异构环境中进化式自动化机器学习计算性能的提升

Nikolay O. Nikitin、Sergey Teryoshkin、Valerii Pokrovskii、Sergey Pakulin、Denis Nasonov

ITMO University，俄罗斯圣彼得堡

arXiv:2301.05102v1 [cs.LG] 12 Jan 2023

摘要

资源密集型计算是限制自动化机器学习解决方案有效性的主要因素。本文提出一种模块化方法，可用于提升具有基于图的结构的建模流水线的进化优化质量。该方法包含多个阶段：并行化、缓存和评估。评估阶段可利用异构资源和远程资源。所开展的实验验证了该方法的正确性和有效性。已实现的算法作为开源框架 FEDOT 的一部分提供。

关键词：自动机器学习、异构基础设施、进化优化、缓存

1. 引言

如今，自动化机器学习（自动机器学习）已广泛应用于科学研究和工业领域 [16, 33]。使用自动机器学习解决现实世界任务时，主要问题在于搜索最优建模流水线的计算成本高昂。在评估候选流水线的质量时，需要训练许多机器学习模型。这项任务对资源的需求很高，因此可能需要花费大量时间才能获得合适的结果。对于任何现有的自动机器学习解决方案而言，这都可以视为一个瓶颈。这一问题在不同领域引发了各种难题：从将自动机器学习集成到业务流程中 [31]，到碳排放和可持续性方面的担忧[35]。

最先进的自动机器学习解决方案采用了许多提高计算性能的方法 [8]。首先，几乎所有解决方案都支持并行执行。其中一些还支持缓存已评估的候选项 [22]。此外，还可以使用图形处理器来缩短训练时间 [15]。

有多种开源工具可以提高机器学习流水线中某些步骤的效率。例如，使用各种机器学习运维工具（如 MLFlow） [38],、特定任务数据库 [37]以及 Ray 等缩放工具 [18]，可以显著提升机器学习应用的效能。

然而，计算策略的最佳设计取决于基础设施和底层自动机器学习算法。当前最先进的自动机器学习解决方案基于不同的优化方法：随机搜索、贝叶斯优化、遗传算法和元学习 [8, 2]。建模流水线采用的结构模式也可能不同：线性流水线或集成技术（堆叠、混合，以及提升） [39]。最复杂的情况是以有向无环图表示的复合流水线 [21]。与此同时，在复合人工智能领域，目前还没有现成的解决方案来提升自动化开放式流水线搜索的计算性能。

本文旨在提出一种自适应方法，以降低复合流水线中自动机器学习的计算成本。我们实现了若干技术：流水线缓存、适应度函数评估的并行化、利用混合系统（图形处理器和中央处理器）进行计算，以及与远程分布式系统集成。

该方法有别于现有解决方案，因为它可配置为在各种计算环境（包括分布式和异构环境）中进行自动化机器学习。此外，缓存过程可有效用于各种流水线设计（线性、加权集成、多层集成等）。

为以实证方式确认所提出方法的有效性，我们使用不同规模的开放数据集（见表1）开展了一系列数值实验。第6节所呈现的结果使我们得出结论：采用该方法的自动机器学习能够评估更多的流水线，并获得更优的质量指标。该软件实现已集成于开源自动机器学习框架 FEDOT 中。

本文结构如下：第 2 节介绍先进的自动化机器学习工具所采用的计算策略。第 3 节提出自动机器学习性能提升的问题陈述。第 4 节提出一系列针对复合进化式自动化机器学习的新颖改进。第 5 节介绍这些技术在开源框架中的软件实现。第 6 节对所提出的技术在不同案例研究中的应用进行实验评估。最后，第 7 节分析所得结果，并探讨研究可能的扩展方向。

电子邮箱地址： nnikitin@itmo.ru (Nikolay O. Nikitin)

2. 相关工作

有数十种开源自动机器学习解决方案可用于设计建模流水线。最早广为人知的框架包括 H2O [15], TPOT [14] 和 Auto-sklearn [4]。较新的自动机器学习解决方案包括 Auto- Gluon [3] 和 LAMA [36] 。此外，还有许多具备各种特定特征的其他自动化机器学习工具 [17]。

上述框架采用了不同的性能改进策略。在 TPOT 框架中实现了流水线缓存 [14]。TPOT-SH [26] 利用逐次减半的概念，更快地探索搜索空间，尤其适用于较大的数据集。人们采用多种技术，在不同的训练数据子集上评估流水线（例如分层 [6]）。

一种广泛使用的并行化工具是用 Python 实现的 joblib 库。不过，也有一些更先进的并行化框架值得关注。例如，Ray[18] 可用于扩展分布式环境中的人工智能和 Python 应用。它提供了多种工具，可用于分布式数据预处理、机器学习模型的分布式训练以及可扩展的超参数调优。

文献中也讨论了自动机器学习之外提升进化算法计算性能的方法。例如，可采用基于图形处理器的并行适应度函数评估 [24] ，以解决与大数据 [23]相关的高成本问题。还有多种技术可以提升并发模式下进化算法的性能 [7]。基于张量的计算模型可用于实现跨平台硬件加速 [12]。此外，该领域还提出了针对特定平台的开源解决方案（例如 scikit-learn-intelex [1]）。

避免进化算法中适应度评估瓶颈的一种常用技术是缓存 [29]。适应度评估的最终值可以缓存 [13] ，部分结果也可以缓存 [30, 11, 10]。

此外，还有许多解决方案能够执行远程/分布式训练（例如 Auto-sklearn、H2O、TPOT 和 LAMA）。这些自动机器学习框架使用不同的分布式计算框架。Auto-sklearn 和 TPOT 使用 Dask[2]，LAMA 使用 Apache Spark[3]。H2O 使用其自行修改的 Apache Spark，称为 Sparkling Water[4]。分布式计算框架能够处理分布在集群系统各节点上的大型数据集。

我们可以得出结论，有大量技术和解决方案能够降低自动机器学习和进化算法的资源消耗。然而，目前仍缺乏成熟的方法，能够在复合人工智能问题中识别异构计算环境下的基于图的流水线。为此，我们决定针对复合自动化机器学习明确问题陈述，并提出可能的解决方案。

[1]https://github.com/intel/scikit-learn-intelex

[2]https://dask.org[3]https://spark.apache.org

[4]https://h2o.ai/products/h2o-sparkling-water

3. 问题陈述

我们希望使用单一的灵活工具，为各种任务设计多任务和多模态流水线。因此，有必要以更加抽象的方式实现框架架构，将流水线搜索过程与顶层 API 分离。在这种情况下，建模流水线表示为有向无环图。每个节点（建模操作或数据转换操作）均由操作名称及其超参数集描述。如有需要，流水线可以包含不同的数据源（表格、时间序列、图像、文本）。此外，元数据附加在数据流上，从而可以在流水线评估过程中多次更改任务（例如，先解决分类任务，然后解决回归任务）。

这种方法的缺点是优化过程中需要探索的搜索空间变大。在自动化建模中，我们希望控制开放性 [25] 与局部搜索之间的平衡。最简单的方法是直接施加约束（例如限制流水线规模）。此外，采用正则化和敏感性分析流程 [21] 以及自适应优化策略来控制优化的收敛，也可能更有效。同时，避免流水线过于复杂，并避免草率耗尽有限的时间预算，也至关重要。这使得计算部分的效率对于开放式自动机器学习尤为关键。

这迫使我们在流水线复杂度与训练时间之间做出权衡。不过，如果能够提高计算效率，该框架或许就能在训练时间不变的情况下构建更复杂、质量更高的模型。提高进化算法计算性能的方法有很多，例如并行化、缓存等。这些方法可以分为单机优化和水平扩展技术。单机优化旨在提升执行计算的机器上的计算性能。水平扩展则通过增加服务器来加速计算。这两种技术既可以单独使用，也可以结合使用。

进化算法的计算时间主要取决于种群规模和代数。增大种群规模会提高获得更优个体的概率。代数越多，就意味着有更多机会基于上一代中的最佳个体，逐步培育出更优的个体。

从计算角度来看，我们需要执行若干次迭代，每次迭代都需要上一轮迭代的结果。因此，要跨迭代扩展计算规模并不容易，总计算时间如公式1所示。

$$T_{total} = \sum_{i=1}^{n} T_i \tag{1}$$

其中n 是迭代次数，T_i - 是第i代的计算时间。在一代中，所有个体彼此独立地进行处理，因此我们可以

根据可用的计算资源扩展这些计算。种群训练时间可以用方程2估算。

$$T_{total} = \sum_{i=1}^{n} \underset{d_j^i \in D_i}{\text{argmax}} (\underset{r_j \in R_i / \{r_{j-1},...,r_1\}}{\text{argmin}} (\tau(d_j^i, \omega(D^{i-1},...,D^1), r_j)) \quad (2)$$

其中，D_i 是种群i 中的个体集合，d_j^i 是个体j，R_i 是第i次迭代中可用资源的集合，ω是一个缓存函数，其中包含第 $i-1$、$i-2$、.....、1次迭代中预先计算的元素；τ 是一个函数，用于返回考虑缓存和评估个体D_i后的计算时间。

根据公式1和2，我们应尽可能增大种群规模。这有助于通过横向扩展加快算法收敛，并改进其结果。为此，我们可以使用生产服务器上的远程计算，也可以利用同构和异构计算集群进行分布式计算。

远程计算允许将模型训练任务委托给远程基础设施。如果本地机器的计算资源不足以在合理时间内训练一组模型，则这种方法是合理的。远程计算可以在专用计算服务器上进行，也可以通过一组服务器组成的集群来完成；这些服务器使用 REST API、RPC 或消息队列接收模型训练任务。

所研究问题面临的主要挑战，是提出一种适用于各种计算基础设施的自动机器学习性能提升策略。图1中列出了五类环境：共享内存系统、采用分布式内存的多节点集群、复杂的同构与异构超级计算机环境，以及混合系统。具有结构(a)的系统能够以直接的方式执行并行任务。在系统(b)至(e)中，会涉及远程节点（同构或异构）。对于系统(e)，由于各远程节点的计算性能和连接开销各不相同，因此其结构属于混合型。基于此，计算策略的适应性尤为重要。

有多种方法可以使计算策略适应特定的基础设施。例如，经验性能模型 [9] 可用于为特定流水线的评估选择最优基础设施。拟合时间较短的简单流水线可以分配给性能较低的计算节点；反之，拟合时间较长的复杂流水线则可分配给性能较高的节点。因此，有必要开发一种模块化方法，以有效利用所有可用资源。

我们的主要动机是开发一种可用于自动机器学习计算层的方法。该层应能在同一种自动机器学习方法的框架内，适配指定的基础设施（本地或远程）。这种解决方案应具有高层次、模块化和灵活的特点，以便与不同的自动化机器学习工具集成。此外，它还应支持不同类型的流水线（从最简单的线性流水线到多层集成）。

4. 提出的改进

本节将介绍所提出的提升进化式自动机器学习计算性能的方法的各个方面。该方法的高层次方案如图2所示。本文探讨了四个主要方面：（1）适应度函数评估的并行化；（2）对已评估个体进行部分缓存；（3）结合中央处理器与图形处理器，加速个体处理；（4）为复杂任务集成远程基础设施。本文不讨论基于算法的改进（例如代理辅助优化）。

提出的方法的详细实现见算法1。在此表示法中，图表示复合流水线的结构。为更清晰地展示所提出的改进，省略了进化优化的细节。

算法1 所提出方法中实现的评估调度算法的高级伪代码。图中展示了并行化、缓存和评估阶段如何处理进化算法的一代。

```
1:  过程ProcessPopulation
2:      输入：
        个体（尚未评估的个体集合），
        目 标 （用于计算适应度的目标函数
        个体的），
        n （并行作业数）
        计时器 （计时器类对象）
        基础设施 （设置说明）
3:      输出：已评估的个体
4:      并行执行(n)
5:      如果计时器.时间足够( )则
6:          图 ← 个体[i].图 ▷获取每个个体的结构。
7:          如果基础设施.is remote( )则
8:              缓存 ← 分布式缓存()
9:              同步缓存                    ▷同步缓存数据库
10:             任务 ID ←创建任务(图)
11:             等待任务 ID
12:             inds[i].适应度 ← 请求结果(图)
13:         else
14:             准备图                      ▷将中央处理器和图形处理器分配给
        节点
15:             缓存 ← 本地缓存()
16:             加载 缓存                   ▷初始化缓存数据库
17:             如果缓存.exists(图)( ) 则
18:                 从缓存中拟合(图)
19:                 inds[i].适应度 ← 目标(图, 缓存)
20:             拟合(图)                    ▷拟合不在缓存中的节点
21:             保存 缓存                   ▷保留已更新的缓存
22:         如果 不inds[i]。有效 那么
23:             删除 inds[i]                ▷针对评估失败
24:     else
25:         删除个体[代]                    ▷时间不足，跳过
26:     返回个体                            ▷待选候选项
```

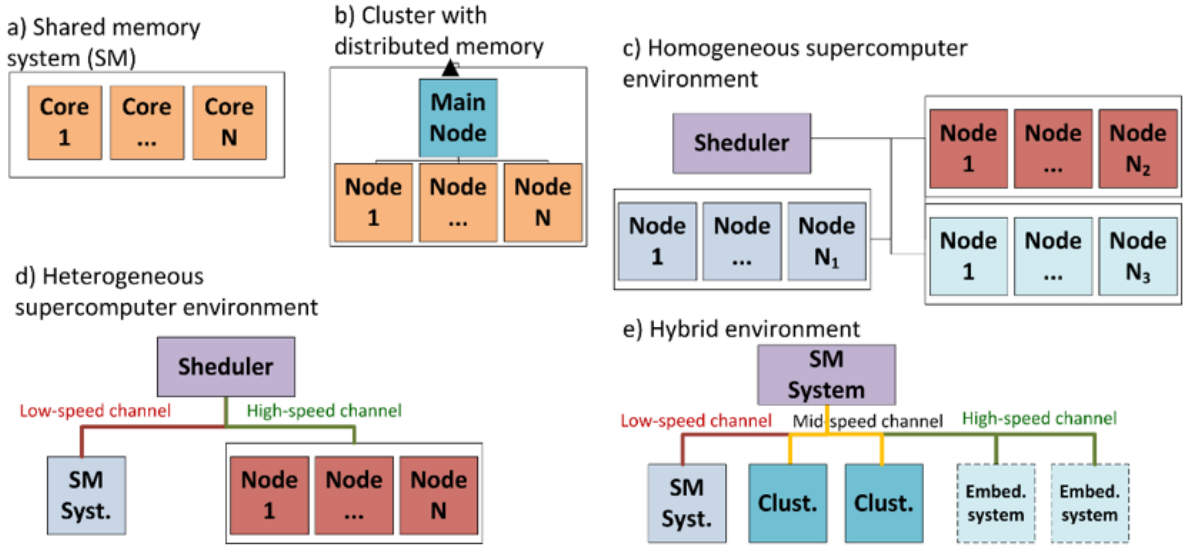


图1：可用于自动机器学习的不同类型计算基础设施：(a) 共享内存（SM）系统；(b) 采用分布式内存的多节点集群；(c) 具有空间分布式基础设施的复杂同构超级计算机系统；(d) 具有异构分布式基础设施的超级计算机系统。

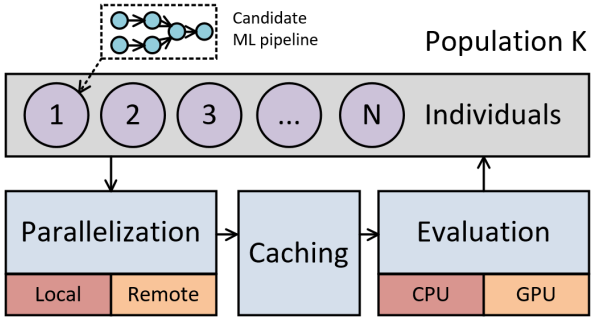


图2：所提出的提升复合自动化机器学习计算性能的方法的工作流程

4.1. 并行化

并行化进化算法并非新颖的想法。许多论文和开源解决方案都致力于解决这一问题。然而，自动机器学习中的并行化有其特殊性。例如，可以采用各种计算高效的并行进化策略 [9]。

我们考虑使用进化算法，在可表示为有向无环图的流水线空间中搜索最优解。并行化进化优化的经典方法是并发评估种群中的所有个体 [9]。这种方法之所以有效，是因为进化算法的特性：同一代中的个体之间不存在依赖关系。其他方法则建议将种群划分为彼此隔离的部分 [32]，或使用协同进化算法将任务拆分为子任务 [5]。

所提出的算法为每次流水线评估设定了最大评估时长，以解决由数据驱动模型训练的随机性所导致的潜在评估时间异常。如果训练过程未能至少在一个交叉验证折中收敛，相应适应度评估所需的时间可能会大幅增加。因此，为保持进化优化器的整体性能，会跳过评估耗时过长的个体。

4.2. 缓存

现有的缓存方法旨在保存并重用已拟合的流水线 [22]。然而，复合流水线中的各个节点可以单独缓存 [21]。这样便可重用拟合后的模型，并缩短适应度函数的评估时间。优化器可以在各个种群及个体之间共享内存缓存 [9]。不过，这也带来了内存消耗问题。

分析现有解决方案后，我们将重点放在用于流水线缓存的关系数据库方法上。具体而言，我们使用 sqlite3 库来实现该方法。首先，它只生成一个输出文件，而非关系型数据库（例如 shelve）无法保证这一点。其次，在适应度函数并行评估期间，所有并发的保存和加载操作都可以得到完整处理，无需直接使用同步原语、原子变量以及同时访问数据所需的其他工具。

最后，该方法支持同时提取多个操作，有助于提升缓存的整体性能。此外，将操作集保存到数据库时，会考虑是否存在主键相同的缓存项。

多层集成流水线的缓存过程应考虑到，缓存的模型/操作仅适用于建模流水线中先前节点和边的特定配置。因此，键包含对结构的递归描述

其中包含先前节点和边的递归描述。此外，还指定了交叉验证折的标识符。使用以下符号：*(/[节点名称] [超参数];)/[节点名称] [超参数]...*”，其中 / 表示节点名称的起始位置，圆括号表示嵌套的边。缓存详情如图3所示。

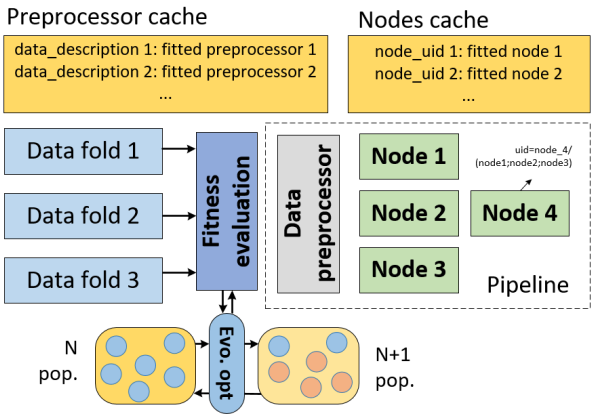


图3：操作缓存与待拟合建模流水线之间的交互

4.3. 图形处理器

使用图形处理器评估机器学习模型，是许多解决方案中发展成熟的一项功能。例如，RAPIDS 库 [34]包含 CuML 模块，可使用图形处理器训练分类、回归和聚类模型。要将此方法应用于复合流水线，我们应考虑一种设置：只有部分节点可以使用图形处理器进行评估。在这种情况下，流水线应以异构方式进行拟合。

所提出的方法通过将机器学习模型类型与其实现分离，使拟合时能够同时使用 CPU 和图形处理器。同一种模型（例如随机森林）可以有多种实现（基于 CPU 和基于 GPU）。

图4展示了一个计算上异构的复合模型结构示例。基于 GPU 的节点（黄色）之间的数据传输在视频内存中进行，而节点中的模型本身则在图形处理器（GPU）上训练。其他节点则在 CPU 上运行。

由于 RAPIDS 库存在多项软件限制（需要兼容 CUDA 的图形处理器驱动程序，且支持的操作系统范围有限），因此在基于 Docker 的容器中执行计算更为可行。

4.4. 远程评估

远程评估可以通过多种方式集成到进化优化器中。为了满足时间或内存限制，可以将数据集折和种群部分分配到多个计算节点上。由于进化算法并非总是需要处理大型数据集，我们重点关注远程计算的并行方面。所提出的实现依赖于 Kubernetes。REST API

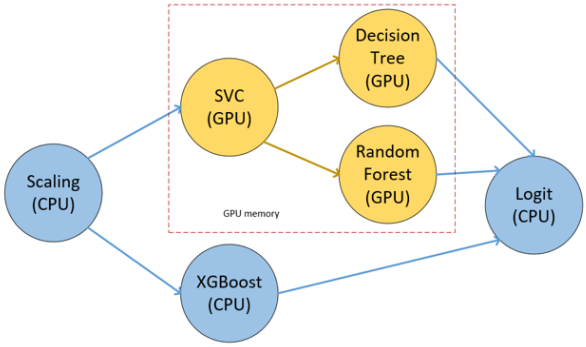


图4：可采用异构方式（中央处理器为蓝色，图形处理器为黄色）进行评估的流水线结构

Kubernetes 集群内的服务用于通过 HTTP 请求运行计算。客户端为请求实现了一个封装器。

在种群训练期间，评估器使用客户端的方法，在 Kubernetes 集群上处理个体。启动对所有个体的处理后，评估器会通过客户端方法等待计算完成。运行请求包含容器镜像、容器的资源限制、挂载路径和模型参数。REST API 服务会创建请求的容器并持续对其进行监控。客户端通过向 REST API 服务发送请求来获取容器的当前状态，以确认其仍在运行、已完成还是失败。

最后，训练完成后，客户端会下载拟合后的流水线。我们将结果打包成压缩归档文件，以减少通过网络传输的数据量。然后将文件发送给客户端。该处理流程如图5所示。

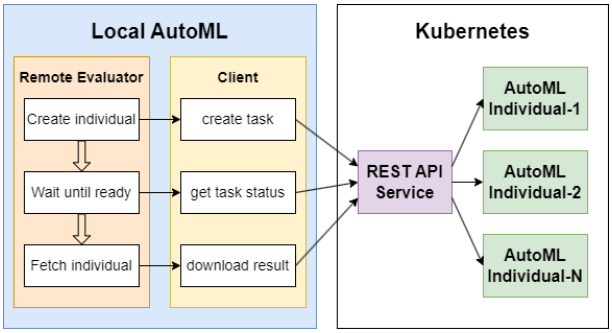


图5：自动机器学习与远程集群之间的通信

这样，我们可以将种群训练过程分为三个阶段：(1) 请求评估个体，(2) 计算并等待完成，以及 (3) 获取结果。

5. 软件实现

所提出的方法可作为以下架构的一部分：

• 模型存储库模块，用于存储和选择预测模型及数据处理模块的各种实现。一个模型可以包含多个实现（例如，适用于中央处理器和图形处理器的实现）；

• 复合模型生成式设计模块通过进化算法创建具有指定属性的模型。模型的属性由传递给优化器的目标函数决定。如果指定了多个目标函数（例如，在自动机器学习中，可以同时将训练时间和建模误差作为优化目标），则采用多准则形式的优化问题，其模型设计结果为包含各种折中方案的帕累托前沿。基因型以图形式表示，因此相应地实现交叉和变异算子。

• 在给定计算基础设施上运行的流水线执行模块。该模块支持在指定的计算节点上执行单个流水线。

该架构已在开源 FEDOT 框架的核心中实现。其实现的不同方面已在一系列论文中详细介绍： [19] 阐述了进化算子的主要方案及其实现， [28] 专门介绍了这种方法的多目标改进，而 [20] 则详细说明了复合建模流水线进化设计的各个方面。流水线超参数的调优策略基于贝叶斯优化。

可将自定义模型放入此节点中。如有必要，应手动指定模型的超参数搜索空间和初始近似值。这样便可以在自动机器学习中使用针对特定基础设施实现的模型。

下面的示例展示了从输入数据处理到获取预测结果的自动机器学习工作流。

```
api  =Fedot（问题 =’分类’，随机种子 =42，超时 =30，预设 =’图形处理器’）api.拟合（特征 =x_train，目标 =y_train）预测结果  =api.预测（特征 =x_test）
```

图 6 展示了用于实现各种评估策略的 UML 类图，这些策略支持在同一建模流水线中组合使用基于中央处理器和基于图形处理器的节点。高层建模方法（例如支持向量分类）可以通过不同算法实现：可从 scikit-learn 库 [27] 获取经中央处理器优化的 SVC 实现。相比之下，CUML 库提供了经图形处理器优化的实现。所提出的架构能够将这些细节隐藏在特定的建模流水线中，并对算法的不同实现采用相同的优化逻辑。

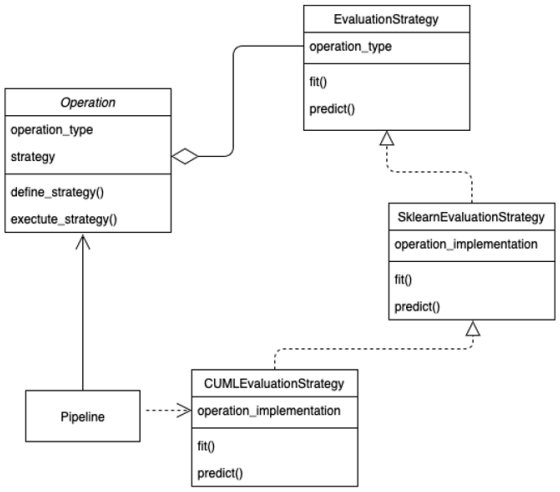


图 6：用于实现由多个操作组成的建模流水线的类图。*Operation* 类表示在操作中使用的高层建模策略。*EvaluationStrategy* 是该策略算法实现的基类。*SklearnEvaluation- Strategy* 表示基于 scikit-learn 库的实现，而 *CumlEvaluationStrategy* 表示基于 CUML 库的实现。

优化器将单个模型视为黑盒，通过输入、输出以及拟合/预测方法与之交互。流水线可使用以下构建模块：模型（伯努利朴素贝叶斯分类器、逻辑回归、多层感知机、随机森林、梯度提升、k近邻分类器、二次判别分析、线性判别分析、决策树）和数据转换操作（缩放、归一化、多项式特征变换、主成分分析、独立成分分析、孤立森林、重采样）。

6. 实验研究

我们开展了一系列实验，以验证所提出方法的正确性和有效性。这些实验可分为使用本地和远程基础设施的实验。我们采用了 OpenML 数据库中的多种分类数据集 [1] 和合成数据集作为基准（完整列表见表1）。每项实验均提供了计算基础设施的说明。

实验研究采用了以下方法：每项实验开始时，先将样本按 70% 和 30% 的比例分为“学习”样本和“验证”样本，以避免数据泄漏。随后，将学习样本传递给进化优化器。在优化过程中，采用五折交叉验证来估计适应度函数的值。

对每个数据集重复进行三次实验，以考虑优化器的随机性。质量指标取这些迭代的平均值。

6.1. 本地基础设施

对于使用本地基础设施进行的实验，我们配置了一台基于 Xeon Cascadelake（2900MHz）、配备 12 个核心的服务器

表1：实验中使用的 OpenML 数据集属性。随机森林模型用作训练时间估计的基线。

数据集 name	Rows, 10$\hat{3}$	特征	元素总数，10$\hat{3}$	基线训练。time, sec	Num. of clas ses
成人	49	14	684	12.5	2
亚马逊-员工-访问	33	9	295	1.5	2
澳大利亚人	0.69	15	10	0.2	2
银行-营销	45	17	769	6.9	2
血液-输血	0.75	5	4	0.1	2
car	1,8	7	12	0.5	4
cnae-9	1,1	857	926	14.7	9
斗兽棋_2pcs	45	7	314	48.0	3
Numerai28	96	22	2119	8.4	2
音位	54	6	32	0.12	2
Sylvine	51	21	108	0.5	2
Volkert	58	181	10554	128.4	10
合成-斑块	100	10	1000	6.2	2
合成-卫星	1	2	2	0.12	2

以及 24Gb 内存。

由于我们的方法声称，缓存可在拟合期间增加评估的流水线数量，因此将开启和关闭缓存选项时的这一指标进行比较是恰当的。为此，我们创建了基准测试，考察自动机器学习的不同计算配置。该基准测试使用 FEDOT 框架中的一个分类数据集作为测试基准。

图7展示了基于缓存和不使用缓存的配置之间的比较。对于前者，流水线缓存和数据预处理缓存均已启用。优化过程中使用的并行作业数为 1。

在进化优化过程中，会评估大量候选解（流水线）。我们针对不同的超时限制重复了实验，这些限制限定了整个 AutoML 运行的执行时间，因为它们会影响评估的流水线数量。此外，还对整个流水线设置了额外的时间限制（用于处理大型流水线的拟合时间异常）。该限制设为总超时时间的 1/4。

由于基于优化的实验具有随机性，每次运行都重复三次，并对所得指标取平均值。

图8中的结果是在n 个作业超参数值为 12 时获得的。

表2总结了单进程缓存和多进程缓存实验的平均指标。平均性能提升了14%，这从经验上证实了所提出方法的有效性，

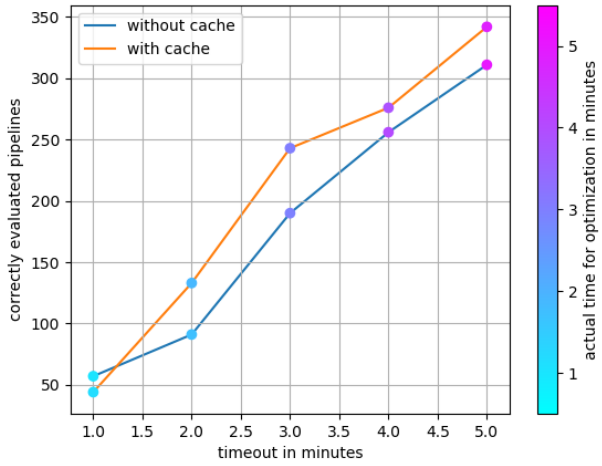


图7：流水线数量与缓存使用情况之间的关系（十次运行的平均值）。采用单进程处理。

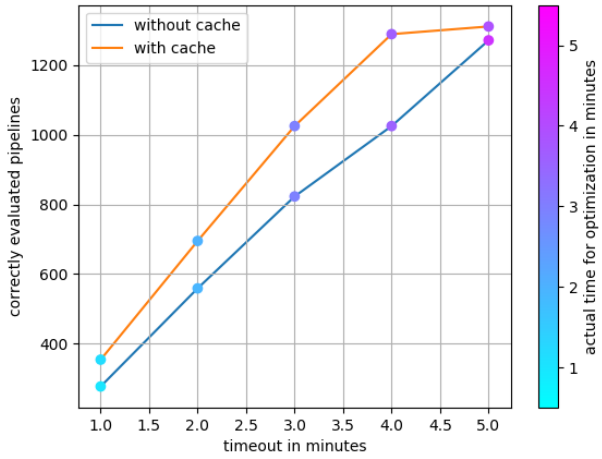


图8：流水线数量与缓存使用情况之间的关系（十次运行的平均值）。采用 8 个并行作业。

进一步证实了所提出方法的有效性。

实验的下一阶段旨在分析进化算法在多进程模式下的性能。我们比较了进程数分别为 1 和 8 时算法的性能，超时设为 10 分钟。为考虑随机性，流水线结构优化在不设定种子的情况下重复进行三次，并采用五折交叉验证。

图9展示了单个数据集上正确评估的流水线数量与指定作业数之间的依赖关系。可以看出，并行加速比近乎线性提升。图10展示了根据配置中的时间戳计算的交叉验证最佳适应度的变化情况。采用 8 个进程的运行比采用单个进程的运行更快地找到更优解。

表3总结了单进程模式和多进程模式下实验结果的平均值。使用交叉验证计算的适应度分数随评估的流水线数量线性增加。这证实了进化式自动化机器学习局部并行化的有效性。

表2：启用流水线节点和数据预处理操作缓存的实验结果。第一列表示是否使用了缓存，第二列表示并行进程的数量。接下来的三列表示不同的指标值（评估的流水线数量、验证样本的 ROC曲线下面积，以及训练样本交叉验证的 ROC曲线下面积）。

数据集	配置		流水线数量	最终ROC曲线下面积	交叉验证ROC曲线下面积
	缓存	进程数			
成人	on	1	**27**	0,92	0,9117
	off		23	**0,9213**	**0,913**
	on	8	**190**	0,921	0,9131
	off		170	**0,922**	**0,9137**
亚马逊员工访问 _	on	1	**85**	0,8447	0,8346
	off		78	**0,8497**	**0,8376**
	on	8	**416**	**0,8507**	0,8356
	off		369	0,849	**0,8398**
澳大利亚人	on	1	**879**	**0,9313**	**0,9432**
	off		838	0,9283	0,9401
	on	8	**6354**	0,928	0,9411
	off		6199	**0,934**	**0,9442**
银行营销	on	1	**38**	0,93	**0,931**
	off		30	**0,9313**	0,93
	on	8	205	0,931	**0,932**
	off		**211**	**0,932**	0,931
输血 -服务中心	on	1	**2175**	**0,748**	0,75
	off		2064	0,7383	**0,759**
	on	8	**13943**	0,745	0,761
	off		13834	**0,749**	**0,7659**
car	on	1	**812**	0,921	**0,933**
	off		728	**0,9233**	0,9319
	on	8	**4856**	**0,922**	**0,935**
	off		4608	0,92	0,934
cnae-9	on	1	**214**	**0,995**	**0,9939**
	off		195	**0,995**	**0,9939**
	on	8	1100	**0,995**	0,9942
	off		**1161**	**0,995**	**0,9953**
斗兽棋 2件 原始 _ -残局完整	on	1	30	**0,9671**	**0,9637**
	off		**44**	0,9667	0,9627
	on	8	89	0,969	0,9631
	off		**99**	**0,9713**	**0,9649**
Numerai28	on	1	3	0.508	0,51
	off		**6**	**0,511**	**0,5182**
	on	8	20	0,527	**0,528**
	off		**23**	**0,5273**	**0,528**
音位	on	1	**354**	**0,9599**	0,951
	off		325	0,9597	**0,9515**
	on	8	**1954**	**0,9631**	**0,955**
	off		1885	0,963	0,9547
Sylvine	on	1	**221**	0,9852	**0,9809**
	off		206	**0,9853**	0,9806
	on	8	**932**	**0,9878**	0,981
	off		827	0,9877	**0,9829**
Volkert	on	1	4	0,932	0,9298
	off		**6**	**0,9393**	**0,9344**
	on	8	20	0,9313	**0,934**
	off		**21**	**0,9317**	0,9273
合成斑块	on	1	**31**	**1**	**1**
	off		27	**1**	**1**
	on	8	**235**	**1**	**1**
	off		224	**1**	**1**
合成卫星	on	1	1124	**1**	**1**
	off		**1026**	**1**	**1**
	on	8	12356	**1**	**1**
	off		**12227**	**1**	**1**

表3：进化并行化实验的结果。每个数据集的前两行（1 和 8 个作业）表示对流水线设置拟合时间限制时获得的结果；“无上限”行表示在 8 个作业且无时间限制的情况下获得的结果。

数据集	进程数	流水线数量	ROC曲线下面积 最终	ROC曲线下面积 交叉验证
成人	1（有上限）	23	0,9213	0,913
	8（有上限）	**170**	**0,922**	0,9137
	8（无上限）	126	0,9217	**0,9141**
亚马逊员工访问 _	1（有上限）	78	0,8497	0,8376
	8（有上限）	**369**	**0,849**	**0,8398**
	8（无上限）	329	**0,849**	0,8387
澳大利亚人	1（有上限）	838	0,9283	0,9401
	8（有上限）	6199	**0,934**	0,9442
	8（无上限）	**10009**	0,9307	**0,9444**
银行营销	1（有上限）	30	0,9313	0,93
	8（有上限）	**211**	**0,932**	**0,931**
	8（无上限）	184	0,9313	0,93
输血服务中心	1（有上限）	2064	0,7383	0,759
	8（有上限）	**13834**	**0,749**	**0,7659**
	8（无上限）	8329	0,716	0,7658
car	1（有上限）	728	**0,9233**	0,9319
	8（有上限）	4608	0,92	0,934
	8（无上限）	**4612**	**0,925**	**0,9345**
cnae-9	1（有上限）	195	0,995	0,9939
	8（有上限）	1161	0,995	**0,9953**
	8（无上限）	**1168**	**0,995**	0,9949
斗兽棋 2件 原始残局 完整 _	1（有上限）	44	0,9667	0,9627
	8（有上限）	99	0,9713	0,9649
	8（无上限）	**144**	**0,9723**	**0,9661**
Numerai28	1（有上限）	6	0,511	0,5182
	8（有上限）	23	**0,5273**	**0,528**
	8（无上限）	**22**	0,5253	**0,5266**
音位	1（有上限）	325	0,9597	0,9515
	8（有上限）	1885	**0,963**	**0,9547**
	8（无上限）	**1917**	**0,963**	0,9536
Sylvine	1（有上限）	206	0,9853	0,9806
	8（有上限）	**827**	**0,9877**	**0,9829**
	8（无上限）	816	0,986	0,9821
Volkert	1（有上限）	6	0,9373	0,9329
	8（有上限）	**21**	**0,9393**	**0,9344**
	8（无上限）	**21**	0,9317	0,9273
合成斑块	1（有上限）	8	**1**	**1**
	8（有上限）	**37**	**1**	**1**
	8（无上限）	30	**1**	**1**
合成卫星	1（有上限）	1026	**1**	**1**
	8（有上限）	12227	**1**	**1**
	8（无上限）	**12569**	**1**	**1**

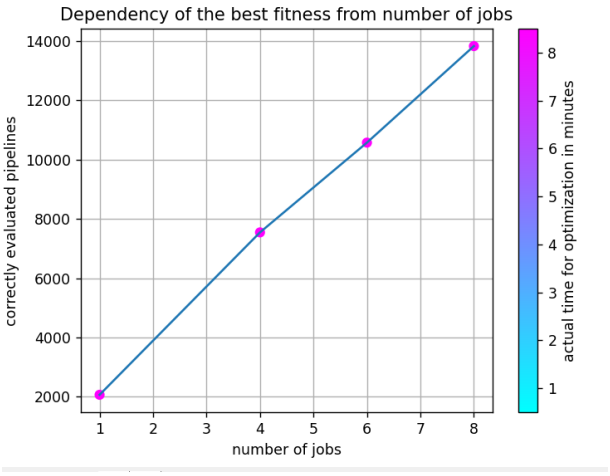


图9：流水线数量与作业数量之间依赖关系的详细分析（数据集：输血服务中心）

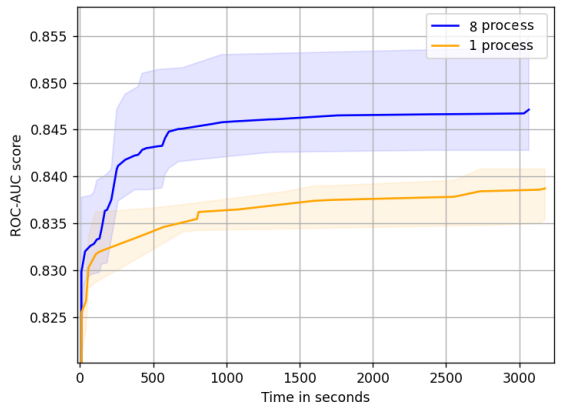


图10：最佳适应度值与优化时间之间的依赖关系。区间表示优化运行的随机性。

6.2. 异构基础设施

在下一系列实验中，我们旨在估算异构基础设施处理大型数据集的效率。计算实验在一个超级计算机环境中进行，该环境由两个 DGX-1 集群构成。每个集群包含八张 Tesla V100 图形处理器，显存为 128 GB。图形核心数量为 40960。

第一个实验比较了仅使用中央处理器的基础设施与混合基础设施在不同任务上的自动机器学习性能。该实验旨在估算引入基于 GPU 的节点后拟合时间的缩短幅度。为降低实验的计算复杂度，我们决定不使用表1中的全部数据集。本实验采用了四个合成二分类数据集，每个数据集包含 10 个特征，行数各不相同（10000、100000、200000 和 300000 行）。为估算流水线中模型之间数据流传输的开销，实验考察了单模型流水线（支持向量分类）和多模型流水线（由支持向量分类、逻辑回归和随机森林组成）。结果见

表4：不同条件下合成数据上的流水线训练时间（考虑单个支持向量分类器和包含多个模型的复合流水线）。文中给出了同构（采用多核 CPU 的单台服务器）和异构（CPU 和 GPU）计算环境下的平均效率估算。

Rows, 10^3	拟合时间，秒				提升，%	
	单个模型		计算流水线		单个模型	计算流水线
	CPU	CPU+GPU	CPU	CPU+GPU		
10	0.3	2.2	0.7	2.4	-	-
100	11.4	1.9	8.3	1.6	91	88
200	39.3	2.8	21	3.2	94	85
300	76.0	4.2	37.8	5.5	95	86

表4。

结果证实，对于少量数据，开销可能超过性能增益。不过，对于大型数据集，所提出的用于流水线评估的混合方法具有较强的实用性。

6.3. 远程基础设施

接下来的一系列实验采用了一个由 Kubernetes 管理、包含 20 个节点的同构集群。每个节点配备 40 个 CPU 核心和 256 Gb RAM。我们训练了种群规模分别为 50、100 和 200 的种群。每种规模的种群均训练四次，然后对估算的时间值取平均。本实验旨在分析远程评估的计算时间构成，并确认无论现有开销如何，远程评估对于大型数据集都切实可行。为使实验结果更加简洁，我们专注于分析一个包含 300000 行和 10 个特征的合成二元分类数据集。

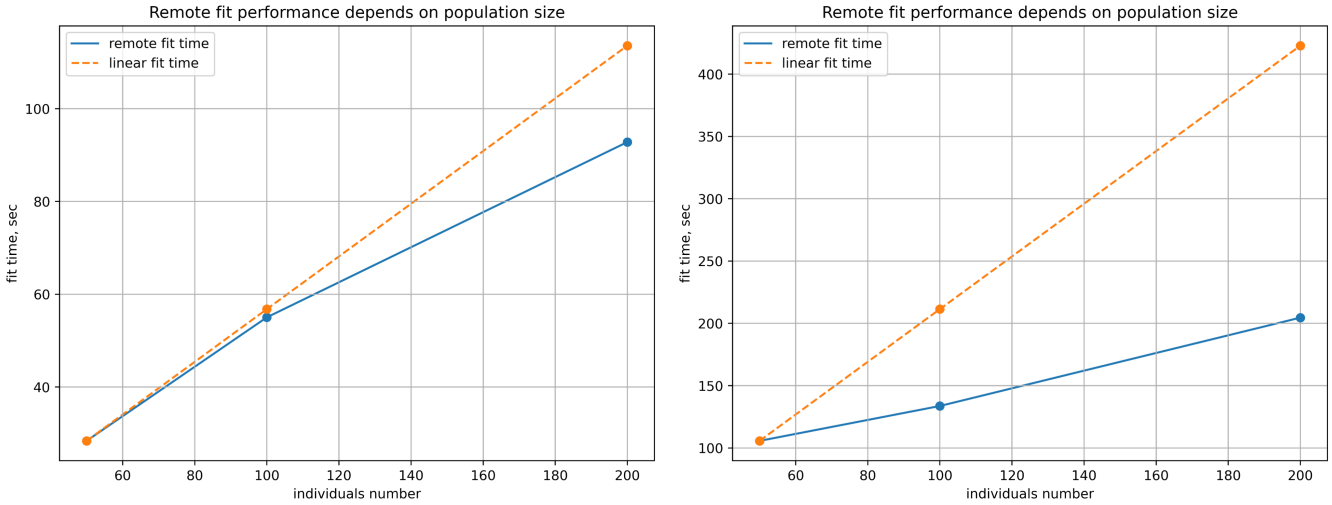
图11a展示了未获取结果时，不同种群规模对应的训练时间。对每个个体的评估不受 CPU 和系统内存限制。此外，我们绘制了一条线性拟合时间线作为参考线。该线的起点是种群规模为 50 时的时间。我们发现，随着种群规模增加，训练时间几乎呈线性增长。

为解释近乎线性的时间增长，我们可以考察计算过程中最耗时的操作及其开销。由于请求开销接近 0 秒，因此未在图12中展示。

图12显示，每个个体的计算时间和开销都相同，且与种群规模无关。这意味着性能瓶颈不在集群端。图13展示了拟合阶段时间线（包括结果获取）。

从时间线可以看出，大部分时间都花在结果获取上。结果获取包括通过网络下载 ZIP 文件、解压缩，然后反序列化模型。由于结果是并发获取的，大量个体会导致本地机器的网络和驱动器负载较高。

此外，尽管运行一个个体的请求开销不到 1 秒，但大量请求也会



(a) 无上限（快速计算）

(b) CPU 限制 = 0.2 核心（模拟繁重计算）

图11：不同计算设置下，总拟合时间与个体数量之间的关系。橙色线表示线性加速；蓝色线表示观测到的拟合时间。

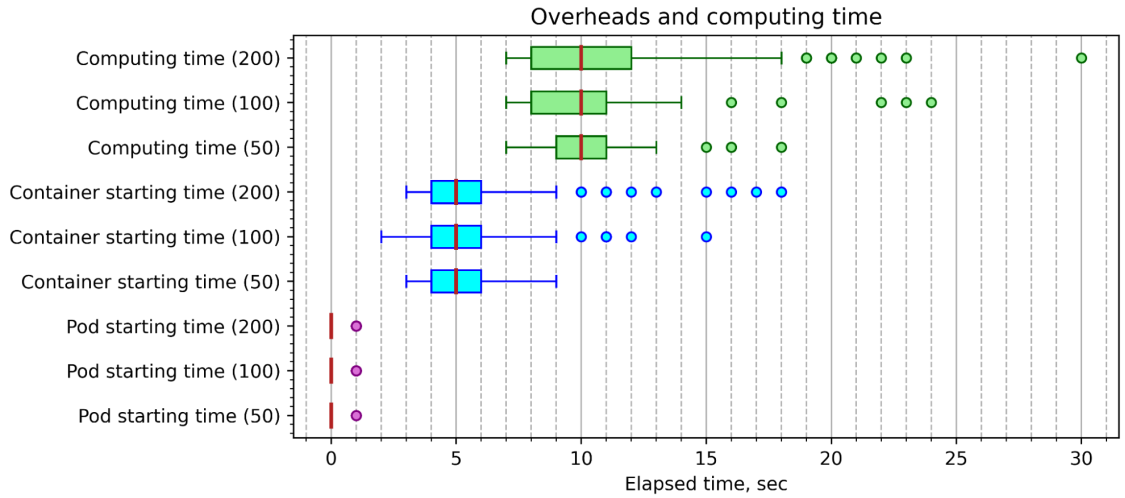


图12：使用远程基础设施进行实验时的开销和计算时间

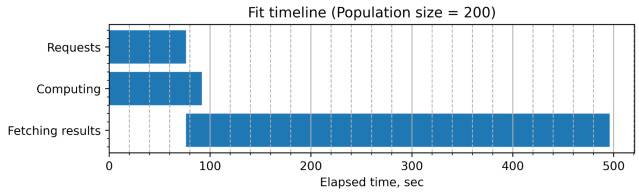


图13：远程基础设施下远程训练时间线说明。

耗费相当多的时间。最后一个请求与最后一个个体完成运行之间的时间间隔不到 20 秒，且处于计算时间的第 75 百分位以内。这意味着计算集群未得到充分利用，因为它等待客户端发送请求，未能像预期那样并行运行个体。

总而言之，如果计算轻量且快速，使用远程训练并不合理。请求和结果获取所产生的开销将远远大于有效载荷。为证明这一假设，我们重复了同一实验，但人为地为每个个体设置了 CPU 限制（最高为 0.2 个 CPU 核心），以模拟“繁重”计算。繁重任务的结果见图11b。

我们可以得出结论：远程计算能够显著加速高计算成本的计算任务。不过，对于较小的数据集，其开销也应予以考虑。

7. 结论与讨论

在本文中，我们提出了一种模块化方法，可提高异构环

境下进化式自动化机器学习的效率。与现有解决方案不同，所提出的方法可配置为适用于各种计算环境的自动化机器学习。它能够在混合和/或远程计算系统中并行化并分配计算任务。此外，我们还实现了缓存算法，以提高复合流水线的优化性能。

基于自动机器学习的实验设置包括：(1) 估算不同进程数量下的并行加速比；(2) 分析缓存效率；(3) 分析 GPU 计算效率；(4) 在使用远程基础设施的情况下运行优化。实验验证了所提出方法的效率。该方法能够显著增加已评估个体的数量，并改善适应度函数。

针对不同瓶颈，有几种提升远程计算性能的方法，可以单独使用，也可以组合使用：

1. 要高效利用集群资源，需要使用自定义调度器，并为 Volcano[5] 等批处理工作负载添加插件。

2. 不要为运行每个个体都单独发送请求。最好通过一个请求批量运行多个个体。这样一来，整个种群只需发送一个独立请求，从而减少请求数量。此外，当剩余个体数量较少时，可以采用推测计算模式；

3. 在本地机器上挂载集群文件系统。这样可以减少下载结果的请求次数，客户端也可以跳过 ZIP 文件解压步骤。客户端将直接从已挂载的文件系统中读取结果。如果无法实现这一点，那么我们不仅要实现批量运行请求，还要实现批量下载请求；

4. 也使用远程基础设施执行模型验证。这样，我们还必须向远程系统提供验证数据集。远程计算将验证个体并保存得分。这样就无需获取训练后的模型，计算出的得分也足以支持后续决策；

5. 异构环境。我们不仅可以在集群层使用异构环境，也可以在客户端-服务器层使用。例如，客户端可以在本地执行轻量级计算，同时将繁重的计算任务发送到集群，而最繁重的计算任务则可以发送到性能最强的集群节点（例如，专用图形处理器节点）。

另一个改进方向是支持大型数据集处理。可以通过实现数据集不同折的分布式评估来实现。缓存系统也可以采用分布式方式实现。

[5] https://volcano.sh

8. 代码和数据可用性

所有上述方法和算法的软件实现均可在公开代码库 https://github.com/ITMO-NSS-team/fedot-performance-improvement-benchmark 中获取。

参考文献

[1] Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., 和 Vanschoren, J.（2017）。OpenML基准测试套件与OpenML100。stat, 1050:11。[2] Cohen-Shapira, N. 和 Rokach, L.（2022）。用于自动机器学习算法选择的数据集表征学习。Knowledge and Information Systems, 64(10):2599–2635。[3] Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., 和 Smola, A.（2020）。AutoGluon-Tabular：面向结构化数据的稳健且准确的自动机器学习。arXiv预印本 arXiv:2003.06505。[4] Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., 和 Hutter, F.（2020）。Auto-sklearn 2.0：新一代。arXiv预印本 arXiv:2007.04074。[5] Gao, Y., Sun, Y., 和 Wu, J.（2016）。用于非线性混合整数规划问题的差分遗传协同进化算法。J. Nonlinear Sci. Appl, 9:1261–1284。[6] Gijsbers, P., Vanschoren, J., 和 Olson, R. S.（2018）。分层TPOT：加速基于树的流水线优化。arXiv预印本 arXiv:1801.06007。[7] Guerv´os, J.-J. M., Laredo, J. L. J., Castillo, P. A., Valdez, M. G., 和 Rojas-Galeano, S.（2019）。提升基于通道的进化算法的算法效率与性能。载于《遗传与进化计算会议伴会论文集》，第320–321页。[8] He, X., Zhao, K., 和 Chu, X.（2021）。自动机器学习：当前最先进技术综述。Knowledge-Based Systems, 212:106622。[9] Kalyuzhnaya, A. V., Nikitin, N. O., Hvatov, A., Maslyaev, M., Yachmenkov, M., 和 Boukhanovsky, A.（2020）。通过进化学习实现计算高效数学模型的生成式设计。Entropy, 23(1):28。[10] Karatsiolis, S. 和 Schizas, C. N.（2014）。为遗传编程算法实现软件缓存以缩短执行时间。载于 IJCCI (ECTA)，第259–265页。[11] Kim, K. J.（2009）。针对离散优化问题部分结果复用的有效分布式缓存策略。达特茅斯学院。[12] Kłosko, J., Benecki, M., Wcisło, G., Dajda, J., 和 Turek, W.（2022）。通过基于张量的加速实现高性能进化计算。载于《遗传与进化计算会议论文集》，第805–813页。[13] Kratica, J.（1999）。通过缓存提升遗传算法的性能。Computing and Informatics, 18(3):271–283。[14] Le, T. T., Fu, W., 和 Moore, J. H.（2020）。利用特征集选择器，将基于树的自动机器学习扩展至生物医学大数据。Bioinformatics, 36(1):250–256。[15] LeDell, E. 和 Poirier, S.（2020）。H2O AutoML：可扩展的自动机器学习。载于ICML自动机器学习研讨会论文集，第2020卷。[16] Li, Y., Wang, Z., Ding, B., 和 Zhang, C.（2021）。自动机器学习：产业与学界相遇的视角。载于第27届ACM SIGKDD知识发现与数据挖掘会议论文集，第4048–4049页。[17] Majidi, F., Openja, M., Khomh, F., 和 Li, H.（2022）。自动机器学习工具使用情况的实证研究。arXiv预印本 arXiv:2208.13116。[18] Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., 等（2018）。Ray：一种面向新兴 {人工智能} 应用的分布式框架。载于第13届USENIX操作系统设计与实现研讨会（OSDI 18），第561–577页。[19] Nikitin, N. O., Polonskaia, I. S., Vychuzhanin, P., Barabanova, I. V., 和 Kalyuzhnaya, A. V.（2020）。用于复合分类模型的结构进化学习。Procedia Computer Science, 178:414–423。[20] Nikitin, N. O., Vychuzhanin, P., Sarafanov, M., Polonskaia, I. S., Revin, I., Barabanova, I. V., Maximov, G., Kalyuzhnaya, A. V., 和 Boukhanovsky, A.（2021）。用于设计复合机器学习流水线的自动化进化方法。Future Generation Computer Systems。

[21] Nikitin, N. O., Vychuzhanin, P., Sarafanov, M., Polonskaia, I. S., Revin, I., Barabanova, I. V., Maximov, G., Kalyuzhnaya, A. V., 和 Boukhanovsky, A.（2022）。用于设计复合机器学习流水线的自动化进化方法。Future Generation Computer Systems, 127:109–125。[22] Olson, R. S. 和 Moore, J. H.（2016）。TPOT：用于自动化机器学习的基于树的流水线优化工具。载于自动机器学习研讨会论文集，第66–74页。PMLR。[23] Orzechowski, P. 和 Moore, a. H.（2019a）。利用双聚类挖掘海量RNA-seq数据集：进化方法是否已准备好应对大数据? 载于《遗传与进化计算会议伴会论文集》，第304–305页。[24] Orzechowski, P. 和 Moore, J. H.（2019b）。提升进化双聚类算法EBIC性能的策略。载于《遗传与进化计算会议伴会论文集》，第185–186页。[25] Packard, N., Bedau, M. A., Channon, A., Ikegami, T., Rasmussen, S., Stanley, K., 和 Taylor, T.（2019）。开放式进化与开放性：开放式进化专题特刊的编者引言。[26] Parmentier, L., Nicol, O., Jourdan, L., 和 Kessaci, M.-E.（2019）。TPOT-SH：一种用于在大型数据集上解决自动化机器学习问题的更快优化算法。载于2019年IEEE第31届人工智能工具国际会议（ICTAI），第471–478页。IEEE。[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., 和 Duchesnay, E.（2011）。scikit-learn：Python中的机器学习。机器学习研究杂志, 12:2825–2830。[28] Polonskaia, I. S., Nikitin, N. O., Revin, I., Vychuzhanin, P., 和 Kalyuzhnaya, A. V.（2021）。复合数据驱动模型的多目标进化设计。载于2021年IEEE进化计算大会（CEC），第926–933页。[29] Przewozniczek, M. W. 和 Komarnicki, M. M.（2021）。适应度缓存——从次要机制到现代进化计算中的重大影响。载于2021年IEEE进化计算大会（CEC），第1785–1791页。IEEE。[30] Santos, E. E. 和 Santos Jr, E.（2001）。遗传算法中的有效且高效缓存。国际人工智能工具杂志, 10(01n02):273–301。[31] Santu, S. K. K., Hassan, M., Smith, M. J., Xu, L., Zhai, C., Veeramachaneni, K., 等（2020）。自动机器学习：迄今进展与未来展望——挑战与机遇。arXiv预印本 arXiv:2010.10777。[32] Silva, F. J. M. D., P´erez, J. M. S., Pulido, J. A. G., 和 Rodr´ıguez, M. A.（2010）。并行Alineaga：一种用于多序列比对的岛屿并行进化算法。载于《2010年软计算与模式识别国际会议论文集》，SoCPaR 2010，第279–284页。[33] Singh, V. K. 和 Joshi, K.（2022）。自动机器学习（AutoML）：应用与研究机遇概述。Journal of Information Technology Case and Application Research，第1–11页。[34] Team, R. D.（2018）。RAPIDS：用于端到端GPU数据科学的库集合。[35] Tornede, T., Tornede, A., Hanselle, J., Wever, M., Mohr, F., 和 H¨ullermeier, E.（2021）。迈向绿色自动化机器学习：现状与未来方向。arXiv预印本 arXiv:2111.05850。[36] Vakhrushev, A., Ryzhkov, A., Savchenko, M., Simakov, D., Damdinov, R., 和 Tuzhilin, A.（2021）。LightAutoML：面向大型金融服务生态系统的自动机器学习解决方案。arXiv预印本 arXiv:2109.01528。[37] Visheratin, A., Struckov, A., Yufa, S., Muratov, A., Nasonov, D., Butakov, N., Kuznetsov, Y., 和 May, M.（2020）。Peregreen——用于在云环境中高效存储历史时间序列的模块化数据库。载于2020年USENIX年度技术会议（USENIX ATC 20），第589–601页。[38] Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., 等（2018）。借助MLFlow加速机器学习生命周期。IEEE Data Eng. Bull., 41(4):39–45。[39] Z¨oller, M.-A. 和 Huber, M. F.（2019）。自动机器学习框架的基准测试与综述。arXiv预印本 arXiv:1904.12054。