# LatentTune：通过潜在表示学习高效调优高维数据库参数*

Sein Kwon* 计算机科学延世大学首尔，韩国 seinkwon97@yonsei.ac.kr

Youngwan Jo 计算机科学延世大学首尔，韩国 jyy1551@yonsei.ac.kr

Seungyeon Choi 计算机科学延世大学首尔，韩国 tmddus1553@yonsei.ac.kr

Jieun Lee 计算机科学延世大学首尔，韩国 jieun199624@yonsei.ac.kr

Huijun Jin 计算机科学延世大学首尔，韩国 jinjuijun@yonsei.ac.kr

Sanghyun Park† 计算机科学延世大学首尔，韩国 sanghyun@yonsei.ac.kr

arXiv:2602.04190v1 [cs.DB] 4 Feb 2026

摘要—**随着数据量持续增长，优化数据库性能变得愈发重要，因此实施有效的调优方法至关重要。在各种方法中，数据库参数调优已被证明是提升性能的有效手段。近期研究表明，机器学习技术能够成功优化数据库参数，从而显著提升性能。然而，现有方法仍存在若干局限。首先，它们需要耗费大量时间生成大型训练数据集。其次，为应对高维优化的挑战，它们通常只优化部分参数，而非整个配置空间。第三，它们往往依赖相似工作负载的信息，而不是直接利用目标工作负载的信息。为解决这些局限，我们提出了一种与传统方法有根本不同的新方法 LatentTune。为缩短数据生成所需时间，LatentTune采用数据增强策略。此外，它构建了一个潜在空间，将所有数据库参数的信息压缩其中，从而能够优化整个配置空间。另外，LatentTune将外部指标信息整合到潜在空间中，从而针对实际目标工作负载进行精准调优。实验结果表明，在 MySQL 和 RocksDB 的四种工作负载上，LatentTune 的表现优于基线模型；对于 RocksDB，性能最高提升 1332%；对于 MySQL，吞吐量提升 11.82%，延迟降低 46.01%**。

索引词——**数据库参数调优、贝叶斯优化、数据库管理系统、机器学习**

## 一、引言

近年来，数据量显著增长，高性能数据库管理系统 (DBMS) 高效处理大量数据的需求也随之增加 [1]。提升 DBMS 性能的方法之一是数据库参数调优 [2],，即通过调整参数值来提升数据库性能的过程。传统的数据库参数调优需要数据库管理员 (DBA) 修改参数值；然而，由于众多参数之间的关系错综复杂，很难高效地调优参数。此外，不同的数据库具有不同的参数，数据库管理员 (DBA) 必须考虑这些差异以及工作负载的多样性。为克服这些局限，近期研究采用机器学习 (ML) 技术开展自动数据库参数调优，在各种工作负载和数据库管理系统中取得了性能提升 [3]–[8]。这些研究使用了多种基于机器学习的优化算法，包括贝叶斯优化 (BO) [9] 和强化学习 (RL) [10] ，以更有效地引导调优过程。

**现有方法的局限性**。机器学习模型需要大量由配置及其对应性能指标组成的训练数据，才能有效训练 [11]。在传统方法中，通常通过对配置值进行采样，并使用基准测试工具评估其性能来生成此类数据集 [12]–[14]。

然而，这一过程十分耗时，难以高效收集数据。例如，在 MySQL 中将每种配置的基准测试时长限制为 300 秒时 [15],，生成包含 1,000 种配置的数据集仍需耗费大量时间，这凸显了创建足够大的训练数据集所面临的实际限制。

为应对高维优化，参数选择过程也带来了另一项挑战。为缩小搜索空间，现有的基于贝叶斯优化的方法 [3]–[6] 通常采用 Shapley 加性解释（SHAP） [16] 和 LASSO [17] 等选择算法，找出对性能影响最大的前$k$ 个参数，并仅对这些参数进行优化。另一方面，基于强化学习的方法 [7], [8],利用深度确定性策略梯度（DDPG） [18] 等强化学习算法，通过学习有效的动作和状态表示来应对高维连续空间。

然而，即使在相同的工作负载下，所选参数也可能因算法而异。如表 I 所示，对于 RocksDB [19],，SHAP 和 LASSO 算法选出的参数仅有三到五个重叠，排名顺序也不同。图 1 展示了使用 RGPE [20], 这一基线模型对

*第一作者。 †通讯作者。

表 I：使用 SHAP 和 LASSO 分析 RocksDB 的 R90W10 和 R50W50 工作负载时排名前十的参数。重叠参数以粗体突出显示。

| R90W10 | | R50W50 | |
|---|---|---|---|
| **SHAP（前10）** | **LASSO（前十名）** | **SHAP（前10）** | **LASSO（前十名）** |
| **目标文件大小基数** | 压缩类型 | **压缩比** | **压缩比** |
| 写缓冲区大小 | 目标文件大小乘数 | 写缓冲区大小 | **最大后台刷新次数** |
| 0级文件数量压缩触发阈值 | **层级数量** | 最大写缓冲区数量 | 层级数量 |
| 合并所需的最小写缓冲区数量 | **目标文件大小基数** | 合并所需的最小写缓冲区数量 | **目标文件大小倍数** |
| 压缩比 | 布隆局部性 | Level-0 文件数量压缩触发阈值 | 布隆局部性 |
| 块大小 | **最大后台压缩数** | 基础层最大字节数 | **MemTable布隆过滤器大小比例** |
| **最大后台压缩数** | 最大后台刷新数 | 压缩优先级 | 0 级停止写入阈值 |
| **层级数量** | 0 级停止写入阈值 | **目标文件大小乘数** | 最大后台压缩数 |
| **MemTable布隆过滤器大小比例** | **MemTable布隆过滤器大小比例** | **0 级停止写入阈值** | 0 级写入减速阈值 |
| **0 级写入减速阈值** | **0 级写入减速阈值** | **最大后台刷新数** | 块大小 |

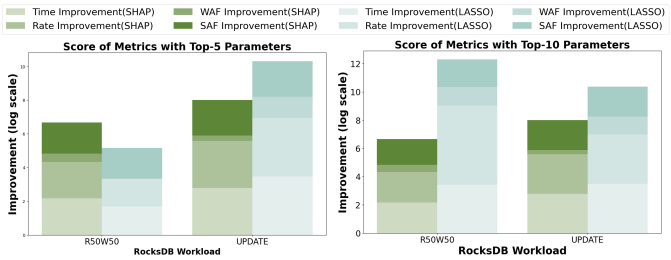



图 1：RocksDB 上前 5 名和前 10 名参数带来的性能提升。

排名前 5 和前 10 的参数进行调优所带来的性能影响。结果表明，参数选择的差异可能导致性能出现显著差异。此外，仅关注排名前$k$ 的参数，可能无法捕捉其余参数之间的关键交互作用，也可能忽略对数据库性能有影响但不易察觉的效应，最终降低调优结果的稳定性。

除了这些挑战之外，适应多样化的工作负载仍然是数据库参数调优中的一大难题。一些现有方法试图通过计算其与目标工作负载的相似度，从数据存储库中识别最相似的工作负载。然而，这种方法并不直接针对目标工作负载本身进行优化，而不准确的工作负载映射可能因假定的工作负载特征与实际特征不匹配而导致次优的调优结果。即使两个工作负载在查询延迟或缓存命中率等内部指标上看起来相似，更深层次的结构差异（包括查询模式、事务大小和读写比的变化）也会显著影响哪些参数对优化最为关键。由于许多内部指标主要捕捉的是低层系统行为而非高层工作负载结构，仅依赖以往的工作负载数据可能导致调优无效，最终降低数据库性能。这凸显了对一种更具目标针对性的优化方法的需求，该方法应直接利用来自实际工作负载的性能反馈。

**我们的方法**。为解决这些局限性，我们提出了**LatentTune**，这是一个通过潜在表示学习高效调优高维数据库参数的框架。当训练数据集有限时，LatentTune 使用拉丁超立方抽样进行数据增强（LHS） [21] 和 TabNet [22]。与传统方法不同，我们的方法无需通过基准测试来构建配置与指标数据对。为应对高维参数空间带来的挑战，LatentTune 利用自编码器（AE） [23] 将所有参数配置压缩到低维潜在空间，从而能够联合优化所有参数。此外，通过将目标工作负载信息直接注入潜在空间，LatentTune 支持针对特定工作负载进行调优，无需依赖基于相似度的工作负载映射。这种整合使其能够在多种工作负载场景下实现更准确、更有效的调优。我们的主要贡献如下：

- LatentTune通过采用数据增强技术，缩短了数据集生成所需的时间。
- 我们提出了一种名为 LatentTune 的新型框架，通过将完整的配置参数集映射到紧凑的潜在空间，解决数据库参数调优中的高维度挑战。这使得全参数优化成为可能，有别于依赖于仅选择部分参数的传统方法。

- LatentTune 将目标工作负载信息直接融入潜在空间，从而实现工作负载特定调优，以适应多样的工作负载特征。

- 与各种基线模型的比较实验结果表明，LatentTune 能够实现更优的性能。

## II. 预备知识

### A. 自动数据库参数调优

数据库参数调优是一种通过优化数据库参数（如内存分配、磁盘存储、缓存大小等）来提升数据库性能的方法。数据库配置是指决定数据库系统运行方式的设置和参数。如果参数表示为 $P$，则配置可定义为 $\text{conf} = \{P_1, \ldots, P_j\}$，其中 $j$ 为参数的数量。图 2 展示了基于机器学习的数据库参数调优的基本流程。调优过程的整体流程如下所述：

◯ **1 调优请求**。当控制器收到有关数据库管理系统和待调优工作负载的信息后，便会启动

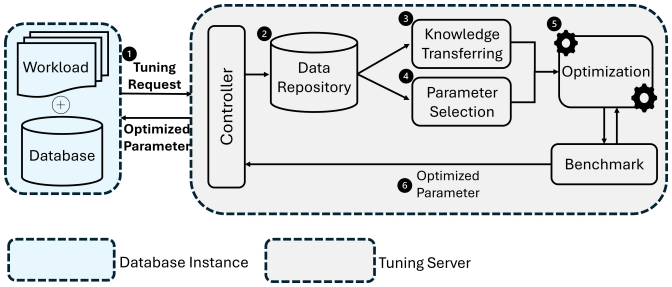



图 2：数据库参数调优架构

调优过程。**2 数据存储库**。在调优过程中，调优请求中提供的数据库管理系统和工作负载信息会存储在数据存储库中。 **3知识迁移**。为优化目标工作负载，该过程会使用欧几里得距离等距离度量指标，计算目标工作负载与数据存储库中已存储工作负载之间的相似度 [24]。随后，利用最相似的工作负载信息来指导调优过程。此过程旨在促进多样化工作负载的高效调优。 **4 参数选择**。为应对高维搜索空间中的优化挑战，参数选择算法会识别对数据库性能影响最大的参数。**5 优化**。优化算法（例如贝叶斯优化或强化学习）会对对数据库性能有显著影响的前$k$ 个参数（4）以及目标工作负载信息（3）进行优化。 **6 优化后的参数**。优化后的参数会传递给控制器，随后控制器将这些参数应用于实际数据库。

### *B.* 相关工作

自动数据库参数调优技术分为两类：基于搜索的方法 [25] 和基于机器学习的方法 [5]–[8]。基于搜索的方法无需详细了解数据库系统的内部信息即可进行调优，而基于机器学习的方法则使用机器学习模型调优参数。

**基于搜索的方法**。 基于搜索的调优是一种通过依据预定义规则寻找数据库参数来优化数据库性能的技术。BestConfig [25]使用分割与发散采样（DDS）在高维参数空间中采样参数，并使用递归界限与搜索（RBS）在其周围的有界空间内寻找最优配置。然而，基于搜索的方法在数据质量较低时可能变得不稳定，而且依赖启发式技术，这些技术往往非常耗时。

**基于机器学习的方法**。基于机器学习的方法使用机器学习模型优化参数。例如，OtterTune [5] 会在高维搜索空间中选择对性能至关重要的参数进行优化，并使用贝叶斯优化和高斯过程（GP） [26]来提升性能。然而，由于 OtterTune 优化的参数集合会因所用参数选择算法的不同而变化，因此其性能可能不稳定。CDBTune [7] 使用强化学习，具体而言是深度确定性策略梯度 [18], , 该方法融合了深度Q网络 [27]和演员-评论家 [28]方法以便在高维空间中进行优化。然而，CDBTune 需要大量的训练时间和基准测试，导致调优耗时较长。排序加权高斯过程集成（RGPE） [20] 是一种用于基于贝叶斯优化的优化的集成模型，它结合有关工作负载的信息和历史知识来优化这一过程。然而，RGPE 在优化高维数据时面临挑战，与 OtterTune 遇到的问题类似：选择用于优化的参数并不能保证性能稳定。此外，RGPE 在为模型设置合适的超参数值方面也存在局限。

## III. 方法

如图 3 所示，我们提出的框架由多个模块组成。为了解决数据库配置调优中难以收集大量训练数据这一挑战，我们的方法首先对少量已测量的配置及其对应的性能指标进行数据增强。通过利用原始样本和增强样本，我们训练出一个低维表示空间，以捕捉配置参数与性能结果之间的基本关系。编码器将高维配置压缩到紧凑的潜在空间中，而解码器则将其重构回原始配置空间。最后，在潜在空间中执行贝叶斯优化（BO），以高效搜索性能优异的配置，并将最优潜在向量解码为完整的配置集以供部署。各模块的更多细节将在以下小节中介绍。

### *A.* 数据增强

基于机器学习的方法需要大量训练数据来提升模型的准确性。然而，在数据库调优研究领域，由于各种硬件条件以及每个数据库系统所处理的特定工作负载（任务或查询集合）的重大影响，目前没有可用的公共数据集。因此，必须针对每种情况单独生成训练数据，但这需要消耗大量资源和时间。为解决这一局限，我们采用样本增强流程。

样本增强模块对配置与指标数据对进行增强。我们将原始数据集定义为 $\mathcal{D} = \{（配置_i，\ y_i）\}_{i=1}^{N}$，其中每个配置 $配置_i \in \mathbb{R}^d$ 是一个由数据库参数构成的 $d$ 维空间，而 $y_i \in \mathbb{R}^e$是一个由外部性能指标构成的 $e$ 维向量。同样，增强后的配置与指标数据对表示为 $配置'_i \in \mathbb{R}^d$ 和 $y'_i \in \mathbb{R}^e$。最终得到的增强数据集表示为 $\hat{} \mathcal{D} = \{(配置'_i，\ y'_i)\}_{i=1}^{M}$。

在配置采样阶段，我们采用拉丁超立方抽样 [21] 生成配置数据 $conf'$。拉丁超立方抽样是一种分层抽样技术，能够对多维参数空间进行均匀且高效的探索。它可确保样本在每个维度上均匀分布，从而实现对整个输入空间的均衡覆盖。这

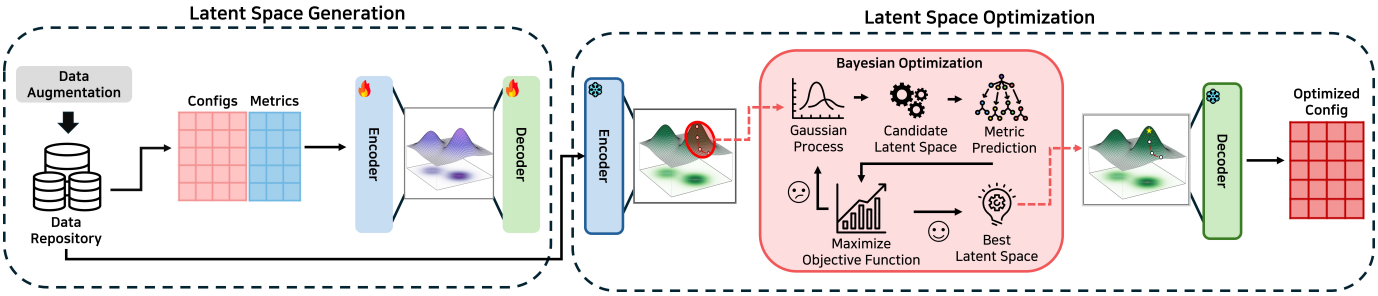



图 3：LatentTune 的总体架构

种系统化方法能够更准确地反映各个参数及其组合对性能指标 [20], [29]–[31]的影响。相比之下，SimpleRandomSampling [32] 和高斯采样 [33]等传统方法在高维空间中的覆盖效果通常较差，尤其是在参数范围较宽时。这些方法往往会产生聚集或稀疏的样本，导致对配置空间的探索存在偏差或效率低下。

鉴于这些局限性，LHS 能够更加均匀且全面地覆盖搜索空间，因此特别适用于数据库参数调优场景。因此，我们采用 LHS 在框架中生成配置样本。

使用基准测试工具测量度量指标值的过程非常耗时，并且资源消耗极大。因此，我们利用预测模型来预测 $conf'$ 的度量指标值。具体而言，我们采用 TabNet [22], ，该模型适用于表格数据，并且能够高效地进行多目标标签预测。TabNet 采用基于掩码的注意力机制，能够在众多列中有效识别并优先处理重要特征，从而通过关注影响较大的属性并抑制信息量较低的属性来提升预测性能。例如，数据库配置数据通常包含大量参数及其对应值，并以表格格式组织。TabNet 系统地识别并学习每个参数的重要性，从而能够准确预测外部性能指标。

TabNet 使用原始数据集 $\mathcal{D}$ 进行训练，并用于预测增强配置数据 $conf'$ 的度量指标值 $y'$ 。随后，将 LHS 抽样得到的配置 $conf'$ 与相应的预测值 $y'$ 结合，构建增强数据集 $\hat{\mathcal{D}}$ 。TabNet 的预测准确性将在第 IV-C 节中进行评估。

### *B.* 潜在空间生成

我们利用自编码器 [23] 构建潜在空间，以捕捉数据库配置与外部指标之间的内在关系。在潜在空间优化（LSO）阶段，使用训练好的自编码器的解码器，将优化后的潜在向量重构为原始高维配置空间中的配置，从而能够直接部署得到的配置。

**训练。** 为生成信息丰富的潜在空间，LatentTune 使用原始数据集 $\mathcal{D}$ 和增强后的数据集 $\hat{\mathcal{D}}$训练自编码器。每个训练样本由一个配置及其对应的性能指标拼接而成，构成输入数据 $I \in \mathbb{R}^t$，其中 $t = d + e$，$d$ 和 $e$ 分别表示配置和指标的维度。拼接配置与指标的目的是将工作负载信息注入潜在空间。由于配置由任意数值构成，仅使用配置数据很难在潜在空间中提取有意义的语义表示。因此，将配置及其对应的性能指标拼接作为输入，使潜在空间能够捕捉参数与性能之间的依赖关系。该方法还使潜在空间能够隐式编码有关不同目标工作负载的信息。

自编码器经过训练，以最小化输入与其解码输出之间的重构损失。具体而言，它最小化以下目标：

$$\mathcal{L}_{\text{AE}} = \frac{1}{N+M} \sum_{i=1}^{N+M} \|x_i - \mathcal{D}_\theta(\mathcal{E}_\phi(x_i))\|^2 \qquad (1)$$

这里，$x_i$ 表示输入数据（拼接后的配置和度量指标），$\mathcal{E}(\cdot)$和 $\mathcal{D}(\cdot)$分别表示编码器和解码器。训练完成后，编码器将高维输入数据压缩到较低维的潜在空间中，而解码器则将其重构回原始输入数据空间。随后，LSO过程在该潜在空间中进行优化，并对优化后的潜在向量进行解码，以生成最终的配置候选项。

**编码器。** $\mathcal{E}: \mathbb{R}^t \to \mathbb{R}^l$ 将高维输入数据 $I$映射为低维潜在表示 $z \in \mathbb{R}^l$，其中 $l \ll t$，在降低维度的同时保留关键特征。**解码器。** $\mathcal{D}: \mathbb{R}^l \to \mathbb{R}^t$ 将潜在空间重构回原始高维空间。具体而言，LSO 步骤得到的优化后潜在向量 $z^*$ 经训练好的解码器重构，生成由配置及其对应度量指标组成的最终输出。

*C.* 潜在空间优化。

通过 LSG 步骤提取的潜在空间 $z \in [0,1]$ 由贝叶斯优化 (BO) [9]进行优化。贝叶斯优化是一种概率优化方法，旨在高效探索和利用黑箱函数的搜索空间，尤其适用于函数评估计算成本高昂且没有显式函数形式的情形。在每次迭代中，贝叶斯优化通过迭代构建概率代理模型，并利用平衡探索与利用的采集函数选择候选点，以最大化黑箱目标函数的值。

在此过程中，我们使用高斯过程（GP） [26] 作为潜在空间中的代理模型，以捕捉目标的后验分布。给定候选潜在向量$z$，我们使用期望改进（EI）采集函数，该函数量化相对于当前观测到的最佳性能的预期改进。形式化地，

$$\mathrm{EI}(z) = (\mu(z) - f(z^*) - \xi) \cdot \Phi(\gamma(z)) + \sigma(z) \cdot \phi(\gamma(z)) \quad (2)$$

其中， $\mu(z)$ 和 $\sigma(z)$ 分别是 GP 预测的均值和标准差，而 $f(z^*)$ 是迄今为止观测到的最佳性能。归一化改进 $\gamma(z)$的计算方式如下：

$$\gamma(z) = \frac{\mu(z) - f(z^*) - \xi}{\sigma(z)} \quad (3)$$

其中， $\xi$ 是一个较小的正数，用于平衡探索与利用。

目标函数通常由基准测试工具评估性能（即指标）。然而，由于基准测试耗时较长，我们实现了一个基于 TabNet 的性能预测模型。需要注意的是，该模型与样本增强步骤中使用的 TabNet 模型不共享权重。

TabNet 在通过 LSG 过程生成的潜在空间 $z$及其对应的指标 $y$ 和 $y'$ 上进行训练。换言之，预测模型的输入是潜在空间，输出是度量指标值。由于 TabNet 的预测结果是多标签的，因此定义了一个评分函数，将这些预测值转换为标量函数，以便进行更平滑的比较。优化过程的目标是最大化该评分函数。

对于 MySQL，主要目标是在最大化吞吐量的同时最小化延迟。为体现这一目标，我们将评分函数定义如下：

$$f_{\mathrm{MySQL}}(x) = \frac{\mathrm{Throughput}(x)}{\mathrm{Latency}(x)} \quad (4)$$

对于 RocksDB，必须同时优化多个性能指标。具体而言，给定四个性能指标（时间、速率、写放大因子和空间放大因子），各指标的详细信息见第 IV-A 节“评估指标”。我们将评分函数定义为：

$$f_{\mathrm{RocksDB}}(x) = -w_0 \cdot \mathrm{TIME} + w_1 \cdot \mathrm{RATE} - w_2 \cdot \mathrm{WAF} - w_3 \cdot \mathrm{SAF} \quad (5)$$

其中，TIME、WAF 和 SAF 是需要最小化的指标，而 RATE 是需要最大化的指标。

表 II：MySQL 工作负载信息

| **MySQL** | | | | | | | |
|---|---|---|---|---|---|---|---|
| 工作负载索引 | 规模因子 | Data Size | 读 | 插入 | 扫描 | 更新 | Read 修改写 |
| A | 12000 | 15GB | **50%** | - | - | **50%** | - |
| B | | | **95%** | - | - | **5%** | - |
| E | | | - | **5%** | **95%** | - | - |
| F | | | **50%** | - | - | - | **50%** |

$w_0$， $w_1$， $w_2$，而 $w_3$ 代表分配给这四个指标的权重。

## IV. 实验

*A.* 实验设置

**硬件**。 所有 MySQL 性能测量均在一台服务器上进行，该服务器运行 Ubuntu 20.04.6 操作系统，配备 Intel® CoreTM i7-11700 @ 2.52GHZ、32 GB 内存和 256 GB 磁盘。所有 RocksDB 性能测量均在一台运行 CentOS 7.9 操作系统的服务器上进行，该服务器配备 Intel® CoreTM i5-8500 CPU @ 3.00 GHz、32 GB 内存和 256 GB 磁盘。

**调优设置**。 我们使用 MySQL v5.7.37 [15]和 Rocks-DB v6.25.0 [19]评估性能。通过 OtterTune，我们在 MySQL 中选取了 138 个可调参数，不包括与调试、安全或路径设置相关的旋钮。初始数据集包含 1,000 个样本，通过数据增强过程扩充至 5,000 个样本。第 4.3 节讨论了增强样本的最佳数量。对于每种工作负载，我们使用不同的随机种子值重复实验五次，并计算平均值。在基线实验中，用于调优的前-$k$ 个选定参数的数量分别设为 5、10 和 15，并使用性能最佳的值进行比较。

**工作负载**。我们针对 MySQL 和 RocksDB 各自的四种不同工作负载进行了实验。所有工作负载的详细信息见表 II 和表 III。**YCSB**（Yahoo! 云服务基准测试） [34] 是一种基准测试工具，提供多种工作负载，每种工作负载都定义了数据库上各种操作（加载、插入、读、更新、扫描等）的速率和模式。它可用于比较和评估各种数据库系统（包括 MySQL）的性能。针对 RocksDB 工作负载，我们使用读、写和更新操作占比各不相同的四种工作负载进行了实验。

**评估指标**。 我们使用以下指标来比较基线与 Latent-Tune 之间的性能提升。**MySQL**采用两项评估指标：吞吐量，表示在给定时间内能够完成的查询处理量；延迟，表示处理查询所需的等待时间。

表 III：RocksDB 工作负载信息

| RocksDB | | | | | |
|---|---|---|---|---|---|
| 工作负载<br>索引 | 值大小 | 条目数量 | 读 | 写 | 更新 |
| R90W10 | 16384 | 65472 | 90% | 10% | X |
| R50W50 | | | 50% | 50% | |
| R10W90 | | | 10% | 90% | |
| 更新 | | | - | - | O |

吞吐量越高、延迟越低，表示性能越好；延迟按第99百分位数进行测量。**RocksDB** 的性能指标包括 TIME，用于衡量给定工作负载下完成数据库操作所需的时间；RATE，用于衡量每秒处理的操作数；写放大因子（WAF），表示特定配置下工作负载的写放大比例；以及空间放大因子（SAF），表示数据存储过程中产生的空间放大比例。TIME、WAF 和 SAF 值越低、RATE 值越高，性能越好。为比较 RocksDB 中这四项指标的性能提升率，我们计算了以下得分：

$$SCORE = \log\left(\frac{\text{TIME}_{\text{default}}}{\text{TIME}}\right) + \log\left(\frac{\text{RATE}}{\text{RATE}_{\text{default}}}\right) + \log\left(\frac{\text{WAF}_{\text{default}}}{\text{WAF}}\right) + \log\left(\frac{\text{SAF}_{\text{default}}}{\text{SAF}}\right) \quad (6)$$

该分数表示优化参数相较于默认设置带来的性能提升，并将各项提升率取对数后相加，以便于比较。

**实现细节**。我们的模型在 MySQL 中映射到 32 维，在 RocksDB 中映射到 16 维。我们在实验中为所有贝叶斯优化过程设置了相同的 300 次迭代。

*B.* 效率比较

我们在 MySQL 和 RocksDB 的四种工作负载上，将 LatentTune 与 OtterTune、CDBTune 和 RGPE 进行了比较。三个基线方法均根据对数据库性能的影响，选取部分排名靠前的参数进行调优，且每个模型都包含一个工作负载映射过程。

图4展示了 MySQL 上的实验结果。结果表明，在所有工作负载中，LatentTune 在吞吐量和延迟方面均实现了最高的提升率。对于工作负载 B，LatentTune 和基线方法的性能提升都很小。这是因为工作负载 B 是一个以读操作为主、由轻量级操作组成的工作负载，其本身对系统造成的压力很小。因此，数据库配置的变化对吞吐量的影响有限。然而，并发控制和 I/O调度相关的参数仍会显著影响延迟，从而使 LatentTune 实现明显的延迟改善。




图 4：MySQL YCSB A、B、E、F 工作负载的性能提升

图5展示了 RocksDB 上的实验结果。为便于清晰比较性能提升，使用评分函数表示相对于默认配置的性能提升。对于全部四种工作负载，LatentTune 均表现出更好的性能，最高得分为9.19，这一结果表明 LatentTune 的性能优于基线模型。

此外，与基线方法相比，我们观察到四项指标均有所提升，从而能够兼顾数据库多个方面的优化。实验结果表明，利用所有参数的信息进行优化，比仅调优最重要的参数更有效，因为后者忽略了未选参数的影响。此外，与三种基线方法不同，LatentTune 无需经过工作负载映射过程，就能调优各种工作负载。这是因为目标工作负载信息被注入了低维空间。

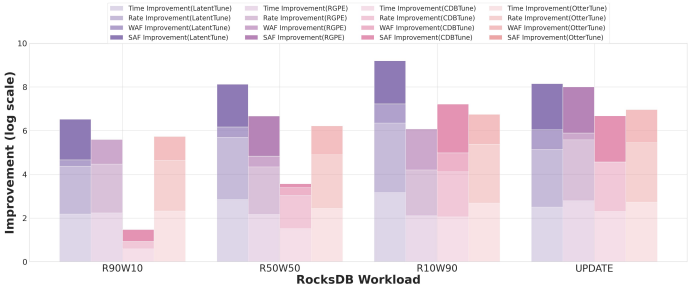



图5：RocksDB 工作负载的性能提升

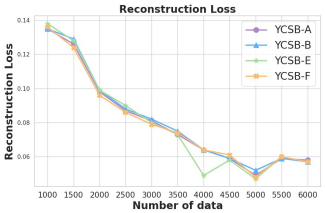



图6：重构损失随数据量的变化。

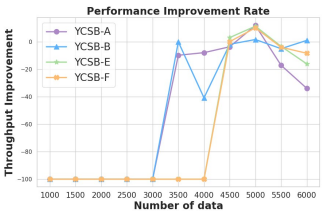



图7：性能提升随数据量的变化。

### *C.* 样本增强分析

*1*）基于数据量的性能比较

为展示数据增强在基于机器学习的调优过程中的效果，我们比较了自编码器的重构损失，以及 LatentTune 基于数据量取得的性能提升率。图 6 展示了自编码器的重构损失随训练数据数量的变化情况。重构损失指自编码器的输入数据与解码器输出数据之间的偏差。换言之，损失越低，重构性能越好。实验结果表明，随着训练数据数量增加，自编码器的重构损失逐渐降低；模型使用 5,000 个数据集进行训练时，损失最低。当数据点数量超过 5,500 时，重构损失开始上升，这是因为数据中的信息有限，自编码器的学习能力也受到约束。这意味着，即使采用数据增强，模型能够学习的相关信息量仍存在上限；如果超过这一阈值，自编码器的性能就可能下降。此外，当增强的数据量很大时，原始数据所占的相对比例会降低。因此，自编码器压缩至潜在空间中的信息准确性会下降，进而导致重构损失上升。这也会使 LatentTune 获得的性能提升随之下降。

图7展示了 LatentTune 的吞吐量提升速率随训练数据量变化的情况。与重构损失最低时的情况类似，训练数据量为 5,000 时，提升速率最高。

总而言之，当数据增强至总计 5,000 个数据集时，自编码器的性能最佳。此外，经过充分训练的自编码器生成的潜在空间能够成功压缩有关数据库参数和目标工作负载的信息。这些结果也证实，LatentTune 通过优化潜在空间来最大限度地提升数据库性能。因此，在配置模型实验时，我们将增强后的数据样本数量设为 5,000。

*2*）样本增强的预测模型

我们比较了样本增强过程中基准测试阶段所用预测模型的准确率。我们选取了五种模型进行比较：基础多层感知机（多层感知机） [35], 两种基于树的模型 [36], [37],以及两种采用提升技术的模型 [38], [39]。

如表 IV 所示，随机森林或 XGBoost 在 MySQL 上以较少的预测标签取得了最高准确率。然而，如表 IV 右侧所示，RocksDB 的结果表明，TabNet 在四个标签上取得了更高的准确率。尽管随机森林在 MySQL 上表现出良好的准确率，但在预测 RocksDB 的性能时，它未能在所有指标上取得一致的预测准确率。其原因在于，TabNet 在对多标签任务进行预测时，会为每个标签动态分配重要特征。因此，我们选择 TabNet 进行样本增强，因为它即使面对多标签数据，也能在所有标签上保持始终较高的预测准确率。

### *D.* 潜在空间分析

*1)* 潜在空间优化的效率

为评估所提出的潜在空间优化方法的有效性，我们在 YCSB A 和 YCSB F 两种不同工作负载下，分析了 MySQL 中优化配置与非优化配置的潜在向量。

如图 8 所示，等高线密度图表明，在两种工作负载中，未优化配置（红色）和优化配置（蓝色）的潜在向量形成了明显不同的簇。这种分离表明，潜在空间优化能够有效捕获并保留高维参数空间中有意义的结构，确保优化配置在潜在空间的特定区域内得到良好表示。此外，两个簇之间的明显区分也表明了我们提出的潜在空间优化方法的效率。为进一步验证这一点，我们使用经过训练的自编码器的解码器，将潜在空间表示重构到原始高维空间，并评估其实际性能。图 9 展示了两种工作负载下吞吐量-延迟关系的等高线密度图。优化配置（蓝色）表现为高吞吐量和低延迟，符合性能最优区域的特征。相比之下，未优化配置（红色）集中在次优区域，其特点是吞吐量较低、延迟较高。等高线密度的明显差异证实，我们的方法成功优化了配置，而且这一规律在不同工作负载下均能保持一致。

图 10 分别展示了吞吐量与延迟的四分位距（IQR），作为对性能的详细分析

表 IV：样本增强中 MySQL（YCSB E、YCSB F）和 RocksDB（R10W90、更新）的指标预测模型性能。

| YCSB E | | | | | R10W90 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 模型 | 吞吐量 | | 延迟 | | TIME | | RATE | | WAF | | SAF | |
| | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) |
| MLP | 0.08 | 0.810 | 0.09 | 0.780 | 0.45 | 0.0060 | 0.31 | 0.00900 | 0.40 | 0.0030 | 0.45 | 0.06000 |
| 决策树 | 0.84 | 0.140 | 0.83 | 0.144 | 0.83 | 0.0010 | 0.87 | 0.00100 | 0.50 | 0.0020 | 0.99 | 0.00005 |
| 随机森林 0.99 | | 0.006 | 0.96 | 0.026 | 0.85 | 0.0007 | 0.93 | 0.00090 | 0.76 | 0.0010 | 0.99 | 0.00002 |
| XGBoost | **0.99** | **0.001** | **0.97** | **0.022** | 0.88 | 0.0005 | 0.93 | 0.00090 | 0.86 | 0.0007 | 0.99 | 0.00003 |
| CatBoost | 0.98 | 0.016 | 0.95 | 0.033 | 0.89 | 0.0005 | 0.92 | 0.00080 | 0.87 | 0.0007 | 0.99 | 0.00007 |
| TabNet | 0.98 | 0.010 | 0.95 | 0.035 | **0.90** | **0.0001** | **0.95** | **0.00003** | **0.87** | **0.0003** | **0.99** | **0.00001** |
| **YCSB F** | | | | | **更新** | | | | | | | |
| 模型 | 吞吐量 | | 延迟 | | TIME | | RATE | | WAF | | SAF | |
| | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) |
| MLP | 0.17 | 0.871 | 0.16 | 0.900 | 0.46 | 0.0030 | 0.35 | 0.007 | 0.51 | 0.030 | 0.46 | 0.0500 |
| 决策树 | 0.98 | 0.018 | 0.95 | 0.045 | 0.80 | 0.0010 | 0.76 | 0.002 | 0.29 | 0.020 | 0.98 | 0.0004 |
| 随机森林 **0.98** | | **0.012** | **0.97** | **0.022** | 0.90 | 0.0005 | 0.87 | 0.001 | 0.68 | 0.003 | 0.99 | 0.0005 |
| XGBoost | 0.98 | 0.017 | 0.97 | 0.023 | 0.92 | 0.0004 | 0.87 | 0.001 | 0.75 | 0.003 | 0.99 | 0.0006 |
| CatBoost | 0.97 | 0.021 | 0.96 | 0.037 | 0.92 | 0.0005 | 0.85 | 0.005 | 0.80 | 0.002 | 0.99 | 0.0005 |
| TabNet | 0.96 | 0.043 | 0.93 | 0.070 | **0.94** | **0.0002** | **0.87** | **0.001** | **0.83** | **0.002** | **0.99** | **0.0004** |

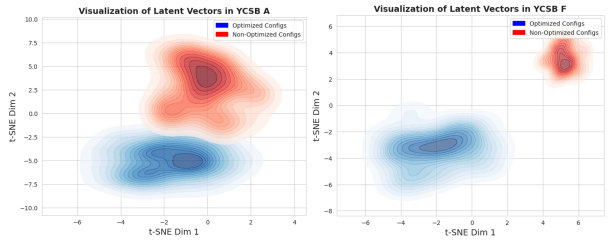



图 8：YCSB A 和 YCSB F 工作负载上的潜在向量密度。

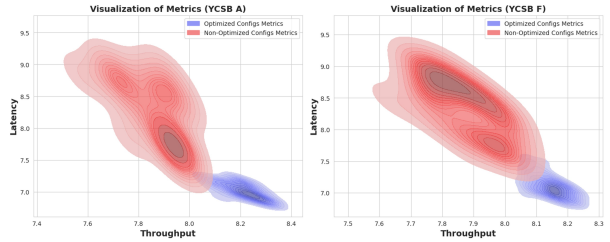



图 9：YCSB A 和 YCSB F 工作负载上的指标密度。

指标如图 9 所示。结果进一步证实，在两种工作负载中，优化配置相比未优化配置均实现了显著更高的吞吐量与更低的延迟。特别是，优化配置（蓝色）表现出更低的方差且无离群值，表明优化过程同时提升了性能与稳定性。

*2）LSO 中目标函数的预测模型*

在 LSO 阶段，我们没有在目标函数中进行实际基准测试，而是替换为优化配置的指标预测模型。我们在与表 IV 中实验相同的设置下，对优化潜在空间的预测模型准确性进行了比较。

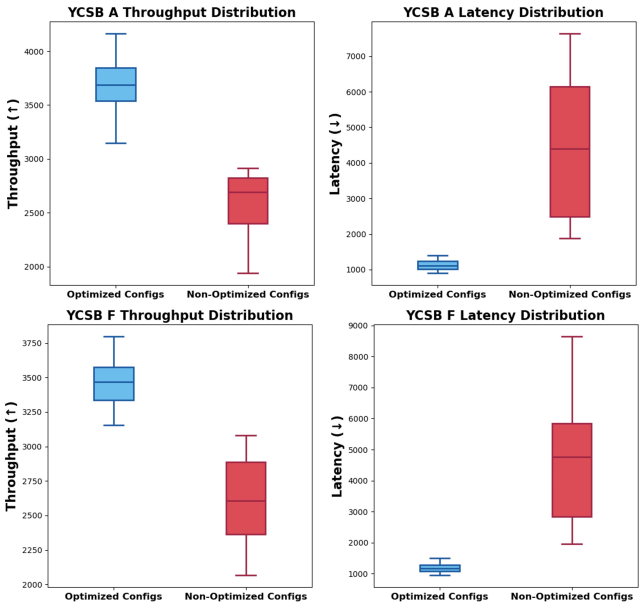



图 10：YCSB A（上）和 YCSB F（下）工作负载上的指标分布。

如表 V 所示，其他预测模型要么性能较差，要么无法在所有标签上准确预测。然而，TabNet 在预测 MySQL 和 RocksDB 的指标时取得了最高准确率，并在所有标签上表现出一致的准确率。预测模型的输入数据采用表格数据形式，其中行包含有关配置和指标的信息，列表示原始数据的压缩关键特征。由于 TabNet 非常适合表格数据预测，它优于五个对比模型，并且

表 V：贝叶斯优化中 MySQL（YCSB E、YCSB F）和 RocksDB（R90W10、R10W90）的指标预测模型性能比较。

| YCSB E | | | | | R90W10 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 模型 | 吞吐量 | | 延迟 | | TIME | | RATE | | WAF | | SAF | |
| | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) |
| MLP | 0.28 | 0.850 | 0.24 | 0.600 | 0.41 | 0.010 | 0.48 | 0.010 | 0.48 | 0.006 | 0.78 | 0.020 |
| 决策树 | -0.47 | 0.080 | -0.51 | 0.040 | -0.30 | 0.040 | -0.26 | 0.030 | -0.29 | 0.170 | 0.22 | 0.080 |
| 随机森林 0.18 | | 0.040 | 0.16 | 0.020 | 0.37 | 0.010 | 0.43 | 0.010 | 0.41 | 0.007 | 0.68 | 0.003 |
| XGBoost | 0.16 | 0.050 | 0.14 | 0.020 | 0.44 | 0.010 | 0.50 | 0.010 | 0.59 | 0.005 | 0.74 | 0.020 |
| CatBoost | 0.45 | 0.030 | 0.42 | 0.010 | 0.49 | 0.010 | 0.54 | 0.010 | 0.61 | 0.005 | 0.79 | 0.020 |
| TabNet | **0.85** | **0.008** | **0.79** | **0.005** | **0.80** | **0.004** | **0.72** | **0.005** | **0.75** | **0.003** | **0.96** | **0.003** |
| **YCSB F** | | | | | **R10W90** | | | | | | | |
| 模型 | 吞吐量 | | 延迟 | | TIME | | RATE | | WAF | | SAF | |
| | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) | 决定系数(↑) | 均方误差(↓) |
| MLP | 0.024 | 0.070 | 0.023 | 0.030 | 0.65 | 0.002 | 0.61 | 0.0070 | 0.18 | 0.002 | 0.82 | 0.010 |
| 决策树 | -0.34 | 0.120 | -0.39 | 0.400 | 0.09 | 0.005 | 0.28 | 0.0080 | -0.83 | 0.006 | 0.34 | 0.070 |
| 随机森林 0.21 | | 0.040 | 0.20 | 0.020 | 0.58 | 0.002 | 0.69 | 0.0030 | 0.13 | 0.003 | 0.74 | 0.020 |
| XGBoost | 0.29 | 0.040 | 0.23 | 0.020 | 0.62 | 0.002 | 0.77 | 0.0080 | 0.16 | 0.002 | 0.78 | 0.002 |
| CatBoost | 0.42 | 0.030 | 0.40 | 0.010 | 0.47 | 0.001 | 0.81 | 0.0020 | 0.22 | 0.002 | 0.73 | 0.020 |
| TabNet | **0.92** | **0.004** | **0.86** | **0.004** | **0.72** | **0.001** | **0.93** | **0.0007** | **0.78** | **0.001** | **0.79** | **0.002** |

在所有标签上表现出同等的性能。

*E.* 调优开销与效率分析

表 VI 比较了两种 MySQL 工作负载下的调优时间。尽管 RelTune 并未实现最短的调优时间，但其整体优化性能最佳。RelTune 的时间成本相对较高，原因是需要通过自编码器进行额外训练以构建潜在空间。不过，由于贝叶斯优化（BO）中的目标函数被轻量级预测模型取代，因此这项成本仍然适中。这一结果凸显了调优效率与优化性能之间的权衡：RelTune 略微增加了预处理时间，但显著加快了向最优配置收敛的速度。此外，RelTune 的数据效率更高，使用更少的基准样本便能取得更好的结果。在实际应用中，这种适度的额外开销是可以接受的，因为调优稳定性和可复现性的提升所带来的收益超过了额外的计算成本。

*F.* 消融研究

我们比较了设计中的各个模块在不同工作负载下的影响。实验采用了样本增强和 LSG 的多种组合。所有组合均使用贝叶斯优化（BO）进行优化，且优化过程的迭代次数相同。

表 VII 展示了在 MySQL 上进行的消融实验结果。各组合的性能提升表明，LatentTune 及其模块的设计在所有工作负载下均有效。LatentTune 带来了最佳的性能提升。总体而言，不包含样本增强和 LSG 的组合性能很差，因为在高维空间中进行贝叶斯优化会导致维度灾难。

表 V：完成各工作负载调优所需的时间（分钟）。越低越好。

| ***方法*** | **YCSB A** | **YCSB B** | **YCSB E** | **YCSB F** |
|---|---|---|---|---|
| OtterTune | **48.52** | **50.16** | **49.70** | **48.24** |
| CDBTune | 362.21 | 370.15 | 381.54 | 421.85 |
| RGPE | 62.13 | 68.46 | 67.51 | 64.24 |
| LatentTune | 57.23 | 60.43 | 59.1 | 58.53 |

这种维度灾难阻碍了贝叶斯优化中代理模型的学习，并显著降低了探索效率。因此，第一个组合在所有工作负载下的性能都不如默认配置。

仅应用数据增强模块时，数据量的整体增加使代理模型得以训练，因此性能相较第一种组合略有提升。然而，由于维度灾难，性能仍与默认情况相近。

仅应用 LSG 模块时，维度的降低缓解了维度灾难。然而，由于训练自编码器同样需要大量数据，在没有数据增强的有限数据下，潜在空间未能得到良好压缩。换言之，它未能有效压缩数据中的重要信息。这进而降低了贝叶斯优化中用于替代目标函数的预测模型的性能，导致优化配置在所有组合中表现最差。

## V. 结论

本文解决了基于机器学习的数据库参数调优的三个局限：需要大型训练数据集，

表 VII：使用 YCSB A、B 和 E、F 对 MySQL 进行消融研究。“Aug”指数据增强，“LSG”表示潜在空间生成。

**(a) YCSB A 和 YCSB B**

| 模块 | | YCSB A | | YCSB B | |
|---|---|---|---|---|---|
| Aug | LSG | 吞吐量 | 延迟 | 吞吐量 | 延迟 |
| X | X | 3651.22 | 1550 | 7448.72 | 714 |
| O | X | 3689.79 | 1281 | 9952.72 | 611 |
| X | O | 3314.65 | 1443 | 6570.68 | 903 |
| O | O | **4100.34** | **1024** | **10000.00** | **318** |
| 默认 | | 3666.90 | 1591 | 9845.56 | 589 |

**(b) YCSB E 和 YCSB F**

| 模块 | | YCSB E | | YCSB F | |
|---|---|---|---|---|---|
| Aug | LSG | 吞吐量 | 延迟 | 吞吐量 | 延迟 |
| X | X | 3570.29 | 1588 | 3049.66 | 1220 |
| O | X | 3996.52 | 1261 | 2934.99 | 1476 |
| X | O | 3439.19 | 1782 | 2196.67 | 7001 |
| O | O | **4100.45** | **1226** | **3600.76** | **1200** |
| 默认 | | 3685.17 | 1549 | 3271.47 | 1791 |

由于维度较高，往往只调优参数的一个子集，并且依赖相似工作负载而非实际的目标工作负载。为克服这些挑战，我们提出了 LatentTune，一种将降维与数据增强相结合的基于潜在空间的调优框架。这使得能够利用小型数据集进行有效训练，并实现全参数优化。通过将工作负载特定信息注入潜在空间，LatentTune 还能更好地适应多样化工作负载。

实验结果表明，LatentTune 显著优于现有基线，在 RocksDB 上实现了高达 1332% 的提升，并在 MySQL 上获得了 11.82% 的吞吐量提升和 46.01% 的延迟改善。在未来工作中，我们计划扩展 LatentTune，以支持动态环境中的实时调优，从而更接近在生产数据库系统中的实际部署。

致谢

本研究得到韩国政府（科学技术信息通信部）资助的国家研究基金会（NRF）支持（编号：RS-2023-00229822）。

参考文献

[1]X. Zhou、C. Chai、G. Li 和 J. Sun，《数据库与人工智能相遇：综述》，*IEEE*知识与数据工程汇刊，第 34 卷，第 3 期，第 1096–1116 页，2020 年。[2] X. Zhao、X. Zhou 和 G. Li，《数据库旋钮自动调优：综述》，*IEEE*知识与数据工程汇刊，2023 年。[3] S. Duan、V. Thummala 和 S. Babu，《使用 iTuned 调优数据库配置参数》，*VLDB*基金会会议录，第 2 卷，第 1 期，第 1246–1257 页，2009 年。[4] A. Fekry、L. Carata、T. Pasquier、A. Rice 和 A. Hopper，《Tuneful：面向大数据分析的在线显著性感知配置调优器》，*arXiv* 预印本 *arXiv:2001.08002*，2020 年。[5] D. Van Aken、A. Pavlo、G. J. Gordon 和 B. Zhang，《通过大规模机器学习实现数据库管理系统自动调优》，载于*2017* 年 *ACM* 数据管理国际会议论文集，2017 年，第 1009–1024 页。[6] X. Zhang、H. Wu、Z. Chang、S. Jin、J. Tan、F. Li、T. Zhang 和 B. Cui，《Restune：借助元学习增强的面向资源的云数据库调优》，载于*2021* 年数据管理国际会议论文集，2021 年，第 2102–2114 页。

[7] J. Zhang、Y. Liu、K. Zhou、G. Li、Z. Xiao、B. Cheng、J. Xing、Y. Wang、T. Cheng、L. Liu等，“一种使用深度强化学习的端到端自动云数据库调优系统”，载于《2019年数据管理国际会议论文集》，2019年，第415–432页。[8] G. Li、X. Zhou、S. Li和B. Gao，“Qtune：一种基于深度强化学习的查询感知数据库调优系统”，《VLDB基金会论文集》，第12卷，第12期，第2118–2130页，2019年。[9] J. Snoek、H. Larochelle和R. P. Adams，“机器学习算法的实用贝叶斯优化”，《神经信息处理系统进展》，第25卷，2012年。[10] M. A. Wiering和M. Van Otterlo，“强化学习”，《适应、学习与优化》，第12卷，第3期，第729页，2012年。[11] M. I. Jordan和T. M. Mitchell，“机器学习：趋势、视角与展望”，《科学》，第349卷，第6245期，第255–260页，2015年。[12] oltpbenchmark，“OLTPBenchmark.com”，http://oltpbenchmark.com，2012年。[13] sysbench，“SysBench：系统性能基准测试”，http:// sysbench.sourceforge.net，2024年。[14] AuctionMark，“V. Angkanawaraphan和A. Pavlo。AuctionMark：面向高性能OLTP系统的基准测试。”http://hstore.cs.brown.edu/projects/auctionmark，2022年。[15] MySQL，“MySQL v5.7文档”，https://dev.mysql.com/doc/ refman/5.7/en/，2024年。[16] S. M. Lundberg和S.-I. Lee，“解释模型预测的统一方法”，《神经信息处理系统进展》，第30卷，2017年。[17] R. Tibshirani，“通过LASSO进行回归收缩与选择”，《皇家统计学会会刊B辑：统计方法论》，第58卷，第1期，第267–288页，1996年。[18] T. Lillicrap，“基于深度强化学习的连续控制”，arXiv预印本arXiv:1509.02971，2015年。[19] RocksDB，“RocksDB”，https://rocksdb.org/，2012年。[20] X. Zhang、Z. Chang、Y. Li、H. Wu、J. Tan、F. Li和B. Cui，“借助超参数优化促进数据库调优：一项全面的实验评估”，arXiv预印本arXiv:2110.12654，2021年。[21] M. D. Shields和J. Zhang，“拉丁超立方抽样的推广”，《可靠性工程与系统安全》，第148卷，第96–108页，2016年。[22] S. ¨O. Arik和T. Pfister，“TabNet：注意力可解释表格学习”，载于《AAAI人工智能会议论文集》，第35卷，第8期，2021年，第6679–6687页。[23] P. Baldi，“自编码器、无监督学习与深度架构”，载于ICML无监督与迁移学习研讨会论文集。JMLR研讨会与会议论文集，2012年，第37–49页。[24] P.-E. Danielsson，“欧几里得距离映射”，《计算机图形学与图像处理》，第14卷，第3期，第227–248页，1980年。[25] Y. Zhu、J. Liu、M. Guo、Y. Bao、W. Ma、Z. Liu、K. Song和Y. Yang，“BestConfig：通过自动配置调优挖掘系统的性能潜力”，载于《2017年云计算研讨会论文集》，2017年，第338–350页。[26] C. Williams和C. Rasmussen，“用于回归的高斯过程”，《神经信息处理系统进展》，第8卷，1995年。[27] V. Mnih、K. Kavukcuoglu、D. Silver、A. A. Rusu、J. Veness、M. G. Bellemare、A. Graves、M. Riedmiller、A. K. Fidjeland、G. Ostrovski等，“通过深度强化学习实现人类水平的控制”，《自然》，第518卷，第7540期，第529–533页，2015年。[28] V. Konda和J. Tsitsiklis，“演员-评论家算法”，《神经信息处理系统进展》，第12卷，1999年。[29] M. M. Khan和W. Yu，“RoboTune：面向基于集群的数据分析的高维配置调优”，载于《第50届并行处理国际会议论文集》，2021年，第1–10页。[30] B. Cai、Y. Liu、C. Zhang、G. Zhang、K. Zhou、L. Liu、C. Li、B. Cheng、J. Yang和J. Xing，“Hunter：面向个性化需求的在线云数据库混合调优系统”，载于《2022年数据管理国际会议论文集》，2022年，第646–659页。[31] J. Xin、K. Hwang和Z. Yu，“Locat：Spark SQL应用的低开销在线配置自动调优”，载于《2022年数据管理国际会议论文集》，2022年，第674–684页。[32] P. West，“在缺少列出个体的抽样框时对单个项目进行简单随机抽样”，《新西兰林业科学杂志》，第46卷，第1–7页，2016年。[33] D. Micciancio和M. Walter，“整数上的高斯采样：高效、通用、常数时间”，载于《密码学进展——CRYPTO 2017：第37届国际密码学年会，圣巴巴拉，

加利福尼亚州，美国，*2017*年*8*月*20*日至*24*日，论文集，第二部分*37*。施普林格，2017年，第455–485页。[34] B. F. Cooper、A. Silberstein、E. Tam、R. Ramakrishnan和R. Sears，“使用YCSB对云服务系统进行基准测试”，载于第*1*届*ACM*云计算研讨会论文集，2010年，第143–154页。[35] M.-C. Popescu、V. E. Balas、L. Perescu-Popescu和N. Mastorakis，“多层感知机与神经网络”，*WSEAS*电路与系统汇刊，第8卷，第7期，第579–588页，2009年。[36] J. R. Quinlan，“决策树的归纳”，机器学习，第1卷，第81–106页，1986年。[37] L. Breiman，“随机森林”，机器学习，第45卷，32，2001年5页。– 。[38] T. Chen和C. Guestrin，“XGBoost：一种可扩展的树提升系统”，载于第*22*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集，2016年，第785–794页。[39] L. Prokhorenkova、G. Gusev、A. Vorobev、A. V. Dorogush和A. Gulin，“CatBoost：针对类别特征的无偏提升”，神经信息处理系统进展，第31卷，2018年。