Distributed Deep Learning on Data Systems: A Comparative Analysis of Approaches

- ABSTRACT
  - **深度学习 (DL) 在许多数据分析应用程序中越来越受欢迎**，包括在企业中。
  - **此类设置中的大型关键业务数据集通常驻留在 RDBMS 或其他数据系统中。**
  - **DB 社区长期以来一直致力于将机器学习 (ML) 引入 DBMS 驻留数据。**
  - 鉴于 DBMS ML 过去的经验教训和可扩展 DL 系统的最新进展，**DBMS 和云供应商越来越有兴趣为 DB 驻留数据添加更多 DL 支持。**
  - 最近，提出了一种**新的并行 DL 模型选择执行方法**，称为 Model Hopper Parallelism (**MOP**)。
  - 在本文中，我们描述了MOP对于数据系统上DL的特殊适用性，但是为了将基于MOP的DL引入数据库驻留数据，我们证明了不存在单一的“最佳”方法，并且存在一个有趣的方法权衡空间。
  - 我们**解释了四种规范方法并在 Greenplum 数据库上构建原型，在多个标准（例如，运行时效率和易治理性）上对它们进行分析比较，并根据经验将它们与大规模 DL 工作负载进行比较。**
  - 我们的实验和分析表明，**很好地满足所有实际需求是非常重要的，并且存在Pareto边界；**例如，某些方法的速度提高了 3 到 6 倍，但在治理和可移植性方面表现更差。
  - 我们的结果和见解可以帮助 DBMS 和云供应商为 DB 用户设计更好的 DL 支持。
- INTRODUCTION
  - 用于数据分析的深度学习 (DL) 越来越受欢迎，导致对更容易采用 DL 的产品的需求不断增长，尤其是在企业中。
    - **DBMS 社区长期以来一直致力于使机器学习 (ML) 更接近企业中的关键业务数据集：DBMS 和其他数据系统。**
    - 这种“数据库内机器学习”（或“数据内系统机器学习”）的范式在过去的 20 年里起起落落，共有 3 波工作浪潮。 现在值得在 DL 时代重新审视它。
  - 有人可能想知道 **DL 对 DBMS 用户是否有用，因为 DL 主要用于非结构化数据，而 DBMS 主要处理结构化数据。**
    - **尽管 DL 的大部分成功都来自通常存储在文件系统或数据湖中的非结构化数据，但 DBMS 长期以来一直为文本、多媒体和其他对象提供存储支持。**
    - 此外，由于深度学习中嵌入学习和较少特征工程的好处，研究和企业应用程序中的许多近期工作表明，**深度学习即使在结构化数据上也变得越来越有用和有效。**
    - **结合结构化和非结构化数据的多模态分析也很流行并且与数据库用户相关。**
    - 最后，**ML 的“可解释性”痛苦曾经是某些企业用户的一大难题**，但 ML 研究人员正在积极缓解。

- 在 ML 实践的现实中，数据科学家不会以全有或全无的方式思考； 不同的模型类型（包括 DL）在不同的用例中很受欢迎。
- 最近的 Kaggle 调查证实，包括深度学习在内的多种模型类型仍然很受欢迎。 本文特别关注 DL，因为我们认为这是一个需要数据库社区更多关注的领域。

- In-RDBMS ML 的经验教训
  - 在 RDBMS ML 的第一波浪潮中，DB 供应商构建了“数据挖掘工具”，将一些 ML 算法扩展到 DB 驻留数据。
  - 他们允许从 SQL 控制台访问 ML。 但随着 ML 算法复杂性的增加，为数据内系统 ML 设计了第二波统一实现抽象；MADlib 和 Spark MLlib 是关键示例。
  - 第三波是云 DBMS 供应商增加了更多的 RDBMS ML 支持，例如 Google 的 BigQuery ML，以及从 DBMS 调用 DL。
  - 在这种背景下，DBMS 和云供应商越来越多地询问：“如何通过 DB 驻留数据实现对 DL 的无缝支持？”。 过去的 RDBMS ML 浪潮至少提供了四个教训。
    - In-RDBMS ML 工具的主要用户群不是面向 Python 的数据科学家，而是面向 SQL 的业务分析师。这些用户越来越希望从 SQL 控制台中访问 DL 训练和推理。根据 MADlib 团队的估计，如今约有 20-25% 的 Greenplum 客户将其 RDBMS 内 ML 分析功能与 SQL 分析一起使用。
    - 尽管治理和出处对于金融和医疗保健等敏感领域的企业来说一直很重要，但由于 GDPR 和 CCPA 等新法律，它们现在对包括科技巨头在内的所有公司来说都重新变得紧迫。**公司可能会开始反对 DL 用户以临时方式手动导出、复制和移动关键业务数据。尽管可以通过编程来自动化这些流程，并使用 MLFlow 和 Kubeflow 等服务作为治理和出处工具，但对于企业用户来说，学习仍然是一个额外的负担，尤其是当他们已经熟悉已建立的 DBMS 对治理/出处的支持时。**
    - **对于 DBMS 开发人员来说，重新实现 DL 算法太乏味了。** 因此，**必须保留 TensorFlow 等 DL 工具的可用性，用于指定复杂的 DL 工作负载。** 这也允许分析师仅重用由数据科学家或其他人编写的 DL 培训规范程序。
    - 并行 RDBMS 已经为分片的大规模数据提供了一个成熟的执行引擎。 但是，众所周知，**最先进的分布式 DL 执行工具（例如 Horovod）的设置、操作和调试仍然非常痛苦。** 这为并行 RDBMS/数据系统提供了弥合可扩展执行差距的机会。
  - 总的来说，我们看到了两种截然不同的范式，用于将 DL 引入 DB 驻留数据。
    - **DL 用户可以将数据导出到文件系统，手动调用 DL 工具，并自行管理所有派生数据/元数据/工件。**
    - 或者，**在“数据系统内 DL”方法中，ETL 和 DL 工作负载由数据系统编排，如图 1 所示。**
    - 至关重要的是，这种方法为 DL 工具如何使用数据的实现灵活性留下了空间； 这种灵活性开辟了我们稍后将探讨的可能性。

- 走向数据内系统深度学习
  - **Apache MADlib 最近开创了 DBMS DL 支持。DL 工作负载是使用 Keras API 指定的，使业务分析师能够重用由数据科学家等编写的 DL 配置。**

    - MADlib 将小批量数据从 DB 传送到在 DBMS 用户定义函数 (UDF)/用户定义聚合函数 (UDAF) 中调用的 TensorFlow 函数。
    - 对于分布式执行，MADlib 对 SGD 使用了“模型平均”（MA） 启发式。
    - Alas，MA 对于高度非凸的 DL 的收敛行为很差。 因此，这种方法对于将 DL 引入 DB 来说是次优的。
- 我们观察到 **MA 错过了 DL 中并行性的一个重要机会：模型选择。**
    - ML 理论告诉我们，**调整超参数至关重要，这需要训练许多模型。**
    - 通常，**DL 用户还会比较替代神经架构、更改基本特征等。**
    - 因此，**实践中的模型选择通常会导致一次性训练数十个甚至数百个模型。**
- 利用上述观察，**最近的工作提出了一种新的分布式 DL 模型选择方法，称为 Model Hopper Parallelism (MOP)。**
    - **MOP 是分片数据并行性和任务并行性的混合体。**
    - MOP 的工作原理如下：**在不同 worker 的本地分片上并行训练不同模型的一个 sub-epoch，检查点并在 worker 之间“跳跃”模型，然后在下一个 worker 的分片上重新开始训练相同的 epoch。**
    - **MOP 是批量异步并行的一种形式，因为它与批量同步并行 (BSP) 数据系统不同，它在工作程序之间没有强加障碍同步。**
    - 总体而言，**MOP 被证明是分布式 DL 模型选择中资源效率最高的方法。**
- 本文重点
    - 鉴于 MOP 的好处，我们问：“**如何将基于 MOP 的 DL 引入 DB 驻留数据？**”我们发现**没有单一的“最佳”方法，替代方法有一个有趣的权衡空间。**
        - **本文解释了这些方法，对它们进行了分析对比，并根据经验将它们与大规模 DL 工作负载进行了比较。**
        - 我们使用 Greenplum 作为原型，但强调比较的方法是通用的，适用于任何并行 RDBMS。
        - 因此，我们的结果可能会引起所有 DBMS 和云供应商的广泛兴趣。
    - 我们**寻求不改变数据系统代码的方法。**
        - **这简化了实际采用，但限制了 MOP 的应用方式。**
        - 比如Spark现在支持灵活调度worker； 这使得在 Cerebro 系统中轻松集成 MOP 和 Spark。
        - 但是并行的 RDBMS，如 Greenplum、AWS Redshift 等，跨工作线程使用 BSP，与 MOP 的异步性相冲突。
        - 我们有多个比较评估轴，包括**运行时效率**、**易治理性**、**实现难度**和**可移植性**。
        - 第 3 节解释了所有方法，第 4 节详细比较了它们，但作为预览，图 2 显示了前两个轴上的方法。
    - 我们比较了 4 种新方法：
        - **使用 UDAF 的 DBMS MOP，已被 MADlib 采用**

        - **使用并发目标查询 (CTQ) 的部分 DBMS MOP；**
        - **In-DB 但不是 in-DBMS（数据在 DB 中，但所有操作都不在）具有直接访问 (DA) 的 MOP；**
        - **使用 Cerebro-Spark 的常规 DBMS 外方法。**
    - MA 主要由 UDAF 方法主导，但所有其他方法都属于Pareto边界。
        - 例如，DBMS 外 Cerebro-Spark 方法和 DB 内 DA 方法比 UDAF 高效得多，但在生产环境中可能更难管理。
        - CTQ 方法在这两个轴上提供了一个中间立场。
    - 我们**对这些方法的比较分析揭示了更多有趣的差距**。
        - 例如，通过理论和仿真分析，我们表明**当模型和硬件更加异构时，CTQ 和 UDAF 之间的效率差距会变得更大，在现实场景中甚至高达 6 倍**。
        - 最后，**使用 ML 基准数据集 ImageNet 和 Criteo 进行的大量实证比较表明，UDAF 和 DA 之间的实际运行时差距高达 3 倍。**
        - 总的来说，我们的实验和分析表明，**将基于 MOP 的 DL 引入 DB 驻留数据是有益的，但满足所有实际需求并非易事。**
        - 我们希望我们的结果能在 DB 和云行业中激发更多关于如何最好地支持 DB 驻留数据的 DL 的对话。
    - 综上所述，本文有以下贡献：
        - 尽我们所知，这是第一篇分析在 DB 驻留数据上支持大规模 DL 模型选择的权衡和设计备选方案的论文。
        - 我们展示了一系列关于效率、易于治理和其他实际需求的Pareto边界的可能方法。 特别是，我们展示了一种在 DB 中而不是在 DBMS 中的新方法，这为 DB 供应商提出了新的可访问性问题。
        - 我们对新方法之间效率差距的限制进行了正式分析。
        - 我们使用大型 ML 基准数据集对这些方法进行了广泛的实证比较，以评估它们的运行时间、可扩展性和内部设计权衡。
- **总结**
    - 题目：《数据系统上的分布式深度学习：方法的比较分析》
    - 提示：它是一篇综述文章，主要就是从不同维度比较了现存的四种把DL引入数据库驻留数据的方法的好坏。
    - 背景
        - 深度学习 (DL) 在许多数据分析应用程序中越来越受欢迎。
        - 大型关键业务数据集通常驻留在 RDBMS 或其他数据系统中。
        - DB 社区长期以来一直致力于将机器学习 (ML) 引入 DBMS 驻留数据。DBMS 和云供应商越来越有兴趣为 DB 驻留数据添加更多 DL 支持。
        - 在经过**RDBMS ML的三波浪潮**后，人们吸取教训，开始思考：“**如何通过 DB 驻留数据实现对 DL 的无缝支持**？”

    - 最近，一种**新的并行 DL 模型选择执行方法出现**，称为 Model Hopper Parallelism (**MOP**)。
- 当前问题
    - 但是为了将基于MOP的DL引入数据库驻留数据，**并不存在单一的“最佳”方法，并且存在一个有趣的方法权衡空间。**
- 解决方案
    - 所以，文章介绍了**四种**将机器学习引入 DBMS 驻留数据的**规范方法**，并在 Greenplum 数据库上构建原型，在多个标准（例如，运行时效率和易治理性）上对它们进行分析比较，并根据经验将它们与大规模 DL 工作负载进行比较。
    - 通过实验和分析表明，**很好地满足所有实际需求是非常重要的，并且其中存在Pareto边界。**
    - 作者希望他的实验结果和见解可以帮助 DBMS 和云供应商为 DB 用户设计更好的 DL 支持。
- 扩充知识（备选介绍）
    - 四种规范方法
        - **UDAF**：使用 UDAF 的 DBMS MOP；
        - **CTQ**：使用并发目标查询 (CTQ) 的部分 DBMS MOP；
        - **MOP**：In-DB 但不是 in-DBMS（数据在 DB 中，但所有操作都不在）具有直接访问 (DA) 的 MOP；
        - **Cerebro-Spark**： 使用 Cerebro-Spark 的常规 DBMS 外方法；
    - 结论
        - DBMS 外 Cerebro-Spark 方法和 DB 内 DA 方法比 UDAF 高效得多，但在生产环境中可能更难管理。
        - 当模型和硬件更加异构时，CTQ 和 UDAF 之间的效率差距会变得更大，在现实场景中甚至高达 6 倍。
        - 使用 ML 基准数据集 ImageNet 和 Criteo 进行的大量实证比较表明，UDAF 和 DA 之间的实际运行时差距高达 3 倍。
        - 将基于 MOP 的 DL 引入 DB 驻留数据是有益的，但满足所有实际需求并非易事。
    - Pareto frontier
        - 指在多目标优化的过程中，如果所考虑的多目标优化问题有两个以上的目标函数，则可以使用该曲线描述这些目标函数之间的关系。其多为一条向外突出的曲线，及代表在照顾A目标的利益时，B目标的利益必定会受到损失。
        - 本文的目标函数有：**运行时效率**、**易维护性**、**实现难度**和**可移植性**。
    - In-RDBMS ML 的经验教训（三波浪潮）
        - In-RDBMS ML 工具的主要用户群不是面向 Python 的数据科学家，而是面向 SQL 的业务分析师。这些面向 SQL 的业务分析师越来越希望从 SQL 控制台中访问 DL 训练和推理。

- 由于 GDPR 和 CCPA 等新法律，对敏感数据的处理对所有公司来说都十分棘手。公司可能会开始反对 DL 用户以临时方式手动导出、复制和移动关键业务数据。尽管可以通过编程来自动化这些流程，并使用 MLFlow 和 Kubeflow 等服务作为管理和导出的工具，但对于企业用户来说，学习仍然是一个额外的负担，尤其是当他们已经熟悉已建立的 DBMS 对治理/出处的支持时。
- 对于 DBMS 开发人员来说，重新实现 DL 算法太乏味了。
- 尽管并行 RDBMS 已经为分片的大规模数据提供了一个成熟的执行引擎。但是**最先进的分布式 DL 执行工具（例如 Horovod）的设置、操作和调试仍然非常痛苦。**