实现。《国际高性能计算应用杂志》。35，254-267（2021年）[75] Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna：新一代超参数优化框架。第 25 届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集。第 2623-2631 页（2019年）[76] Loni, M., Sinaei, S., Zoljodi, A., Daneshtalab, M. & Sjödin, M. DeepMaker：面向嵌入式系统中深度神经网络的多目标优化框架。《微处理器与微系统》。73，第 102989 页（2020年）[77] Safavian, S. & Landgrebe, D. 决策树分类器方法综述。《IEEE 系统、人类与控制论汇刊》。21，660-674（1991年）[78] Benatia, A., Ji, W., Wang, Y. & Shi, F. BestSF：用于优化 GPU 上 SpMV 的稀疏元格式。《ACM 架构与代码优化汇刊》（TACO）。15，1-27（2018年）[79] Abu-Sufah, W. & Karim, A. 一种在图形处理器上实现稀疏矩阵-向量乘法的有效方法。2012 年 IEEE 第 14 届高性能计算与通信国际会议暨 IEEE 第 9 届嵌入式软件与系统国际会议。第 453-460 页（2012年）[80] Tsai, Y., Cojean, T. & Anzt, H. AMD 与英伟达 GPU 上的稀疏线性代数——竞赛已经开始。高性能计算国际会议。第 309-327 页（2020年）[81] Zardoshti, P., Khunjush, F. & Sarbazi-Azad, H. 用于高效矩阵-向量乘法的自适应稀疏矩阵表示。《超级计算杂志》。72，3366-3386（2016年）[82] CUSP：英伟达针对 CUDA 架构 GPU 上稀疏线性代数与图计算的通用并行算法库。（https://developer.nvidia.com/cusp），[在线][83] Davis, J. & Chung, E. SpMV：GPU 上一种受内存限制、进退两难的应用。微软硅谷研究院，技术报告，9 月 14 日。2012（2012年）[84] Ashari, A., Sedaghati, N., Eisenlohr, J., Parthasarath, S. & Sadayappan, P. 面向图应用的 GPU 快速稀疏矩阵-向量乘法。SC’14：高性能计算、网络、存储与分析国际会议论文集。第 781-792 页（2014年）[85] Burtscher, M., Zecena, I. & Zong, Z. 使用 K20内置传感器测量 GPU 功耗。使用 GPU 进行通用处理研讨会论文集。第 28-36 页（2014年）

[86] Bradley, T. GPU性能分析与优化。英伟达公司。（2012年）[87] Steinberger, M., Derlery, A., Zayer, R. & Seidel, H. GPU上的朴素SpMV究竟有多朴素？*2016*年*IEEE*高性能极限计算会议。第1-8页（2016年）[88] Bell, N. & Garland, M. 在吞吐量导向处理器上实现稀疏矩阵-向量乘法。高性能计算、网络、存储与分析会议论文集。第1-11页（2009年）[89] Godwin, J., Holewinski, J. & Sadayappan, P. 面向结构化网格计算的GPU高性能稀疏矩阵-向量乘法。第五届通用图形处理器处理年度研讨会论文集。第47-56页（2012年）[90] Choi, J., Singh, A. & Vuduc, R. GPU稀疏矩阵-向量乘法的模型驱动自动调优。*ACM Sigplan* 通知。**45**，115-126（2010年）[91] Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, J., Olakanmi, O. & Akinjobi, J. 监督式机器学习算法：分类与比较。《国际计算机趋势与技术期刊》（*IJCTT*）。**48**，128-138（2017年）