迈向一种可扩展且自适应的时间序列异常检测评估指标

Emmanouil Sylligardos *、Mingyi Huang †、John Paparrizos †‡、Pierre Senellart *、Paul Boniol *
*DI ENS、ENS、PSL大学、法国国家科学研究中心、法国国家信息与自动化研究院，法国巴黎
†俄亥俄州立大学，美国俄亥俄州哥伦布 ‡塞萨洛尼基亚里士多德大学，希腊塞萨洛尼基 emmanouil.sylligardos@ens.fr、huang.5749@osu.edu、paparrizos.1@osu.edu、pierre@senellart.com、paul.boniol@inria.fr

arXiv:2611.30385v1 [cs.DB] 23 Nov 2026

摘要——**时间序列异常检测（TSAD）是时间序列分析中的一项基础任务。不同于其他领域的异常检测主要关注点异常，TSAD 还必须处理区间异常。然而，TSAD 方法通常采用继承自分类任务的指标进行评估，例如需要阈值的精确率和召回率，或采用阈值无关的指标（如 AUC），但后者忽略了异常的时间维度。为解决这些问题，曲面下体积（VUS）指标在所有阈值和一系列缓冲区长度下对检测器进行评分，此后已被广泛应用于 TSAD。然而，VUS 存在两个局限。首先，VUS 的计算开销大：在 870 个 TSB-AD 时间序列上评估 32 个检测器需要 7 小时 38 分钟。其次，VUS 对每个异常和检测器使用相同的缓冲区长度，这可能产生过于乐观的得分，从而阻碍对准确检测器和普通检测器的区分。为解决这些问题，我们首先提出 FastVUS，它能够精确计算 VUS，平均速度提升 $76\times$ 倍，具体通过以下方式实现：(i) 以并行方式联合计算所有缓冲区；(ii) 高效剪枝与存在性奖励计算无关的缓冲区；(iii) 将混淆矩阵转换为一次简化的矩阵乘法；(iv) 只需一次遍历即可计算每个缓冲区长度对应的 VUS；(v) 使用全局掩码丢弃非异常区域。我们还开发了 FastVUS-GPU，将每一步重新表示为批量张量运算，在最大误差为 5×10^{-3} 的情况下实现平均 $304\times$ 的加速。基于 FastVUS，我们进一步提出 AutoVUS，这是一种自适应扩展方法，将每次检测与随机基线进行比较，并选择使相对于该基线的增益最大化的缓冲区长度。我们的案例研究表明，AutoVUS 能够减少过于乐观的得分，并更好地区分准确检测器和普通检测器。我们将 FastVUS 和 AutoVUS 作为开源**

推动大规模时间感知评估的源头。

索引词—**时间序列、异常检测、评估指标**

I. 引言

时间序列数据广泛存在于各个领域 [1], [2],，从医疗诊断 [3] 到环境监测 [4]。在所有这些领域中，时间序列异常检测（TSAD）是一项基础任务 [5], [6]，用于标记故障、失效及其他关键事件 [7]–[9]。此外，随着时间序列数据量不断增长，对精准检测器的需求也随之增加，学界对此作出了回应，提出了许多检测方法 [10]–[18]。评估指标负责评估检测器的质量。因此，TSAD 的进展是否可靠，取决于用于选出最佳检测器的指标是否可靠。

然而，流行的评估指标并非为时间序列而设计，而是沿袭自分类领域，包括依赖阈值的指标，如精确率、召回率和F分数，

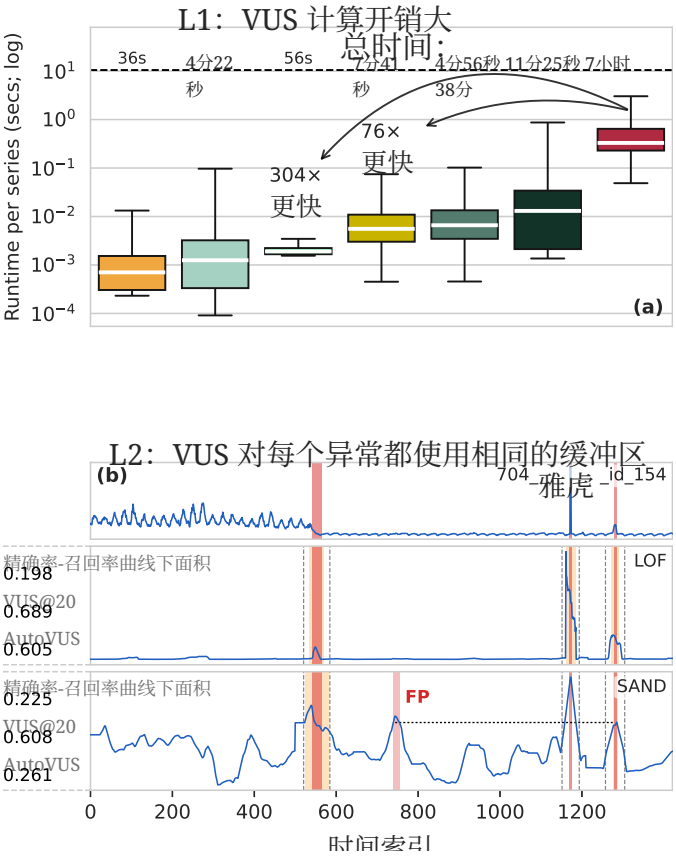


图1：我们的两项主要贡献。(a) TSB-AD 上每条时间序列的运行时间及总运行时间。(b) 一条包含两个得分的时间序列示例。红色标记表示已标注的异常，虚线表示固定的 VUS 缓冲区，橙色表示 **AutoVUS** 选取的缓冲区，浅红色表示假阳性（FP）。VUS@20 是具有 $L = 20$ 的 VUS-PR。

以及 ROC 曲线或精确率-召回率曲线下面积（AUC）等阈值无关指标 [19], [20]。这些指标数十年来一直服务于该领域，但越来越多的研究指出，它们并不适用于时间序列异常检测（TSAD）[14], [21]–[23]。问题在于这些指标对待时间的方式：它们对每个点独立评分，即通过对异常分数设置阈值，将每个点标记为异常或非异常 [14]。如果检测器在标注异常发生前后不久发出警报，所指向的都是同一事件，因此几乎可以算作正确；但按照逐点评分，这仍会被计为漏检。这种僵化会在两种常见于时间序列异常检测的

情形中造成很大代价：(a) 标注者对异常的起止位置意见不一；(b) 许多检测器（例如预测器和自编码器）会在异常完全显现之前或之后稍微提前或延后发出异常信号 [10]。

为解决这些不足，Tatbul 等人 [21] 提出了基于区间的精确率、召回率和 F分数，将传统定义扩展为考量异常是否被检测到、其中有多少个点被检测到，以及检测到的是哪些部分。Range-AUC [14] 和 SoftED [23]进一步引入了一个缓冲区，即围绕每个异常边界的容差区域，使得接近边界的检测结果能够获得奖励，而不是被视为漏检。然而，这些指标需要预先固定参数，例如缓冲区长度，并且检测器分数取决于这一选择 [14]。曲面下体积（VUS）指标 [14]通过对所有阈值以及不超过最大值的所有缓冲区长度进行性能积分，避免固定阈值或缓冲区长度 L，从而得到一个同时对阈值选择和边界不精确性具有鲁棒性的得分。此后，VUS 已成为评估时间序列异常检测的事实上的选择 [24]–[29],，但 VUS 仍存在两个局限性。

局限性1（L1）。VUS 的计算开销大，因为对所有阈值和缓冲区长度进行积分的复杂度为$O(N_\theta N_b n)$ [30], ，其中 N_θ 为阈值数量， N_b为缓冲区长度数量， n 为时间序列长度。VUS 无法扩展到现代时间序列异常检测所需的大规模基准数据集和重复评估。例如，在最近的一篇综述中，由于部分 VUS 结果需要“过长的计算时间”，因而无法报告 [29], ；而在我们的实验中，为 TSB-AD 基准中的 870 个时间序列上的 32 个检测器计算 VUS [13] 需要 7 小时 38 分钟（图 1a）。因此，实践者只能在两种指标之间进行选择：一种能够遵循时间序列的时间结构但无法扩展，例如 VUS；另一种能够扩展但最初是为孤立点设计的，例如 AUC。

局限性2（L2）。VUS 对每个异常和检测器都采用相同的缓冲区长度。对所有不超过 L 的缓冲区长度取平均，使 VUS 不易受任何单一缓冲区长度的影响，但不会根据每次检测所需的容差调整缓冲区。这种统一处理可能过于宽松。例如，短暂的异常可能会分配到远长于事件本身的缓冲区，此时增大缓冲区会抬高得分，但这只是增加了随机水平奖励，并不能说明检测器对事件的定位更加准确。图 1b 展示了一个典型示例：LOF 除了在三个已标注异常附近外始终保持平稳，而 SAND 在整个时间序列中广泛波动，并在第一个和第二个异常之间产生了一个假阳性。AUC-PR 将 SAND 排在首位（0.225 对比 0.198），而 VUS@20 则认为两个检测器的表现相近（0.689 对比 0.608）。因此，VUS 固定的缓冲区方案可能产生过于乐观的得分，使得强弱检测器难以区分。

在本工作中，我们同时解决这两个局限。首先，我们通过**解决 L1**引入**FastVUS**，以极低的成本计算 VUS（平均速度提升 $76\times$ 倍）。我们的贡献与跨时间序列或检测器并行化 VUS 相互正交。相反，我们从头重新设计了其按时间序列进行的计算。FastVUS 通过以下方式重新设计 VUS 的计算：(i) 在一次向量化遍历中注入所有缓冲区；(ii) 利用存在性奖励的单调性，剪枝冗余计算；(iii) 将所有缓冲区和阈值上的混淆矩阵转换为一个单独的归约矩阵乘法步骤；(iv) 在一次遍历中计算每个缓冲区长度对应的 VUS；以及 (v) 使用全局掩码丢弃非异常区域。我们进一步引入**FastVUS-GPU**，将 VUS 的每个步骤重新设计为批量张量运算（平均加速 $304\times$ 倍，最大误差为 5×10^{-3}）。它将顺序存在性计算替换为并行计算，并对较长的时间序列进行分块，从而使评估始终在设备上进行。

其次，我们通过引入**AutoVUS**来**解决 L2**，该方法在 VUS 的基础上，对每个异常和检测器分别应用自适应缓冲。FastVUS 使这一过程成为可能，因为它能够在一次遍历中计算所有缓冲区长度 0、.....、L 对应的 VUS。该选择需要评估每个异常周围不同缓冲区长度下奖励的变化，而原始 VUS 的运行时间无法支持这一点，因为这会使本已昂贵的评估成本再乘以异常数量。实际上，AutoVUS 会为每个异常和检测器选择相对于解析随机基线增益最大的缓冲区，这遵循了与调整聚类指数 [31]相同的机会校正原则。当更大的缓冲正区域主要解释了性能提升时，这一随机归一化规则会限制缓冲区的增长；而当检测器确实在异常附近触发时，则保留其容差。例如，在图 1b 中，AutoVUS 能够清晰地区分局部离群因子和 SAND（0.605 对比 0.261），并为每次检测选择不同的缓冲区，即局部离群因子第一次检测选择 6，第二次检测选择 11。因此，AutoVUS 将 VUS 转变为一种事件级诊断工具，揭示每次检测所需的时间容差。

总体而言，我们的贡献如下：

- 我们提出了**FastVUS**，这是一种对 VUS 的精确重构，重新设计了其整个计算流水线（第*III-A*节）。
- 我们引入了**FastVUS-GPU**，将该流水线重新构建为批量张量运算，同时保持较低的内存复杂度，从而支持设备端评估（第*III-B*节）。
- 我们提出了**AutoVUS**，采用自适应缓冲，即相对于随机基线进行归一化的缓冲区选择，将 VUS 转化为一种事件级度量，揭示每次检测所需的时间容差（第*III-C*节）。
- 我们在 TSB-AD 基准和合成数据上进行了**大规模评估**，结果表明：(i) FastVUS 在中央处理器上平均比原始 VUS 快 $76\times$，同时能够精确复现其结果；(ii) 尽管需要对阈值和缓冲区长度进行积分，FastVUS-GPU 仍实现了平均 $304\times$的加速，运行时间接近 AUC（图 1a），且仅使用了一次近似，最大误差为 5×10^{-3}。我们还通过深度学习流水线实验进一步验证了 FastVUS-GPU 的实际影响（第 *IV* 节）。
- 我们提供**案例研究**和全基准比较，表明 AutoVUS 在保留有意义的时间容差的同时，减少了缓冲区导致的分数膨胀（第五节）

最后，我们总结研究发现（第 *VI* 节）。我们将代码开源 [32]。

II. 背景与相关工作

我们首先介绍全文使用的符号，并回顾 TSAD 评估指标（第 II-A 节）。随后，我们说明如何计算 VUS 及其两项局限（第 II-B 节）。最后，我们回顾 AutoVUS 所依据的随机归一化测度（第 II-C 节）。

时间序列： 时间序列 $T \in \mathbb{R}^n$ 是一个长度为 $n = |T|$ 的序列 $[T_1, T_2, ..., T_n]$。数据集 $\mathcal{D}$ 是由多个时间序列组成的集合，其大小为 $|\mathcal{D}|$。

异常分数序列： 对于一个时间序列 $T \in \mathbb{R}^n$，异常检测方法 D 会返回一个异常分数序列 $D(T) = s$。异常分数与时间序列的长度相同。

异常检测准确率： 对于 $T \in \mathbb{R}^n$，标签 $y \in \{0,1\}^n$ 表示 T 中的点是正常（0）还是异常（1）。我们定义一个准确率函数 Acc： $\mathbb{R}^n \times \{0,1\}^n \to [0,1]$,，其中准确率得分 $Acc(D(T), y)$ 表示 D 的准确程度。

异常区间： 异常区间是一个连续区间$R_k = [t_k^s, t_k^e]$，其中标签为 1， t_k^s 和 t_k^e 分别是其起始索引和结束索引。我们将真实异常区间记为$R = \{R_1, \ldots, R_{N_\alpha}\}$，其中 N_α 表示异常数量。

A. TSAD 评估指标

TSAD 评估指标分为阈值相关和阈值无关两类。阈值相关指标（例如 F分数）评估根据所选阈值生成的二元预测；阈值无关指标（例如 AUC 和 VUS）则汇总所有阈值下的性能。这两类指标又都分为基于点和基于区间两种，也就是说，它们分别评估单个点或异常区间，由此构成下文将要介绍的四个类别。

1）阈值相关：
基于点。基于阈值的评估将异常分数 s 转换为二元预测 $\hat{y}(\theta) \in \{0,1\}^n$，其中当 $s_t \geqslant \theta$ 时，$\hat{y}_t(\theta) = 1$；否则为 0。阈值 θ 通常设为 $\mu(s)+3\sigma(s)$ [7],，其中 $\mu(s)$ 为均值， $\sigma(s)$ 为标准差。将 $\hat{y}(\theta)$与真实标签 $y \in \{0,1\}^n$ 进行比较，可得真正例 TP $= y^\top \cdot \hat{y}(\theta)$、真负例 TN $= (\mathbf{1}-y)^\top \cdot (\mathbf{1}-\hat{y}(\theta))$、假阳性 FP $= (\mathbf{1}-y)^\top \cdot \hat{y}(\theta)$ 和假阴性 FN $= y^\top \cdot (\mathbf{1} - \hat{y}(\theta))$，其中 $\mathbf{1}$ 是长度为 n的全 1 向量。由此可得标准的精确率 Pr $=$ TP/(TP + FP)、召回率 Rc $=$ TP/(TP + FN)、假正例率 FPR $=$ FP/(FP + TN)，以及 F分数（F），即精确率和召回率的调和平均数。

基于区间。 基于区间的精确率和召回率 [21] 将基于点的方法扩展到异常区间，包含四个组成部分：用于表示某个区间是否被检测到的<style id='4'>存在性 奖励（定义见第 II-A2 节）、重叠大小、检测结果在区间内的位置，以及检测结果分散在多个预测中的碎片化程度；每个组成部分均由特定于应用的函数加权。它们的调和平均值即为**基于区间的 F 分数（RF）**，我们在第五节中将其作为基线。尽管这种表述考虑了时间序列结构，但仍需要参数调优，更重要的是，还需要设定一个阈值。

最后，**Affiliation** [22]将每个预测结果归属到距离其最近的真实异常，并根据二者之间的时间距离计算精确率和召回率，因此即使是接近但未完全命中的结果也能获得奖励。它们的调和平均值即为**归属度 F 分数（Aff-F）**，这是我们在第五节中使用的第二个阈值相关基线。

2）阈值无关：
基于点。与上述所有需要阈值的指标不同，阈值无关指标会在所有阈值下评估检测器性能。受试者工作特征曲线下面积（AUC-ROC） [19]衡量真正率（TPR）与假正率（FPR）之间的权衡。真正率定义为 TPR(θ) $=$TP(θ) TP(θ)+FN(θ)，其中 TP(θ) 和 FN(θ) 使用阈值 θ将得分转换为二元预测后计算得到。类似地，~~假正率~~为 FPR(θ) $=$FP(θ) FP(θ)+TN(θ)。给定一个有序的 N_θ阈值集合 $\Theta = [\theta_1, \theta_2, ..., \theta_{N_\theta}]$，其中 $0 = \theta_1 < \theta_2 < ... < \theta_{N_\theta} = 1$，AUC-ROC 使用~~梯形法则~~，在这些阈值上对真正率相对于假正率进行积分。类似地，精确率-召回率曲线下面积（AUC-PR） [20]评估不同阈值下精确率与召回率之间的权衡，并在相同的有序阈值上对 Pr(θ) 相对于 Rc(θ) 进行积分。我们使用 AUC-PR （称为 AUC ）作为基线之一。

基于区间。为了解决这些基于点的度量方法的不足， [14]中的作者定义了**Range-AUC-ROC**和**区间AUC-PR**，将带有标注不精确性容差的子序列级评估纳入其中。他们在异常边界处加入缓冲区域，以便在正常子序列与异常子序列之间创建过渡区，从而奖励在异常边界附近取得较高异常分数的情况。

缓冲区是在每个异常边界前后各添加长度为 b 的容差区域。缓冲区不再将标签视为二值（0 或 1），而是使用连续值扩展标签：这些值从异常边界处的 1，随着距离增加到 b [14]时~~逐渐衰减~~至下限$Z = 1/\sqrt{2}$。这种构造使检测器能够针对接近但未与异常边界完全对齐的预测获得部分得分。形式上，对于跨越索引 $R_k = [t_k^s, t_k^e]$的异常：

$$\tilde{y}_b(t) = \begin{cases} \sqrt{1 - \dfrac{|t_k^s - t|}{2b}}, & \text{if } t_k^s - b \leqslant t < t_k^s \text{ and } \hat{y}_t(\theta) = 1 \\ \sqrt{1 - \dfrac{|t_k^e - t|}{2b}}, & \text{if } t_k^e \leqslant t < t_k^e + b \text{ and } \hat{y}_t(\theta) = 1 \\ y_t, & \text{otherwise} \end{cases} \tag{1}$$

使用连续标签，我们可以计算真阳性$_b = \tilde{y}_b^\top \cdot \hat{y}(\theta)$、假阳性$_b = (\mathbf{1} - \tilde{y}_b)^\top \cdot \hat{y}(\theta)$，并以类似方式计算真负例和假阴性。

上述基于区间的度量中引入的存在性奖励表示是否至少检测到一次异常区间。形式上，对于一个异常区间 R_k 和一组预测范围 P，若存在某个 $P_j \in P$满足 $R_k \cap P_j \neq \emptyset$，则 Existence($R_k$, P) $= 1$，否则为 0。基于范围的真正率将其纳入如下：

$$TPR_b = \frac{TP_b}{P_b} \cdot \frac{\sum_{R_k \in R} \text{Existence}(R_k, P)}{N_\alpha} \tag{2}$$

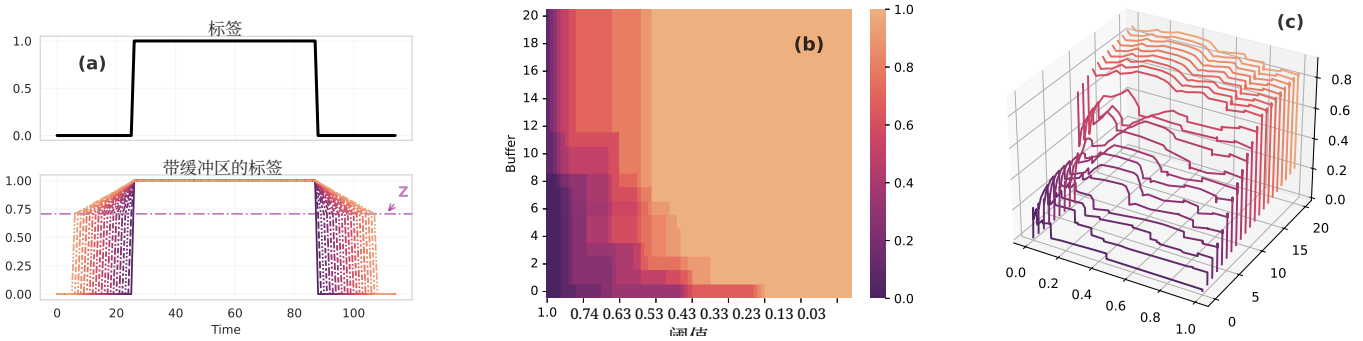


图 2：(a) 添加 20 个缓冲区前（上）和后（下）的标签（缓冲区高度 Z），(b) 所有缓冲区 $\times$阈值对的存在性值，以及 (c) 由 20 个缓冲区得到的精确率-召回率曲面（为清晰起见，以线条表示）。

R 和 P 分别是真实标注和预测异常区间的集合，P_b 是正例数量，在 [14]中定义为原始正标签质量与缓冲后正标签质量的均值。区间AUC-ROC 和区间AUC-PR 遵循标准 AUC 公式，以缓冲后的对应指标替代 TPR 和 FPR（或精确率和召回率）。我们使用区间AUC-PR（称为**Range-AUC**）作为基线之一。尽管这两种度量都具有阈值无关性且考虑了时间信息，但它们增加了一个需要调整的参数：缓冲区长度 b。

B. VUS：计算概述

为避免选择单一缓冲区长度，VUS通过在一个网格中对每个缓冲区长度分别计算ROC曲线或PR曲线来扩展Range-AUC。因此，二维Range-AUC曲线（精确率对召回率，或TPR对FPR）变成了三维空间中的曲面，其中缓冲区长度为第三个轴。由此得到的度量称为曲面下体积（VUS），分别对应ROC曲面（VUS-ROC）或PR曲面（VUS-PR）。在我们的研究中，我们使用VUS-PR，为简洁起见将其称为**VUS**。

为便于直观理解，我们以图2为指导，逐步介绍其计算过程（伪代码见算法 1、第III-A节）。首先，如图2a所示，通过在每个异常边界前后添加缓冲区来扩展原始二元标签（算法 1，第12行）。需要特别指出的是，添加缓冲区时，会忽略那些可能引入额外假负例的区域。其次，对于缓冲区长度和检测阈值的每种组合，我们计算存在性奖励，并将其以热图形式展示在图2b中（第13行）。第三，对于每个缓冲区 $\times$ 阈值对，我们计算混淆矩阵（TP、FP、TN、FN）（第15行）。第四，根据这些混淆矩阵，我们计算所有组合下的精确率和召回率，并将存在性奖励纳入召回率，以确保检测器在检测出更大比例的异常时能够获得相应的认可（反之亦然）。这些值在缓冲区和阈值两个维度上构成一个三维曲面，并在图2c中以多条独立曲线表示（第19行）。第五，计算该精确率-召回率曲面下面积，并按缓冲区数量进行归一化，从而得到单一的VUS 分数（第21行和第23行）。VUS-ROC采用相同的过程，只是用假正率替代精确率。

尽管VUS日益被用于时间序列异常检测评估，但它仍存在两个局限：

（L1）时间感知与可扩展性。通过同时对阈值和缓冲区长度进行积分，VUS引入了计算瓶颈。其时间复杂度为 $O(N_\theta N_b n)$ [30], 其中，N_θ 是阈值数量，$N_b = |\mathcal{L}| = L + 1$ 是网格中的缓冲区长度数量 $\mathcal{L} = \{0, \ldots, L\}$，最大值为$L$，而 n 是时间序列的长度。对于包含一百万个点的时间序列，VUS耗时约10秒，而AUC耗时0.1秒（第IV-B节）；在TSB-AD 基准的870个时间序列上评估32个检测器 [13]则需要7小时38分钟（第IV-A节）。因此，对于需要在大量时间序列上评估许多检测器的研究（例如 [33]），VUS并不实用。当按每个异常检查时间容差时，问题会进一步加剧，因为同一时间序列中的异常和检测结果可能需要不同的缓冲区，此时必须以更细的粒度重复计算。

(L2) 固定缓冲区限制。一旦设定缓冲区长度 L，VUS 就会对每个异常和检测器使用相同的缓冲区网格 $\mathcal{L}$。针对每个时间序列或每个异常调整该参数并不实际。短时异常可能不需要与长时异常相同的容差；而精确定位的检测结果，不应与只有在较大缓冲区下才与异常对齐的宽泛检测结果一视同仁（第 V-A 节）。此外，扩大缓冲区会增大软化正区域，因此得分的提升可能部分反映了更高的随机水平奖励，而非更好的时间定位。例如，在 TSB-AD 上，均匀随机分数的 VUS 平均值在 $L = 0$ 时为 0.03，在 $L = 128$ 时为 0.16；而 AutoVUS 在 $L = 128$ 时给出的分数为 0.04。这种膨胀会让检测器性能更难区分。这些局限表明，应针对每个异常—检测器组合分别选择缓冲区；但要使局部搜索切实可行，就需要高效的计算方法。要找出观测到的增益超过随机检测器预期增益的缓冲区，还需要一个机会校正准则。

C. 随机归一化测度

如上所述，扩大缓冲区会增大软化正区域，因此 VUS 分数可能因随机水平奖励提高而上升，而非因为局部定位效果变好。这一问题在聚类领域已有充分研究，即使

随机划分也可能偶然匹配部分真实标签，从而使兰德指数 [34] 和互信息 [35]等指标得到非零得分。调整指标会减去随机模型下的期望得分：

$$\mathrm{Adj}(V) = \frac{V - \mathbb{E}[V_{\mathrm{rand}}]}{V_{\max} - \mathbb{E}[V_{\mathrm{rand}}]},$$

其中， V 是该度量的值， V_{rand} 是其在随机模型下的值， $V_{\max}$ 是其在完全一致情况下的值。调整兰德指数（ARI）将这一校正应用于基于配对计数的一致性 [36]，，调整互信息（AMI）则将其应用于信息论一致性 [35]。此类针对偶然因素进行调整的聚类比较指标已得到系统研究 [31]。我们在第三节 C-3 中采用相同原理，将缓冲区带来的增益相对于随机检测器预期的得分进行归一化。

III. Proposed Approach:一种快速且自适应的 *VUS*

本节介绍我们的两项贡献，分别针对 VUS 的一个局限，即其计算成本（L1）和固定缓冲（L2）。我们的第一项贡献是 **FastVUS**，这是一种计算现有 VUS 度量（第二节 B）的算法，在消除计算瓶颈的同时，返回完全相同的得分，**从而解决 L1**。其图形处理器变体 **FastVUS-GPU** 在设备上以完全并行的方式执行相同计算，仅在存在性项中采用了一处近似。我们的第二项贡献是**AutoVUS**，这是 VUS 的一种自适应扩展，并由 FastVUS 使其具备实用性。AutoVUS 不再为整个时间序列采用单一缓冲区长度，而是为每个异常和检测器分别选择缓冲区长度，使 VUS 成为一种可解释的事件级诊断，**从而解决 L2**。

A. FastVUS：*VUS* 的精确重构

FastVUS 重新设计了 VUS 的五个步骤（第 II-B 节），并在此基础上增加了一项优化，如算法 1 所示。每一步都是精确的。

1) 缓冲区注入：单次遍历中注入所有长度不超过最大值的缓冲区。**2) 存在性计算**：计算检测到的异常所占比例，同时跳过冗余的缓冲区 ×阈值评估。**3) 混淆矩阵**：通过一种构造，在一步中得出所有缓冲区 ×阈值对对应的 TP、FP、FN、TN。**4) 精确率和召回率**：所有缓冲区 ×阈值对上的精确率-召回率曲面。**5) VUS 积分**：将该曲面积分为 VUS，且无需额外成本，即可得到完整的 VUS（b）曲线。

这项优化（第 III-A6 节）是**全局掩码**，它会丢弃非异常区域。

1) 缓冲区注入：缓冲区是 VUS 的核心。VUS 会在每个异常周围添加一个软容差区域，以便
靠近边界的检测结果会获得奖励。原始实现一次只为一个异常插入这些缓冲区，并且每经过一个缓冲区长度就重写一次标签，这会成为瓶颈。相反，我们通过一个由 FastVUS 和 FastVUS-GPU 共享的闭式函数，一次性计算所有缓冲区。

我们首先根据二元标签计算一个位置 序列（二元标签位于图3顶部）。对于异常之外的每个点，位置表示该点到最近异常边界的距离；各点

算法 1 FastVUS 主计算循环

要求：标签 $y \in \{0,1\}^n$，异常分数 $s \in \mathbb{R}^n$1： $R \leftarrow$ 获取异常区间 (y) $\Theta \leftarrow$去重(s) 2：# 可选（参见第III-A6节） 3：如果全局掩码处于启用状态，则 4：# 在丢弃非缓冲区区域前，计数其中的假阳性 5：$M_{\text{safe}} \leftarrow$ 创建安全掩码 (y, R) 6：假阳性 $_{\text{ext}} \leftarrow$ 计数超阈值项 ($s[\neg M_{\text{safe}}], \Theta$) 7：# 过滤输入，仅保留缓冲区区域
8 $y \leftarrow y[M_{\text{safe}}]$ $s \leftarrow s[M_{\text{safe}}]$：，：# 在缩减后的向量上重新
9 计算区间 10：$R \leftarrow$获取异常区间(y) 11：结束如果 12： $Y \leftarrow$ 添加缓冲区(y, R) 13： $E \leftarrow$ 计算存在性(Y, s, Θ) 14： $S \leftarrow$($s \geqslant \Theta$) 15：真阳性、假阳性、假阴性 $\leftarrow$混淆矩阵(Y, S) 16：如果全局掩码处于启用状态，则 17：假阳性 $\leftarrow$ 假阳性 +假阳性$_{\text{ext}}$ {将经过滤区域中的假阳性加回}18：结束如果 19：精确率、召回率 $\leftarrow$精确率-召回率曲线(真阳性，假阳性，假阴性 E) 20：# 逐缓冲区曲线下面积：每个缓冲区对应一个面积 $b \in \mathcal{L}$21： $A \leftarrow$ AUC(召回率，精确率) 22：# 累积均值：VUS(b)是截至缓冲区 b的面积均值 23：VUS $\leftarrow$累积均值(A) 24：# 完整曲线；VUS(L)是标准标量 25：返回VUS

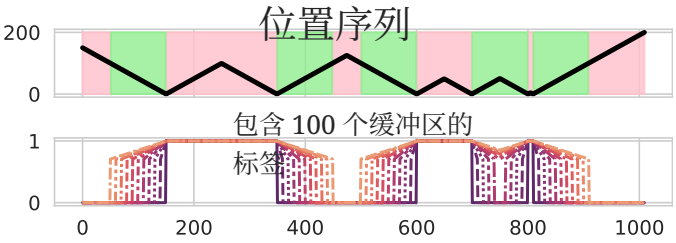


图 3：所提出的向量化缓冲区注入。上图：位置序列（每个点到最近异常边界的距离），其中缓冲区域为绿色，未受影响的区域为红色。下图：生成的缓冲标签。

异常内部的位置为 0（因此其标签保持为 1）。它在所有时间点上并行计算。给定缓冲区长度 b，缓冲标签如下

$$\tilde{y}_b(t) = \begin{cases} \sqrt{1 - (1 - Z^2)\,\dfrac{\mathrm{pos}(t)}{b}}, & \text{if } \mathrm{pos}(t) \leqslant b, \\ y_t, & \text{otherwise,} \end{cases}$$

其中， Z 控制缓冲区奖励的强度。奖励在边界处从 1 开始，并随着距离增加而衰减，在距离为 b 时降至 Z （公式1所示的曲线），而距离任何异常超过 b 的点则保留原始标签。两个缓冲区重叠时，点取两者中的较大值。为了同时获得所有缓冲区，我们对所有长度 $b \in \mathcal{L}$计算此函数，从而在一次向量化遍历中生成完整的缓冲标签矩阵 Y （图3底部）。

2) 存在性计算：存在性奖励是至少被得分中的一个点检测到的异常所占的比例，也是 VUS 的主要瓶颈。标准定义会针对每个缓冲区和阈值遍历每个异常，代价为 $O(N_\theta N_b N_\alpha)$。然而，我们观察到，存在性在这两个维度上都是单调的。在给定阈值下的检测结果会在所有更低阈值下持续存在，而在给定缓冲区长度下的检测结果会在所有更大缓冲区长度下持续存在。一种检测

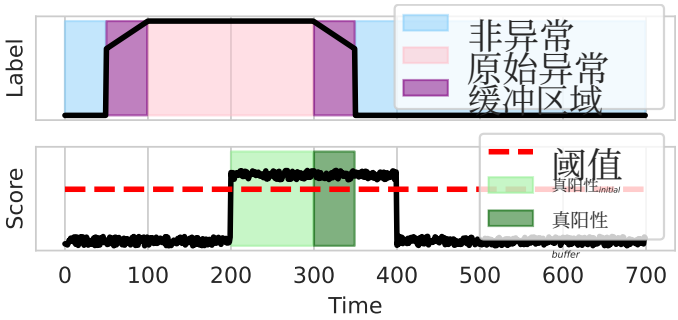


图 4：混淆矩阵的计算。上：标签被划分为三个区域。下：用于计算真阳性的区域。

在给定阈值下的检测结果会在所有更低阈值下持续存在，而在给定缓冲区长度下的检测结果会在所有更大缓冲区下持续存在。我们如下利用这一点：

阈值传播。对于每个异常，我们取其范围内的最大得分，将其作为检测阈值，并一次性将该异常标记为在此阈值及所有更低阈值下均已检测到。这样便无需进行阈值循环，将成本从$O(N_\theta N_b N_\alpha)$降至 $O(N_b N_\alpha)$。

缓冲区跳过。我们确定最大得分落在缓冲区域中的位置。如果它落在原始异常内部（图 4 中的粉色区域），那么无论缓冲区长度为何，该异常都会被检测到，因此我们一次性填充所有缓冲区，然后处理下一个异常。如果它落在缓冲区内部（图 4 中的紫色区域），我们就计算从哪个缓冲区索引开始检测到该异常，将所有更大的缓冲区标记为已检测到，然后对较小的缓冲区递归处理。按从大到小的顺序处理缓冲区时，每一步要么终止处理，要么缩小剩余范围，因此大多数缓冲区评估都无需执行。扩大的缓冲区可能导致相邻异常合并。我们会跟踪这些合并，并按缓冲区调整异常数量，以确保各比例保持精确。在 TSB-AD 上，该过程平均跳过了$N_b N_\alpha$ 次评估中的 93.9%，同时精确计算存在性奖励。

3）混淆矩阵：对于每一对缓冲区和阈值，VUS 都会构建一个混淆矩阵（真阳性、假阳性、假阴性、真负例）。原始 VUS 通过对 N_b 缓冲区和N_θ 阈值进行双重循环来填充该矩阵。我们的关键贡献是将整个计算重新表述为一次矩阵乘法：真阳性 $= YS^\top$由缓冲标签矩阵 Y 与阈值化分数矩阵 S 相乘得到，以一次计算取代 $N_b N_\theta$ 次独立计算。随后通过逐元素运算得出假阳性、假阴性和真负例。据我们所知，这是首次以这种方式表示 VUS；这种重新表述使计算能够利用中央处理器和图形处理器上的优化稠密线性代数运算。

我们进一步利用缓冲标签的结构，使这一乘法计算更加高效。每个标签分为三个区域（图 4）：(1) 非异常区域（$\tilde{y}_b(t) = 0$）、(2) 原始异常（$\tilde{y}_b(t) = 1$）以及 (3) 缓冲区（$0 < \tilde{y}_b(t) < 1$）。非异常区域对 真阳性没有贡献，因此可以排除；这是 VUS [30] 期刊版本中引入的一项优化。反过来，原始异常会为每个缓冲区产生 $\tilde{y}_b(t) = 1$，因此其贡献在不同缓冲区之间保持不变，并可归结为一个点积。综合这些观察结果可得到 真阳性 的最终形式：原始异常上的常数项 真阳性$_{\text{initial}}$，加上一个限制在总长度为 $2LN_\alpha$ 的缓冲区区域上的矩阵乘法真阳性$_{\text{buffer}}$：

$$\begin{aligned} TP &= TP_{\text{initial}} + TP_{\text{buffer}} \\ &= y[M_a] \cdot S[:, M_a]^\top + Y[:, M_b] \cdot S[:, M_b]^\top, \end{aligned}$$

其中，y 是原始二元标签，M_a 是异常掩码，M_b 是缓冲区掩码。总成本为 O（$N_b L N_\theta N_\alpha$）$+O$（n（$N_b + N_\theta$））。由于异常通常只占时间序列的一小部分（$2LN_\alpha \ll n$），因此该成本远低于原始 VUS 的O（$N_\theta N_b n$）。人们可能会认为，通用稀疏矩阵库可以替代这些计算，因为两个矩阵都是稀疏的。然而，与中央处理器（scipy.sparse）和图形处理器（cuSPARSE）上的稀疏矩阵库相比，我们针对时间序列的专用计算速度约快 9× 倍（在 TSB-AD 上按序列计算的中位数比值）。无论如何，混淆矩阵只是 VUS 的多个步骤之一，因此仅替换这一环节并不能消除瓶颈，第 IV-A 节中的消融实验对此进行了说明。

4) 精确率和召回率：有了混淆矩阵，便可根据真阳性 和假阳性计算精确率，根据真阳性 和假阴性计算召回率，并按存在性奖励进行缩放。在 N_b 个缓冲区和 N_θ 个阈值范围内，它们共同构成精确率-召回率曲面。

5) VUS 积分：VUS-PR 定义为该曲面下方的体积。对于每个缓冲区，我们计算其精确率-召回率曲线下的面积，再对所有缓冲区的面积取平均，得到最终得分。其 ROC 变体 VUS-ROC 只需在最后一步做一处改动：用假正例率替代精确率。此前的步骤均保持不变，FastVUS 可实现这两种方法。

这种对缓冲区取平均的做法掩盖了一种可供我们利用的结构。我们保留每个缓冲区对应的面积，并将面积的累积和除以截至目前的缓冲区数量，从而在单次遍历中，为从 0 到 L 的每个缓冲区长度 b 生成曲线$VUS(b)$。这是可行的，因为缓冲区长度具有嵌套关系：较大的缓冲区包含所有较小的缓冲区。因此，在最大长度 L 下计算 VUS 时，也会计算每个较短缓冲区对应的精确率-召回率曲线，而 VUS 的原始实现会将这些结果丢弃 [14]。标准的 VUS 标量值可由最后一个元素$VUS(L)$ 得到。由此，VUS 成为缓冲区长度的函数，而不再是单一数值。正是这条曲线使得 III-C 节中针对每个异常选择缓冲区成为可能。若没有这条曲线，要获得每个候选缓冲区长度下的 VUS，就必须对每个长度重新计算该度量。

6) 全局掩码：异常很少，因此大多数时间序列只产生真负例和假阳性，我们利用这一点引入全局掩码。我们从标签中移除每个非异常区域，仅保留异常及其缓冲区，并在这一更短的序列上运行完整流程。由于被移除的区域只能产生假阳性，因此准确性得以保持。我们在应用掩码前对这些假阳性进行一次计数，并在最后将其加回。在 TSB-AD 中，保留区域（原始异常及其缓冲区）平均占时间序列的 24%，因此全局掩码移除了剩余的 76%（其对加速效果的贡献在第 IV-A 节的消融实验中进行了测量）。

B. FastVUS-GPU：一种完全并行的实现形式

现在，我们介绍 FastVUS-GPU，这是 FastVUS 的完全向量化改写版本：整个计算过程在图形处理器上以批量张量运算的形式执行，使评估过程始终在设备端进行。其中有两个部分专门针对图形处理器进行了设计，即存在性计算和内存分块步骤。其余步骤与 FastVUS（第 III-A 节）完全相同，并保持精确性。

1) 向量化存在性：图形处理器一次对整个批次执行单个操作，而不是按顺序逐步执行。然而，FastVUS 的存在性计算通过顺序处理并根据中间结果跳过缓冲区来减少计算量，而并行执行无法表达这种方式。因此，我们将存在性计算重新表述为完全并行的计算，以充分发挥图形处理器的优势。

我们首先为缓冲标签中的每个异常片段分配一个唯一且递增的索引，从而构造出类似阶梯的序列。实际操作中，这相当于复制标签，并为每个异常标注其自身的索引（1，2，3，.....），而不是 1。正常区域则保持为 0。我们通过累积和为每个缓冲区构造一个这样的序列，得到一个 $N_b \times n$ 张量。借助这个阶梯张量，我们通过一次分组归约提取每个异常的最大得分，因为每个点都标记了其所属的片段。这样便得到一个 $N_b \times N_\alpha$张量，其中为每个异常和缓冲区分别保存一个片段最大值。当且仅当某个异常的最大值超过给定阈值时，该异常才会被检测到；因此，存在性奖励是最大值超过阈值的片段所占的比例，构成一个 $N_b \times N_\theta$ 张量。这与中央处理器版本（第III-A2节）中的检测阈值判定标准相同，只是现在以并行方式计算。

除了图形处理器上的单精度舍入外，向量化存在性计算是 FastVUS-GPU 中唯一的近似来源。当两个经过缓冲的异常片段互不重叠但彼此相邻时，它们会获得相同的索引并合并为一个，而顺序版本则会将它们保持为独立片段。在 TSB-AD 上的 5×10^{-3} 和 $L = 128$ （第IV-A节）条件下，实证表明误差始终低于该值。

2) 内存分块：并行处理每个缓冲区和阈值会具体化大型张量，即 $N_b \times n$ 缓冲标签和 $N_\theta \times n$ 得分掩码；对于较长的时间序列，它们可能超出可用显存。为解决这一问题，我们将时间序列划分为多个分块，仅在非异常间隙内进行切分，从而确保每个异常及其缓冲区都位于同一个分块中。每个分块都会统计其混淆矩阵计数、检测到的异常数量和异常总数量，并在最后将这些结果合并。由于切分位置均位于每个缓冲区之外，合并后的结果与一次性处理整个时间序列完全相同；也就是说，分块影响的是内存，而不是得分。

C. AutoVUS：一种自适应、可解释的度量方法

FastVUS 和 FastVUS-GPU 消除了 VUS 的计算瓶颈，也使我们能够解决 VUS 的第二个局限。StandardVUS 对所有异常和检测器都在相同的缓冲区网格上取性能平均值，尽管不同事件所需的时间容差可能有所不同。精确定位的检测可能需要很小的容差，甚至不需要容差；而较早或延迟的响应，则可能只有在允许更宽的缓冲区后才能与异常对齐。要针对每个异常分别确定这一容差，就需要在事件级别评估完整的、针对不同缓冲区的性能曲线；使用原始 VUS，这样做的成本将高得难以承受。FastVUS 只需单次遍历即可返回完整曲线，使这项分析切实可行。在此基础上，AutoVUS 依据相对于随机基线归一化后的增益，为每个异常选择特定的缓冲区，并报告一个经过机会校正的得分，该得分在由此产生的不同缓冲区配置间仍具有可解释性。

1) 问题设置：AutoVUS 的输入包括一个二元标签 $y \in \{0,1\}^n$、一个检测器分数 $s \in \mathbb{R}^n$以及一个最大缓冲区长度 L。设 y中的异常区间为 R_1、.....、R_{N_α}，并设 $\mathcal{L} = \{0, \ldots, L\}$ 为候选缓冲区网格。标准 VUS 会在这个公共网格上评估每个异常，并对所得性能取平均值。与之不同，AutoVUS 会选择一个逐异常缓冲向量 $b^* = (b_1^*$、.....、$b_{N_\alpha}^*)$，其中 $b_k^* \in \mathcal{L}$是针对异常 R_k，根据分数 s选出的最大时间容差；也就是说，它会为每个异常与检测器的组合选择一个缓冲区。因此，与 VUS 不同，在 VUS 中，最大缓冲区长度 L 是给予每个异常的容差；而在 AutoVUS 中，L仅仅是所授予容差的上限。

选定的缓冲区还提供了检测器行为的事件级诊断。较小的 b_k^* 表示检测器能够精确定位异常 R_k，而较大的值则表示只有在允许更大的时间容差后，检测器响应才与该事件相吻合。

2) 随机归一化缓冲区选择：对于每个异常R_k，我们通过平分R_k 与其相邻异常之间的间隔，定义一个局部窗口 W_k。这将隔离出可将检测器响应归因于 R_k 的区域。在 W_k 内，令 $C_k(b; s) = (b+1)^{-1} \sum_{b'=0}^{b} A_k(b'; s)$ 表示累积局部 VUS 曲线，其中 $A_k(b; s)$ 是在缓冲区长度为 b 时得到的边际 AUC。由于 FastVUS 返回累积曲线，因此可通过去累积精确恢复边际值：$A_k(0;\ s) = C_k(0;\ s)$ 和 $A_k(b;\ s) = (b+1)C_k(b;\ s) - bC_k(b-1; s)$，其中 $b \geq 1$。

令 $\tilde{y}_{k,b} \in [0,1]^{|W_k|}$ 表示在 W_k 内对异常 R_k 应用缓冲区长度 b 后得到的局部缓冲标签。在检测器分数与标签相互独立的随机排序模型下，预期 AUC 等于相应缓冲标签的平均正类质量。因此，我们将解析随机基线计算为

$$g_k(b) = \mathbb{E}[A_k(b; s_{\text{rand}}) \mid \tilde{y}_{k,b}] = \frac{1}{|W_k|} \sum_{t \in W_k} \tilde{y}_{k,b}(t),$$

其中，s_{rand} 表示一个与标签相互独立的随机检测器分数。该基线直接根据缓冲标签计算，无需进行蒙特卡罗采样。由于增大缓冲区会扩大软化正区域，$g_k(b)$通常会随 b 增大。因此，直接最大化 $A_k(b; s)$ 会倾向于选择更长的缓冲区，即使表面上的提升仅仅源于随机水平奖励的增加。

为分离可归因于检测器响应的增益，我们将每个缓冲区与无缓冲区情形进行比较。令 $\Delta A_k(b; s) =$

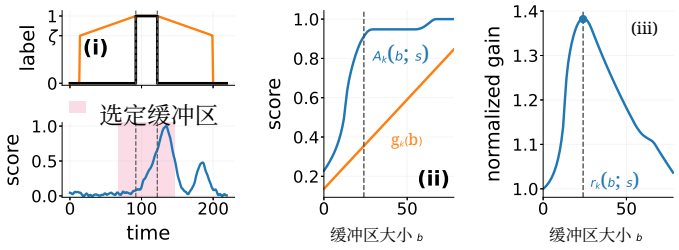


图5：自适应缓冲区选择。(i) 标签 y、得分 s，以及为异常 R_k 选定的缓冲区。(ii) 边际 AUC $A_k(b;s)$ 和随机基线 $g_k(b)$ 随缓冲区长度 b 的变化。(iii) 随机归一化增益 $r_k(b;s)$，其最小最大值对应 b_k^*。

$A_k(b;s) - A_k(0;s)$ 表示检测器的缓冲区诱导增益，令 $\Delta g_k(b) = g_k(b) - g_k(0)$ 表示相应的随机增益。我们选择使随机归一化增益最大的最小缓冲区：

$$b_k^* = \min \arg\max_{b\in\mathcal{L}} r_k(b;s), \quad r_k(b;s) = \frac{\Delta A_k(b;s)+1}{\Delta g_k(b)+1}.$$

加性常数使目标函数在 $b = 0$ 处定义良好，此处 $\Delta A_k(0;s) = \Delta g_k(0) = 0$，因此 $r_k(0;s) = 1$。选择最小的最大值可打破平局，并优先选取检测器所支持的最小时间容差。具体而言，如果没有正缓冲区能产生高于随机基线的增益，那么对于所有 b，$r_k(b;s) \le 1 = r_k(0;s)$，而选定的缓冲区为 $b_k^* = 0$。

图5展示了选择过程。随着缓冲区长度增加，$A_k(b;s)$ 会增大，因为更多检测结果可能落在缓冲标签内；但随着软化正区域扩大，$g_k(b)$ 也会增大。因此，AutoVUS 选择随机归一化增益曲线的峰值，而不是最大的原始边际 AUC。

由于 $r_k(b;s) - 1 = \frac{\Delta A_k(b;s) - \Delta g_k(b)}{\Delta g_k(b)+1}$，因此，数值 1 是缓冲区诱导增益的机会水平参考值；$r_k(b;s) > 1$ 当且仅当检测器获得的缓冲区诱导增益高于随机基线的预期值时成立。对于独立于标签的随机检测器，$\mathbb{E}[A_k(b;s_{\text{rand}})] = g_k(b)$，因此 $\mathbb{E}[\Delta A_k(b;s_{\text{rand}})] = \Delta g_k(b)$。由于 $\Delta g_k(b)$ 由标签和缓冲区构造固定，$\mathbb{E}[r_k(b;s_{\text{rand}})] = 1$ 对于每个固定的 b 都成立。因此，零假设曲线在期望意义下是平坦的，较长的缓冲区不会仅仅因为增加了缓冲标签的正类质量而获得系统性优势。

3) 机会校正VUS：选择步骤生成逐异常缓冲向量 $b^* = (b_1^*, \dots, b_{N_a}^*)$。给定 b^* 后，我们对每个 $b \in \mathcal{L}$ 执行 VUS缓冲区扫描，但将每个异常的扩展 R_k 限制在其选定的缓冲区内，因此在扫描级别 b，异常 R_k 使用的缓冲区长度为 $c_k(b) = \min\{b, b_k^*\}$。令 $\tilde{y}_{b^*}^{(b)} \in [0,1]^n$ 表示扫描级别 b 下得到的全局缓冲标签，令 $A_{b^*}(b;s)$ 表示根据该标签和检测器分数 s 计算得到的 AUC。于是，调整后的 VUS 分数为：

$$\text{VUS}_{b^*}(s) = \frac{1}{N_b}\sum_{b\in\mathcal{L}} A_{b^*}(b;s).$$

不同的选定缓冲区向量会产生不同的全局缓冲标签序列，因此也会产生不同的机会基线。在随机排序模型下，扫描级别 b 对应的期望 AUC 等于相应缓冲标签的平均正类质量，即 $g_{b^*}(b) = \frac{1}{n}\sum_{t=1}^{n} \tilde{y}_{b^*}^{(b)}(t)$。完整配置的解析随机基线是整个扫描的均值，即 $\bar{g}_{b^*} = \frac{1}{N_b}\sum_{b\in\mathcal{L}} g_{b^*}(b)$。我们报告机会校正AutoVUS分数：

$$\text{AutoVUS}_{b^*}(s) = \frac{\text{VUS}_{b^*}(s) - \bar{g}_{b^*}}{1 - \bar{g}_{b^*}}.$$

这遵循机会校正形式，即观测性能减去预期随机性能，再除以完美性能减去预期随机性能。两项均使用相同的异常特定缓冲区上限。

4）校准与可比性：我们首先针对任意固定的逐异常缓冲向量 b^* 建立校准。由于 $A_{b^*}(b;s) \le 1$ 在每个扫描级别上都成立，因此 $\text{VUS}_{b^*}(s) \le 1$，进而 $\text{AutoVUS}_{b^*}(s) \le 1$。

对于机会端点，假设 s_{rand} 是一个独立于标签的随机检测器。在每个固定的扫描级别上，$\mathbb{E}[A_{b^*}(b;s_{\text{rand}})] = g_{b^*}(b)$，因此，对于任意固定的选定缓冲区向量 $\mathbb{E}[\text{VUS}_{b^*}(s_{\text{rand}}) \mid b^*] = \bar{g}_{b^*}$，进而 $\mathbb{E}[\text{AutoVUS}_{b^*}(s_{\text{rand}}) \mid b^*] = 0$。因此，对于每种固定配置，随机性能的期望值都映射为 0。

对于完美端点，考虑一个完美定位检测器，其所有异常点的排名都高于非异常点，因此对于每个异常 R_k，$A_k(0;s) = 1$。对于任意 $b > 0$，$A_k(b;s) \le 1$，而 $g_k(b) \ge g_k(0)$ 给出$r_k(b;s) \le 1 = r_k(0;s)$，因此对于每个异常，最小最大化者为 $b_k^* = 0$。于是，$c_k(b) = 0$ 在每个扫描级别上均成立，每个全局缓冲标签 $\tilde{y}_{b^*}^{(b)}$ 都等于原始二元标签，并且该检测器在整个过程中都获得 $A_{b^*}(b;s) = 1$。因此，$\text{VUS}_{b^*}(s) = 1$，$\text{AutoVUS}_{b^*}(s) = 1$。

随机基线校正还使选择不同缓冲区向量的检测器之间具备可比性。假设两个检测器 s_1 和 s_2 分别选择 b_1^* 和 b_2^*。它们的原始自适应曲线下面积体积值针对每个异常使用不同的缓冲区进行评估，因此其随机基线可能不同。AutoVUS 会移除每个所选配置的预期随机性能，并以其与完美性能之间的剩余差距进行归一化。因此，在给定每个所选缓冲区配置的条件下，0 表示预期的随机水平性能，1 表示完美性能。

上述校准以选定的缓冲区向量固定为条件。由于 b^* 是根据观测到的检测器分数选出的，最大化步骤可能会捕捉到随机检测器的正向有限样本波动。因此，完整自适应过程的无条件期望不一定恰好为零（TSB-AD 上为 0。04，见 II-B 节）。

IV. FASTVUS 实验评估

我们从三个维度评估 FastVUS：它与原始 VUS 的等价性（第IV-A节）、其运行时间（第IV-A节和第IV-B节），以及它在真实评估工作流程中的表现（第IV-C节）。我们的实验使用 TSB-AD 基准 [13]（870 条单变量序列和 32 个检测器），但深度学习流水线（第IV-C节）除外，该流水线使用其前身 TSB-UAD [11]。为了研究可扩展性，在

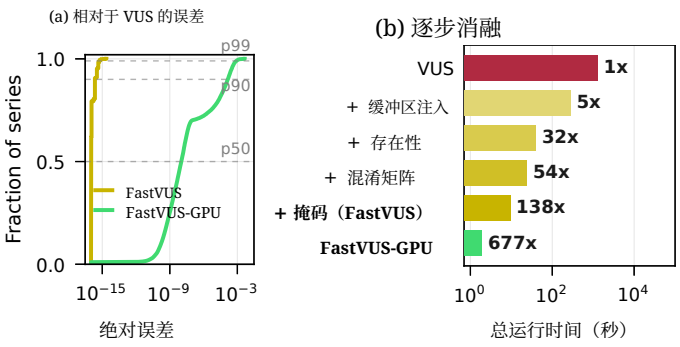


图6：TSB-AD 上的准确性和消融实验。(a) 相对于原始 VUS 的绝对误差，FastVUS 至多为 2×10^{-15}，FastVUS-GPU 至多为 5×10^{-3}。(b) 依次在前述技术的基础上添加每种技术，速度提升以相对于原始 VUS 的总耗时比表示。

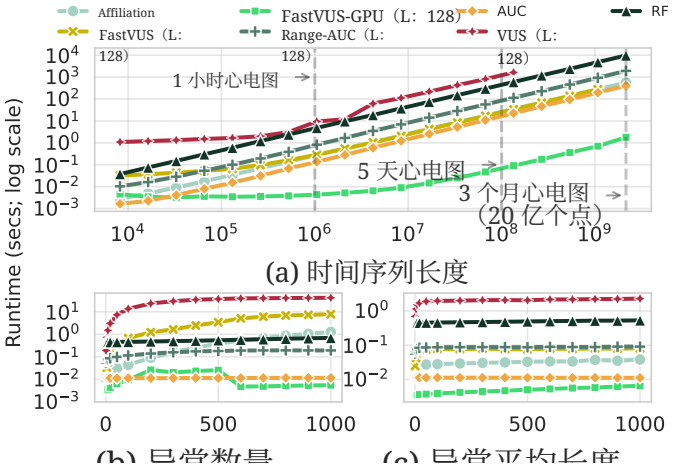


图 7：合成时间序列上的运行时间与关键特征的关系（在 (a) 中，FastVUS 和 Affiliation 重叠显示；ECG 表示心电图）。

受控条件下，我们还生成合成序列，并改变三个特征：(a) 序列长度，(b) 异常数量，以及 (c) 平均异常长度。每次只改变其中一个特征，同时保持另外两个特征固定，并且每种配置都在 10 个随机数据集上取平均。

我们将 FastVUS 与原始 VUS [14],的优化实现 [30],进行比较，并与另外四种 TSAD 评估指标进行比较：AUC、Range-AUC、 [14],Range F-分数 [21], 和 Affiliation [22]。在本节中，我们使用长度为 $L = 128$ 的缓冲区，这是一种计算上具有挑战性但符合实际的设置。在第 V-A 至 V-C 节的案例研究中，我们考察 VUS 得分本身，使用 $L = 20$；在第 V-D 节中，我们将 L 从 2 调整到 128。VUS 和 FastVUS 均采用与原始 VUS 相同的 250 个均匀分布的阈值。除合成实验外，所有实验（包括 GPU 变体）均在一台笔记本电脑上运行（Intel Core Ultra 7 265H、64GB 内存，以及配备 8 GB 显存的 NVIDIA RTX PRO 1000 Blackwell GPU）；合成实验则在配备 AMD EPYC 7352 CPU（256 GB 内存）和 NVIDIA H100 NVL GPU（94 GB 显存）的集群上运行。

A. TSB-AD 上的准确性与速度

我们首先在 TSB-AD 的真实世界数据上比较所有评估指标。图 1a 报告了每条时间序列的平均运行时间以及整个基准的总耗时。原始 VUS 速度最慢，完成整个基准需要 7 小时 38 分钟；FastVUS 将其缩短至 7 分 41 秒，FastVUS-GPU 则缩短至 56 秒。我们将加速比报告为各时间序列加速比的平均值，因此每条时间序列的权重相同，也就不等于上述总耗时之比。按此计算，FastVUS 比 VUS 快 $75.91\times$，FastVUS-GPU 快 $303.75\times$。FastVUS-GPU 的速度快于其他范围感知指标 Range-AUC（$98.25\times$）和 Range F-score（$94.97\times$），且与基于点的 AUC（$1137\times$）和 Affiliation（$684.75\times$）的差距不到一个数量级。

我们将误差定义为本实现与原始 VUS 之间的绝对差值（图 6a）。FastVUS 在数值上精确无误，误差仅处于浮点精度范围内（最大为 2×10^{-15}）；FastVUS-GPU 的最大误差为 5×10^{-3}，由于存在近似项项（第 III-B1 节）。这是在 TSB-AD 的 27，840 个时间序列和检测器对中最大的误差。

最后，我们量化每个计算步骤对整体加速比的贡献。在这项消融实验中，我们使用单个检测器（不像上文那样使用所有检测器，因此加速比有所不同），并从原始 VUS 开始，逐步加入我们的技术。图 6b 显示，FastVUS 的每一步都带来了提升，其中有两步尤为突出。优化后的存在性计算带来了最大幅度的提升，使运行时间减少到原来的七分之一（从 $5\times$ 降至 $32\times$），这证实存在性计算是主要瓶颈。随后，全局掩码又使加速比翻了一倍以上（从 $54\times$ 提升至 $138\times$），由此得到 FastVUS。在图形处理器上执行相同的流程进一步叠加了这些收益，使 FastVUS-GPU 相较于原始 VUS 达到 $677\times$。

B. 合成数据上的可扩展性

接下来，我们使用合成数据评估运行时间如何随关键时间序列特征变化而扩展。图 7a 展示了运行时间随序列长度的变化。VUS 的速度最慢，当数据点数超过约 1 亿时便会失败；相比之下，FastVUS-GPU 的速度最快，即使处理测试中的最长序列（2^{31} 个点，超过 20 亿个样本），也只需不到两秒。此外，在达到 100 万个点之前，其运行时间几乎保持不变，这解释了图 1a 中的低方差。图 7b 考察了异常数量的影响。大多数方法的运行时间先增加，随后趋于平台期，其中 VUS 的增长最为明显（从 0.2 秒增至超过 40 秒）。FastVUS 呈现出类似趋势，在异常数量最大时达到 8 秒；而 Affiliation 的表现相近，但运行时间约低一个数量级。相比之下，RF、Range-AUC 和 AUC 基本不受异常数量影响。FastVUS-GPU 始终保持最快，即使异常数量最大时也只需 6 毫秒。最后，图 7c 表明，与序列长度或异常数量相比，异常长度对运行时间的影响较小。VUS 仍然最慢，其次是 RF、Range-AUC 和

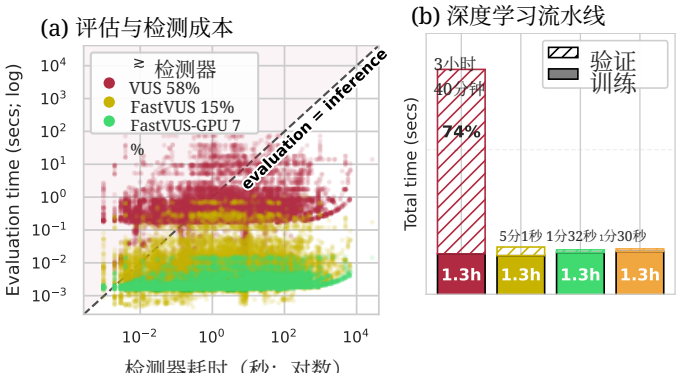


图 8：FastVUS 的实用性。(a) TSB-AD 上评估指标与检测器推理时间的对比，每个点对应一个时间序列与检测器配对。(b) 每种评估指标对应的 100 轮模型选择流水线的训练时间（实线）和验证时间（斜线）。每个柱状图上方标注总验证时间。

FastVUS；而 Affiliation、AUC 和 FastVUS-GPU 受影响最小。在所有设置下，FastVUS-GPU 的运行时间均最低，即使对于较长的异常，每个时间序列的运行时间也保持在 10 毫秒以内。由于篇幅所限，VUS、FastVUS 和 FastVUS-GPU 随最大缓冲区长度（从 0 到 512）变化的运行时间已在我们的代码仓库 [32] 中报告。

C. 实用性：评估是否是瓶颈？

评估与检测的成本。一个反复出现的问题是，评估 VUS 是否会成为瓶颈，因为检测器本身的运行成本可能很高。图 8a 通过绘制 TSB-AD 中每个时间序列与检测器组合的指标评估时间和检测器推理时间，回答了这个问题。对角线表示评估成本与检测成本相当；在线上方时，评估便是瓶颈。在 58% 的运行中，原始 VUS 的耗时与检测器处于同一数量级，甚至更慢。FastVUS 和 FastVUS-GPU 仅在 15% 和 7% 的运行中耗时相当，从而将评估耗时压缩到整个流程中的一小部分。因此，VUS 确实曾是一个瓶颈，而 FastVUS 消除了这一瓶颈。

在深度学习流水线中的集成。用于时间序列异常检测的深度学习方法发展迅速 [37]，，因此高效的评估指标至关重要。为展示 FastVUS-GPU 在这一场景中的优势，我们设计了一个模型选择流水线：使用来自 TSB-UAD 的 1,400 条时间序列训练一个 3 层多层感知机，训练 100 个训练轮次 [11]（其中 1,153 条用于训练，247 条用于验证）。每个训练轮次结束后，我们使用 AUC（通过*torchmetrics*在图形处理器上计算）、VUS、FastVUS 和 FastVUS-GPU 对模型进行评估。我们将缓冲区长度设为 512；对于以 256 赫兹采样的心电图（ECG）信号，这相当于围绕每个异常设置一个 2 秒的容差窗口。图 8b 展示了各项指标的运行时间分解。VUS 占据了流水线的大部分时间，仅验证阶段每个训练轮次就耗时 132 秒；相较于训练阶段，这带来了 282% 的额外开销，占流水线总时间的 74%。相比之下，FastVUS 将验证时间降至每个训练轮次 3 秒（额外开销为 6.5%），加速了 43.9×；而 FastVUS-GPU 则将其降至不足

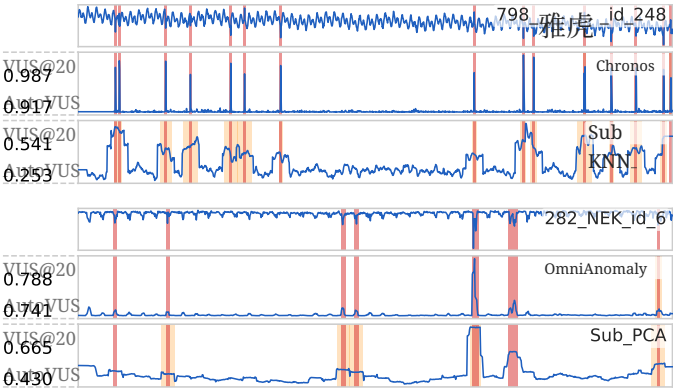


图9：两个时间序列及各自两个检测器的异常分数。红色标示已标注的异常，橙色标示 AutoVUS 选择的缓冲区。VUS@20 是带有 $L = 20$ 的 VUS-PR。

每个训练轮次 1 秒（额外开销为 2.0%），实现了与 AUC（145.7×）相当的 142.3× 加速。因此，两个 FastVUS 变体完成 100 个训练轮次约需 1.4 小时，与 AUC 相同；相比之下，使用 VUS 则约需 5 小时。

V. AutoVUS 实验评估

在本节中，我们评估 AutoVUS 如何改变时间序列异常检测评估。我们首先考察 TSB-AD 基准中的两个案例（第 V-A 节），然后使用受控的单异常和多异常案例来分离自适应缓冲的影响（第 V-B 节和第 V-C 节），最后比较整个 TSB

-

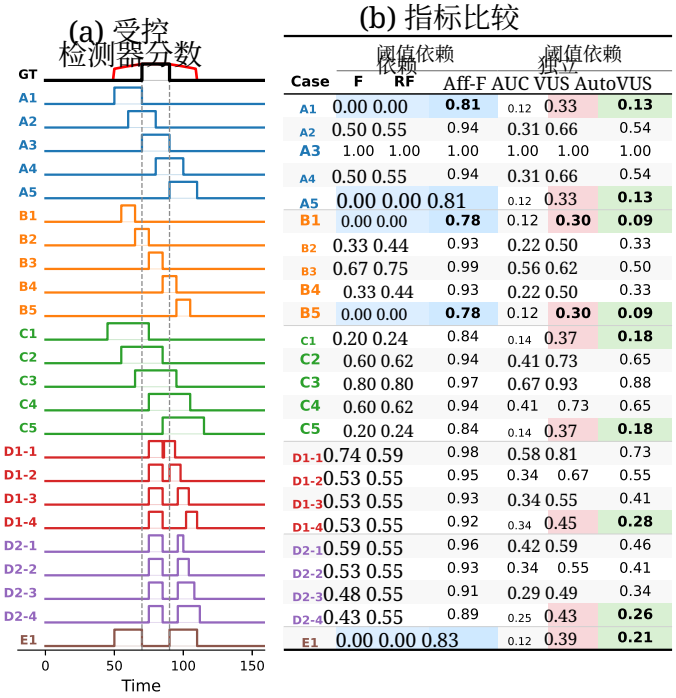
AD 基准（第 V-D 节）。

A. 真实时间序列案例研究

图 9 展示了 AutoVUS 对两个 TSB-AD 时间序列的影响：798 YAHOO id 248 和 282 NEK id 6，每个时间序列均使用两个检测器，如 TSB-AD [13] 中所实现。在第一个时间序列中，Chronos 对标记的异常作出明显响应，而 Sub KNN 则产生大范围的得分波动，且这些波动与异常并不总是吻合。在第二个时间序列中，OmniAnomaly 在标记的异常处出现峰值，而 Sub PCA 则形成更宽的高原，并延伸至异常周围。VUS@20 已能区分这两组检测器，将 Chronos（0.987）排在 Sub KNN（0.541）之上，并将 OmniAnomaly（0.788）排在 Sub PCA（0.665）之上。AutoVUS 保留了这一排序，并拉大了差距：Chronos 和 OmniAnomaly 的得分仍然较高，分别为 0.917 和 0.741；而 Sub KNN 和 Sub PCA 的得分则降至 0.253 和 0.430。这表明 AutoVUS 并非一概降低所有 VUS 值。相反，当检测器响应得到异常区域的支持时，它会保留较高得分；而当表面性能依赖于宽泛或定位不精确的缓冲区诱导奖励时，它会降低得分。

B. 受控案例研究

我们采用受控案例研究，以隔离自适应缓冲对评估所产生的影响。图 10a 顶部显示了一个真实异常，下方显示了多个合成检测器分数，并列出了具有代表性的阈值相关和阈值无关指标（图 10b）。



(b) 指标比较

Case	阈值依赖 F	阈值依赖 RF	阈值依赖 Aff-F	阈值独立 AUC	阈值独立 VUS	阈值独立 AutoVUS
A1	0.00	0.00	**0.81**	0.12	0.33	**0.13**
A2	0.50	0.55	0.94	0.31	0.66	0.54
A3	1.00	1.00	1.00	1.00	1.00	1.00
A4	0.50	0.55	0.94	0.31	0.66	0.54
A5	0.00	0.00	0.81	0.12	0.33	**0.13**
B1	0.00	0.00	**0.78**	0.12	**0.30**	**0.09**
B2	0.33	0.44	0.93	0.22	0.50	0.33
B3	0.67	0.75	0.99	0.56	0.62	0.50
B4	0.33	0.44	0.93	0.22	0.50	0.33
B5	0.00	0.00	**0.78**	0.12	**0.30**	**0.09**
C1	0.20	0.24	0.84	0.14	0.37	**0.18**
C2	0.60	0.62	0.94	0.41	0.73	0.65
C3	0.80	0.80	0.97	0.67	0.93	0.88
C4	0.60	0.62	0.94	0.41	0.73	0.65
C5	0.20	0.24	0.84	0.14	0.37	**0.18**
D1-1	0.74	0.59	0.98	0.58	0.81	0.73
D1-2	0.53	0.55	0.95	0.34	0.67	0.55
D1-3	0.53	0.55	0.93	0.34	0.55	0.41
D1-4	0.53	0.55	0.92	0.34	0.45	**0.28**
D2-1	0.59	0.55	0.96	0.42	0.59	0.46
D2-2	0.53	0.55	0.93	0.34	0.55	0.41
D2-3	0.48	0.55	0.91	0.29	0.49	0.34
D2-4	0.43	0.55	0.89	0.25	0.43	**0.26**
E1	0.00	0.00	0.83	0.12	0.39	**0.21**

图10：在时间偏移、长度不匹配和额外检测情况下比较评估指标的受控案例研究。(a) 真实异常和合成检测器分数。(b) 阈值依赖指标和阈值无关指标的取值。彩色单元格突出显示文中讨论的数值。

真实标注行上的红色曲线表示 VUS 所使用的最大软缓冲区。缓冲区长度和最大缓冲区长度均与异常长度相匹配，因此 VUS 与 AutoVUS 之间的任何差异都源于缓冲奖励的选择和归一化方式，而不是不同的容差尺度。由于 AutoVUS 经过机会校正，其数值无法直接与 VUS 的数值进行比较，因此本研究关注的是每种指标在各个案例中的数值分布。

前三组测试了标准的时间容差行为。A组保持预测长度与异常长度相等，并将其从较早位置逐渐移至较晚位置；B组使用较短的预测，C组则使用较长的预测。这些案例突显了依赖阈值的度量所发挥的不同作用。F分数不进行时间调整，对标注区间之外的检测赋予零分，即使它们靠近异常边界也是如此。若检测与标注区间没有重叠，RF也会采取保守评分。相比之下，Aff-F会因奖励预测事件与真实事件之间的时间接近度而保持较高得分。A1/A5 和 B1/B5 展示了这一点：F分数和 RF 均为零，而 Aff-F 仍然较高。

阈值无关指标展示了自适应缓冲的主要效果。在 AUC、VUS 和 AutoVUS 下，精确或几乎居中的检测会获得较高得分，而距离异常越远，检测得分越低。不过，当预测落在扩展后的标签内时，VUS 仍高于 AUC，因为固定缓冲会为接近边界的检测提供部分得分。AutoVUS 保留了这种有意义的时间容差，但会降低主要由更大的缓冲区域带来的提升所获得的奖励，因此拉大了精确检测与偏移检测之间的差距。这就是高亮显示的 A1/A5、B1/B5 和 C1/C5 行中，AutoVUS 值低于 VUS 的原因。不过请注意，AutoVUS 是在机会校正尺度上衡量的，因此任何高于 0 的 AutoVUS 值都表示，在相同的缓冲评估下，检测器获得的奖励高于随机分数。因此，A1 中像 AutoVUS $= 0.13$ 这样的数值看起来可能较小，但仍表明检测信号高于随机水平。接近边界的检测应获得一定得分，因为它在时间上接近异常，但其得分应低于定位准确的检测，例如 A3，其中检测器与异常完全匹配。

D1 和 D2 组引入了更具挑战性的设置，其中每个检测器都在异常内部包含一段部分正确的检测结果，并在异常外部包含一段额外片段。这些案例用于检验某个度量是否能够将有用的时间容差与额外检测区分开来，后者仅因落入缓冲区域而获得得分。由于标签扩展后异常外部的片段也能获得奖励，VUS 通常仍保持较高水平。AutoVUS 在这些案例中始终较低。突出显示的 D1-4 和 D2-4 行表明，异常外部的检测结果仍会提高 VUS，但当增益很大一部分来自缓冲区诱导的奖励时，AutoVUS 会更加保守。

最后，E 组是对虚假得分的压力测试。检测器在异常两侧一直触发到最大缓冲范围，但在真实异常区间内部没有触发。由于没有任何部分与异常重叠，F分数和 RF 均分配零得分，而 Aff-F 则因时间接近度而保持较高水平。类似地，AUC 较低，而 VUS 会因缓冲奖励而提高，这种膨胀现象会被 AutoVUS 所减弱。总体而言，该案例研究展示了 AutoVUS 的价值：它保留了 VUS 对时间不精确性的容差，同时使最终得分对仅通过过度缓冲区重叠获得的奖励不那么敏感。

C. 事件级可解释性

接下来，我们使用多异常受控案例来检验 AutoVUS 的事件级行为。除生成最终标量分数（此前各小节仅报告了这一分数）外，AutoVUS 还会为每个异常选择一个逐异常缓冲向量 $b^* = (b_1^*, \ldots, b_{N_\alpha}^*)$。这些缓冲区不仅用于最终的 AutoVUS 计算，还能为检测器行为提供可解释诊断。较小的 b_k^* 表明得分在第 k 个异常附近具有良好的定位性，而较大的b_k^* 则表明检测器需要更大的时间容差才能匹配该异常。因此，b^* 可以揭示检测器是分别响应单个异常事件，还是受益于较宽的缓冲区域。

图 11 展示了三个多异常案例，每个案例都包含异常标签、两个检测器分数，以及 AutoVUS 针对每个异常所选择的缓冲区。请注意，AutoVUS 会为每个异常选择一个缓冲区，并将其应用于两侧；逐侧选择则留待未来工作。在第一个案例中，异常标签包含四个密集事件。得分 $S1$ 分别检测出这四个事件，但每次都略有延迟，因此选定的

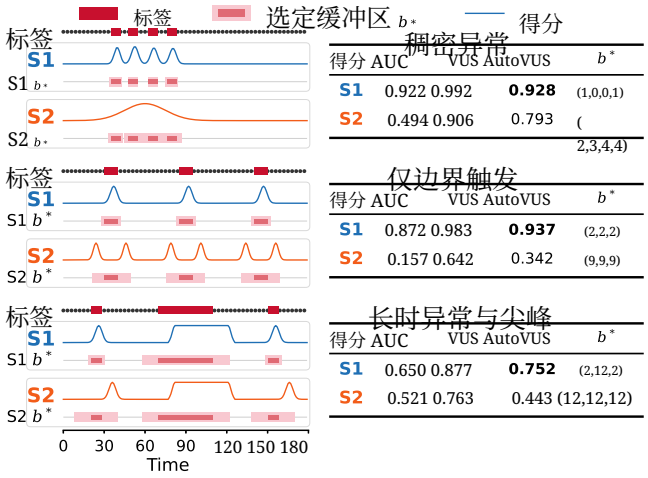


图 11：每个案例都在相同的异常标签序列下比较两个检测器分数。所选的逐异常缓冲向量 b^* 显示在每个分数下方，并用于计算 AutoVUS。该表报告最大缓冲区长度为 $L = 20$ 时的 AUC、VUS、AutoVUS 和 b^*。

缓冲区向量 $b^* = (1, 0, 0, 1)$ 仍然较小。相比之下，$S2$在密集区域上形成一条宽阔的桥接，并将附近的异常视为一个宽泛的事件，正如其缓冲区 $b^* = (2, 3, 4, 4)$ 所显示的那样，因为它在事件之间的间隔处与事件内部一样具有较高的响应。VUS 已经更偏好 $S1$(0。992 而非 0。906)，而 AutoVUS 进一步扩大了差距（0。928 对比 0。793）。这种行为是理想的，因为能够分别解析出四个异常的检测器，应当优于将它们合并为一个宽泛响应的检测器。

第二个案例评估仅边界触发。在此，$S1$ 在每个异常区间内侧边缘附近触发，而 $S2$ 在每个异常两侧之外触发。由于 VUS 会奖励落在扩展缓冲区域内的预测，$S2$ 仍获得了非平凡的 VUS 分数 0.642。不过，AutoVUS 更为保守，为 $S2$ 赋予 0.342，而 $S1$ 的得分为 0.937。这表明 AutoVUS 保留了对近边界检测的时间容差，但不会对主要在异常区间之外触发的分数给予过多奖励。

第三个案例包含一个长时异常和两个尖峰。$S1$ 能够检测到这三处异常，且滞后会随异常长度增加；而 $S2$ 在每处异常上的滞后量都相同。VUS更青睐 $S1$ （0.877 对比 0.763），AutoVUS则进一步拉大了差距（0.752 对比 0.443）。选定的缓冲区说明了原因：$S1$ 对应的缓冲区为 $b^* = (2, 12, 2)$，而 $S2$对应的缓冲区为(12, 12,12)。这表明缓冲区会针对同一时间序列中的各处异常分别调整，而单一固定缓冲无法做到这一点。

D. 全基准比较

最后，在图 12 中，我们比较了各检测器在 TSB-AD 基准上的全局排名，指标包括 AUC、最大缓冲区长度 L 从 2 到 128 的 VUS，以及取值为 $L = 128$的 AutoVUS。对整个基准计算 AutoVUS 需要 25 分钟。我们将检测器分为两组：尖锐组（带下划线），AutoVUS 平均在 85% 的异常上选择不使用缓冲区（$b^* = 0$）

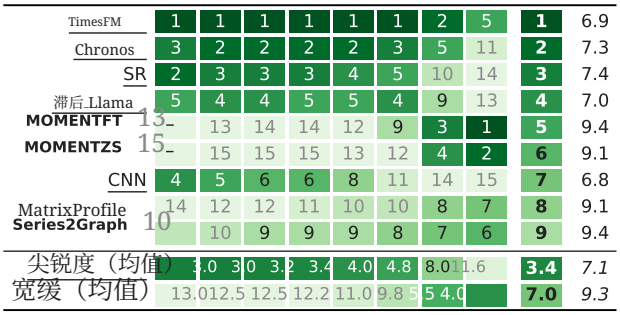


图 12：TSB-AD 上 AutoVUS 得分最高的九个检测器的排名。带下划线的名称表示在大多数异常上不选择缓冲区（尖锐型），粗体名称表示在大多数异常上选择缓冲区（宽缓冲型）。最后两行给出各组的平均排名。最后一列为平均选定缓冲区 b^*。

，另一组为宽泛组，AutoVUS 会在其中 68% 的异常上选择使用缓冲区。两组的平均排名表明，VUS 排名会随 L变化，因为较长的缓冲区更有利于宽泛型检测器，而 AutoVUS 则在两者之间取得平衡。VUS 下尖锐组的平均排名从 $L = 2$ 时的 3.0 变为 $L = 128$时的 11.6；宽泛组则从 12.5 变为 4.0，超过了尖锐组。AutoVUS 将尖锐组排在第一位，平均排名为 3.4；宽泛组排在第二位，平均排名为 7.0。

总体而言，AutoVUS 的排名彼此之间比不同 L 下的 VUS 排名更接近。AutoVUS 与 VUS@128 的全局排名之间，肯德尔 τ 为 0.72；而 VUS@128 与 VUS@16 之间则为 0.54。最后，平均而言，在 L中的 78% 时间序列上，AutoVUS 与 VUS 对最佳检测器的判断一致；在剩余的 22% 中，有 70% 的案例里，AutoVUS 下的最佳检测器已位列 VUS 排名前三。全部 32 个检测器的排名及更多结果见我们的代码仓库 [32]。

VI. 结论

曲面下体积（VUS）是一种面向时间的时间序列异常检测评估方法，但其计算成本限制了它在大规模场景中的应用。此外，VUS 对每个异常和检测器都使用相同的缓冲区长度，尽管不同案例所适用的时间容差可能有所不同。在这项工作中，我们通过引入**FastVUS** 和**FastVUS-GPU**解决可扩展性限制，并通过引入 **AutoVUS**解决缓冲区长度限制。前两种方法使 VUS 兼具时间感知能力和可扩展性，在 TSB-AD 上以完全相同的得分实现平均提速 $76\times$，而以最大误差 5×10^{-3}实现提速 $304\times$，其运行时间接近 AUC 等逐点评估方法。后一种是一种自适应评估方法，它以 VUS 为基础，通过与随机基线比较，为每个异常和检测器选择缓冲区，从而将 VUS 转变为逐异常评估方法，减少由缓冲区导致的得分虚高。我们将 FastVUS 和 AutoVUS 开源 [32]，以便社区能够在不再承担以往计算成本的情况下采用时间感知评估。

人工智能生成内容声明

作者使用大语言模型协助代码开发，并对文本进行编辑以提升清晰度。所有技术内容、实验、结果和结论均由作者完成、审阅和核实，作者对此论文负全部责任。

参考文献

[1]T. Palpanas 和 V. Beckmann, 《第一届和第二届跨学科时间序列分析研讨会（ITISA）报告》, SIGMOD 记录, 第 48 卷, 第 3 期, 第 36–40 页, 2019 年 12 月。 [在线]。网址：https://doi.org/10 1145 3377391 3377400 。/。S. Liu、T. Mangla、T. [2] Shaowang、J. Zhao、J. Paparrizos、S. Krishnan 和 N. Feamster, 《AMIR：在互联环境中通过视频和网络流量进行主动多模态交互识别》, 《ACM交互、移动、可穿戴和普适技术汇刊》, 第 7 卷, 第 1 期, 第 1–26 页, 2023 年。[3] E. Sylligardos、M. Sigalas、S. Soundoulounaki、K. Vaporidi 和 P. Trahanias, 《一种检测呼吸机过度辅助的深度学习方法》, 载于《国际模式识别与人工智能会议》。施普林格, 2022 年, 第 504–515 页。[4] S. Goddard、S. K. Harms、S. E. Reichenbach、T. Tadesse 和 W. J. Waltman, 《干旱风险管理的地理空间决策支持》, 《ACM 通讯》, 第 46 卷, 第 1 期, 第 35–37 页, 2003 年。[5] E. Page, 《关于参数在未知时点发生变化的问题》, 《生物计量学》, 第 44 卷, 第 1/2 期, 第 248–252 页, 1957 年。[6] A. J. Fox, 《时间序列中的离群值》, 《皇家统计学会杂志：B辑（方法论）》, 第 34 卷, 第 3 期, 第 350–363 页, 1972 年。[7] V. Barnett、T. Lewis 等, 《统计数据中的离群值》。威利，纽约，1994 年，第 3 卷，第 1 期。[8] S. Subramaniam、T. Palpanas、D. Papadopoulos、V. Kalogeraki 和 D. Gunopulos, 《使用非参数模型在线检测传感器数据中的离群值》, 载于《第 32 届超大型数据库国际会议论文集》, VLDB '06 系列。VLDB基金会，2006 年，第 187–198 页。[9] C.-C. M. Yeh、Y. Zhu、L. Ulanova、N. Begum、Y. Ding、H. A. Dau、D. F. Silva、A. Mueen 和 E. Keogh, 《矩阵剖面 I：时间序列的全对相似性连接：涵盖模式片段、异常片段和形状片段的统一视角》, 载于《2016 IEEE 第 16 届国际数据挖掘会议（ICDM）》, 2016 年, 第 1317–1322 页。[10] A. Bl´azquez-Garc´ıa、A. Conde、U. Mori 和 J. A. Lozano, 《时间序列数据中的离群点/异常检测综述》, 《ACM计算调查》, 第 54 卷, 第 3 期, 第 1–33 页, 2021 年。[11] J. Paparrizos、Y. Kang、P. Boniol、R. S. Tsay、T. Palpanas 和 M. J. Franklin, 《TSB-UAD：单变量时间序列异常检测的端到端基准测试套件》, 《VLDB基金会会刊》, 第 15 卷, 第 8 期, 2022 年。[12] S. Schmidl、P. Wenig 和 T. Papenbrock, 《时间序列异常检测：全面评估》, 《VLDB基金会会刊》, 第 15 卷, 第 9 期, 第 1779–1797 页, 2022 年 7 月。[13] Q. Liu 和 J. Paparrizos, 《房间里的大象：迈向可靠的时间序列异常检测基准》, 《神经信息处理系统进展》, 第 37 卷, 第 108 231–108 261 页, 2024 年。[14] J. Paparrizos、P. Boniol、T. Palpanas、R. S. Tsay、A. Elmore 和 M. J. Franklin, 《曲面下体积：一种新的时间序列异常检测准确度评估指标》, 《VLDB基金会会刊》, 第 15 卷, 第 11 期, 第 2774–2787 页, 2022 年。[15] V. Jacob、F. Song、A. Stiegler、B. Rad、Y. Diao 和 N. Tatbul, 《Exathlon：时间序列可解释异常检测基准》, 《VLDB基金会会刊》, 第 14 卷, 2626, 2021 年 11 月, 第 2613 页–。[16] S. Kim、K. Choi、H.-S. Choi、B. Lee 和 S. Yoon, 《迈向严格的时间序列异常检测评估》, 载于《AAAI人工智能会议》, 2022 年。[17] A. Lavin 和 S. Ahmad, 《评估实时异常检测算法——Numenta异常检测基准》, 载于《2015 IEEE 第 14 届机器学习与应用国际会议（ICMLA）》, 2015 年, 第 38–44 页。[18] R. Wu 和 E. J. Keogh, 《当前的时间序列异常检测基准存在缺陷，正在制造进步的假象》, 《IEEE知识与数据工程汇刊》, 第 35 卷, 第 3 期, 第 2421–2429 页, 2023 年。[19] T. Fawcett, 《ROC分析简介》, 《模式识别快报》, 第 27 卷, 第 8 期, 第 861–874 页, 2006 年。[20] J. Davis 和 M. Goadrich, 《精确率-召回率与ROC曲线之间的关系》, 载于《第 23 届国际机器学习会议论文集》, ICML '06 系列。美国纽约州纽约市：计算机协会，2006 年，第 233–240 页。 [在线]。网址：https://doi.org/10 1145 1143844 1143874 。/。N. Tatbul、T. J. Lee、S. Zdonik、M. [21] Alam 和 J. Gottschlich, 《时间序列的精确率和召回率》, 载于《神经信息处理系统进展》, 第 31 卷。Curran Associates, Inc.，2018 年。 [在线]。网址：https://proceedings.neurips.cc/paper/2018/hash/8f468c873a32bb0619eaeb2050ba45d1-Abstract.html[22] A. Huet、J. M. Navarro 和 D. Rossi, 《时间序列异常检测算法的局部评估》, 载于《第 28 届 ACM SIGKDD知识发现与数据挖掘会议论文集》, 2022 年, 第 635–645 页。[23] R. Salles、J. Lima、M. Reis、R. Coutinho、E. Pacitti、F. Masseglia、R. Akbarinia、C. Chen、J. Garibaldi、F. Porto 等, 《SoftED：时间序列事件检测软评估指标》, 《计算机与工业工程》, 第 198 卷, 第 110728 页，2024 年。[24] Y. Chen、C. Zhang、M. Ma、Y. Liu、R. Ding、B. Li、S. He、S. Rajmohan、Q. Lin 和 D. Zhang, 《ImDiffusion：用于多变量时间序列异常检测的插补扩散模型》, 《VLDB基金会会刊》, 第 17 卷, 第 3 期, 第 359–372 页, 2023 年。[25] Y. Yang、C. Zhang、T. Zhou、Q. Wen 和 L. Sun, 《DCdetector：用于时间序列异常检测的双重注意力对比表示学习》, 载于《第 29 届 ACM SIGKDD知识发现与数据挖掘会议论文集》, 2023 年, 第 3033–3045 页。[26] Z. Zhong、Z. Yu、X. Xi、Y. Xu、W. Cao、Y. Yang、K. Yang 和 J. You, 《Simad：一种基于不相似度的简易时间序列异常检测方法》, 《IEEE神经网络与学习系统汇刊》, 2025 年。[27] H. Si、J. Li、C. Pei、H. Cui、J. Yang、Y. Sun、S. Zhang、J. Li、H. Zhang、J. Han 等, 《TimeSeriesBench：时间序列异常检测模型的工业级基准》, 载于《2024 IEEE 第 35 届软件可靠性工程国际研讨会（ISSRE）》。IEEE，2024 年，第 61–72 页。[28] M. Goswami、K. Szafer、A. Choudhry、Y. Cai、S. Li 和 A. Dubrawski, 《MOMENT：一系列开源时间序列基础模型》, 载于《第四十一届国际机器学习会议，ICML 2024》, 奥地利维也纳，2024 年 7 月 21–27 日, 《机器学习研究论文集》第 235 卷。PMLR，2024 年，第 16 115–16 152 页。 [在线]。网址：https://proceedings.mlr.press/v235/goswami24a.html [29] Q. Ma、Z. Liu、Z. Zheng、Z. Huang、S. Zhu、Z. Yu 和 J. T. Kwok, 《时间序列预训练模型综述》, 《IEEE知识与数据工程汇刊》, 2024 年。[30] P. Boniol、A. K. Krishna、M. Bruel、Q. Liu、M. Huang、T. Palpanas、R. S. Tsay、A. Elmore、M. J. Franklin 和 J. Paparrizos, 《VUS：有效且高效的时间序列异常检测准确性度量》, 《VLDB期刊》, 第 34 卷, 第 3 期, 第 32 页，2025 年。[31] S. Romano、N. X. Vinh、J. Bailey 和 K. Verspoor, 《偶然校正聚类比较度量》, 《机器学习研究期刊》, 第 17 卷, 第 134 期, 第 1–32 页，2016 年。[32] E. Sylligardos、M. Huang、J. Paparrizos、P. Senellart 和 P. Boniol, 《FastVUS 和 AutoVUS 代码仓库》, https://github.com/sylligardos/autovus，2026 年。[33] E. Sylligardos、P. Boniol、J. Paparrizos、P. Trahanias 和 T. Palpanas, 《明智选择：时间序列异常检测模型选择的广泛评估》, 《VLDB基金会会刊》, 第 16 卷, 第 11 期, 第 3418–3432 页，2023 年。[34] W. M. Rand, 《聚类方法评估的客观标准》, 《美国统计学会期刊》, 第 66 卷, 第 336 期, 第 846–850 页，1971 年。[35] N. X. Vinh、J. Epps 和 J. Bailey, 《聚类比较的信息论度量：变体、性质、归一化与偶然校正》, 《机器学习研究期刊》, 第 11 卷, 第 95 期, 第 2837–2854 页，2010 年。 [在线]。网址：http: // jmlr.org/papers/v11/vinh10a.html [36] L. Hubert 和 P. Arabie, 《比较划分》, 《分类学期刊》, 第 2 卷, 第 1 期, 第 193–218 页，1985 年。[37] Z. Zamanzadeh Darban、G. I. Webb、S. Pan、C. Aggarwal 和 M. Salehi, 《时间序列异常检测的深度学习：综述》, 《ACM计算调查》, 第 57 卷, 第 1 期, 第 1–42 页，2024 年。