中国科学:信息科学 在审文章 onlinelatex

SCIENTIA SINICA Informationis

论文

智能攻击流量检测系统的训练加速方法：基于流量特征空间点云形心的数据集压缩

傅川溥[1]，李琦[2,3]，徐恪[1,3*]

1. 清华大学, 计算机科学与技术系，北京 100084
2. 清华大学, 网络科学与网络空间研究院，北京 100084
3. 中关村实验室, 北京 100194

* 通信作者. E-mail: xuke@tsinghua.edu

国家自然科学基金 (批准号: 62425201, 62221003, 62132011) 资助项目

摘要 攻击流量检测系统在海量的互联网流量中识别攻击流量，并拦截攻击流量保护合法用户。近年来，人工智能算法被广泛应用于攻击流量检测，它们从流量数据集中学习流量模式，能够检测隐蔽威胁流量，其准确度较传统固定规则检测有显著提高。然而，随着互联网流量规模的激增，流量数据集的规模也显著上升，这意味着在大规模流量数据集上训练攻击流量识别模型将消耗大量算力，伴随着巨大的部署时间延迟。

本文中，我们提出了一种攻击流量训练加速方法，旨在通过压缩流量数据，利用有限的计算资源在大规模流量数据集上快速训练攻击流量检测模型，使得检测系统快速地部署到网络。具体的数据压缩方法基于高维流量特征空间的几何结构。我们将数据集中的全体样本流量映射为高维流量特征空间中的点云，并将体素的概念拓展到高维流量特征空间中，通过计算体素中点的形心来代表全体点（流量样本），从而显著减小了数据集的规模。实验验证表明，该方法可以在 8 个数据集上为当前最先进的 10 种检测方法提升 75.82 倍的训练效率，同时几乎不影响检测的准确度和鲁棒性，并确保了压缩流量数据的实时处理。

关键词 网络安全, 人工智能, 攻击流量检测, 点云, 体素

1 引言

自二十一世纪开始，互联网在全世界迅速普及。目前，承载着千万用户的广域网环境中正传输着 Tb/s 规模的正常用户流量 [1]，同时其中也包含了攻击者生成的恶意流量，例如分布式拒绝服务攻击 [2~5]，网络扫描流量 [6~9]，以及恶意软件的通信流量 [10,11]。因此，从海量的互联网流量当中，检测、分析和拦截这些恶意流量成为了网络空间安全研究中的一个关键科学问题。

引用格式: 傅川溥, 李琦, 徐恪. 智能攻击流量检测系统的训练加速方法：基于流量特征空间点云形心的数据集压缩. 中国科学: 信息科学, 在审文章
Chuanpu Fu, Qi Li, Ke Xu. Accelerate Training of AI-Powered Attack Traffic Detection: Dataset Compression Based on Centroids of Point Clouds in Traffic Feature Spaces. Sci Sin Inform, for review

同时，随着人工智能应用的迅速普及，一系列基于人工智能的攻击流量识别算法被提出并得到实际落地应用[12~14]。这些系统借助人工智能算法，自动地分析数据包的特征[15,16]，例如数据包的长度和协议类型；同时也关注于分析数据包序列（被称为流）的特征[17,18,20]，例如流的完成时间和流中的数据包数量等等。最终实现实时的恶意流量检测。目前这些系统被大量部署在企业网关[16,19]等高速网络场景下，保护大量的合法用户。相比于传统的基于固定规则匹配的检测系统，基于人工智能的方法在实时性、准确度、和吞吐量上均有显著提升[21~23]。智能攻击流量检测系统是具备良好应用前景的新兴安全应用。

但随着互联网的普及，流量规模的激增[1]。因此，训练智能攻击流量识别系统的数据集的规模也在显著地上升[20,24~26]。而人工智能算法的训练，特别是深度学习算法的训练，大量消耗计算资源[15,27,28]，这意味着借助复杂模型在大规模流量数据集上构建智能攻击流量识别模型将消耗大量的时间和算力开销。本项研究工作探究，能否对海量的大规模流量和数据集进行预处理，压缩数据集规模，仅仅保存有信息量的流量样本。最后在这一压缩的数据集上训练攻击流量识别模型，降低训练时产生的计算开销，约束部署系统地延迟。

为此，本文提出了流量数据集压缩方法，**Centroids**，希望在保持原始流量数据信息的前提下，最大程度降低数据集的规模。由于本方案压缩了数据集的规模，训练算法处理数据集的计算开销下降，最终保证人工智能攻击流量识别模型被高速实时地训练。这一方案的核心思想是：将数据集当中的全体流量样本视为高维流量特征空间中的点云，对分布密集区域的点计算形心（Centroids）来表示附近的全体点（相似的流量样本），因此可以通过压缩点云的方式实现数据集规模的降低。

为实现高效流量数据集压缩，方案首先将流量特征转换为高维空间中的点，同时将三维空间中的体素概念扩展到这一高维流量特征空间中，即用高维空间中的立方体划分高维空间中表示流量训练样本的点。而后，对每个体素中进行基于密度的分类，分类处理包含不同密度的点的体素。之后，对表示高密度点的体素内部进行采样，初步降低计算开销。最后，对每一体素内部的点计算对应的形心，用形心代表体素中全部的点，显著降低了点的数量，从而降低了流量数据的规模。

在这一压缩后的数据集上，可以高效训练人工智能攻击流量识别模型，显著降低了计算开销。同时，这些压缩后的样本去除了原始数据集中的冗余信息，仍然保持了关键信息；即基于点云形心可以保持流量数据集在高维空间中对应的流形的基本拓扑关系。因此，在其上训练的模型不会存在显著的准确度下降或鲁棒性下降。

基于真实世界流量数据集的实验证明了，**Centroids** 可以为目前最先进的 10 个攻击流量检测系统，包括有监督系统/无监督系统，流粒度检测系统/包粒度检测系统，基于深度学习的系统/基于传统统计机器学习的系统，在 10 个数据集上降低 75.82 倍的训练时间。同时这一方法不会对训练模型的准确度造成显著的影响，例如仅存在 0.122% 的训练准确度下降。与此同时，研究还发现，这一方法不会对检测鲁棒性造成显著的影响，现有方法在 **Centroids** 压缩后的数据集训练后仍然可以对现有的全部逃逸攻击保持良好的鲁棒性。最后，该数据集压缩方法具有良好的实时性，可以在 5 秒内完成 10GB 规模的流量数据的实时压缩。总结以上，本文的技术贡献如下：

- 本文定义流量数据集压缩科学问题，旨在解决高吞吐互联网场景中部署智能攻击流量识别系统面临的数据集规模问题。
- 本文揭示了流量特征空间中样本稠密分布的特性，这是流量数据集压缩这一科学问题可解性的基础。
- 将流量样本映射为高维空间中的点云，并将三维空间中的体素概念拓展到高维流量特征空间，用于分析流量特征之间的拓扑关系。

- 对高密度体素中的点提出一种采样方法，计算形心以表示体素中的全体点，实现稠密点的去冗余，并保持数据集的高维流形结构。
- 在真实世界流量数据集上验证了压缩方法对训练效率的显著提升，同时不会对训练后的模型准确度产生显著影响。

本文的后续各节安排如下：第二节定义科学问题，并介绍解决这一问题的研究动机；第三节介绍方法的高层次设计；第四节形式化地描述方案的细节；第五节中，验证实验方法的有效性；第六节中，本文总结一系列相关的研究工作；第七节中，分析本研究的局限性并展望未来研究方向。最后，在第八节中，总结本项研究。

2　研究背景和问题定义

2.1　研究背景：智能攻击流量检测系统

攻击流量检测系统的目的在于从海量的互联网流量中区分出攻击流量，通过对攻击流量的有效拦截保护海量的互联网用户[15,16,18,19,28]。具体而言，这些系统通常被部署在因特网网关[29~31]，分析每一条进入网络的流量，并根据已知的流量特征区分出攻击流量[17,26]。早些年，这一类任务通常采用固定规则算法来完成[21,23,30]，其局限性在于无法应对未知的攻击，因为设计规则的人类专家无法预知未知的攻击。近年来，基于机器学习的攻击流量识别系统逐渐成为了主流方法，借助机器学习的泛化性识别未知的攻击模式，甚至借助无监督机器学习算法不依赖任何攻击流量样本检测未知攻击[15,32,33]。

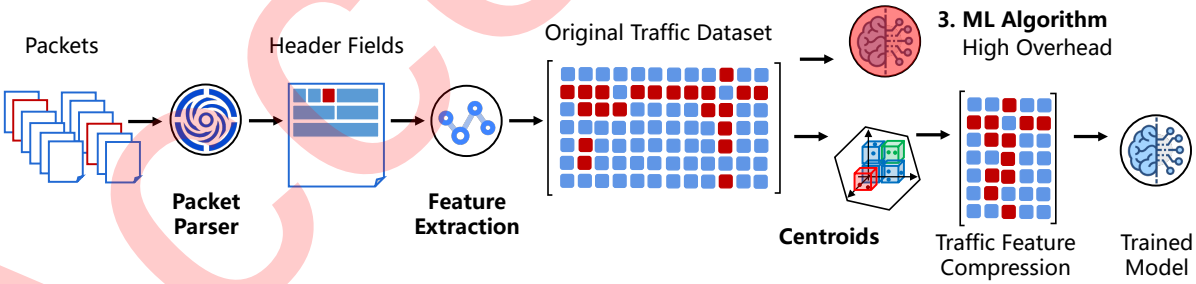
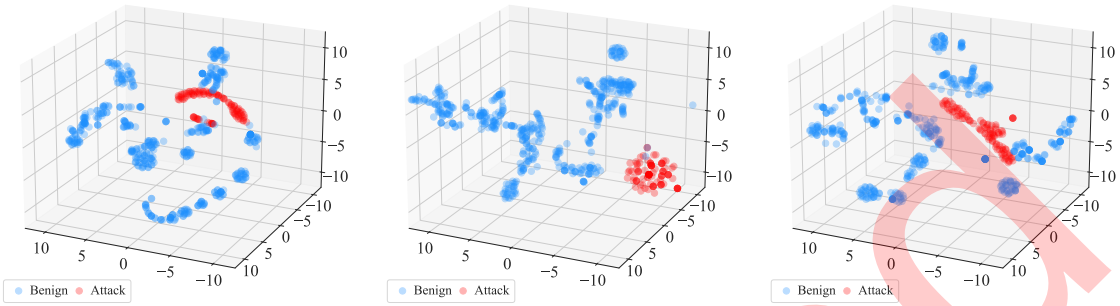


图 1　智能攻击流量检测系统高层架构

Figure 1　Architecture of AI Powered Malicious Traffic Detection Systems.

这类系统通常包含了三个关键模块：包解析器，流量特征提取，和机器学习算法，如图 1所示。首先，包解析器从海量的攻击流量中提取出关键的数据包字段，例如协议号和数据包的长度等。而后，特征提取器从这些数据包字中提取特征，作为机器学习算法的向量形式的输入。目前常见的特征包含两大类别：包级别特征，即从一个数据包抽取一个特征，例如数据包之间的到达间隔[15,34,35]；流级别特征，即从五元组相同的一组数据包抽取一个特征，例如 TCP 流中的数据包的数目等[16~19]。最后，机器学习算法在这些特征上进行训练，在测试阶段即可根据流量特征区分正常和异常流量。相比于有监督学习算法，无监督学习算法被更多地用于这一任务[15,18,32]，这是因为无监督学习算法只需要正常流量的特征进行训练，将全部偏离正常流量特征的流量均分类成为攻击流量，具备更好的泛化性。

然而，虽然经历了十余年的发展，这些攻击流量识别系统仍然仅被部署在小规模网络环境下[36,37]。在高带宽网络场景下处理海量的流量仍然是一个开放性的困难任务。因为在高带宽网络场景下，需



(a) HyperVision Datasets.　(b) Whsiper Datasets.　(c) NetBeacon Datasets.

图 2　在各个流量数据集上提取的 CICFlowMeter [46] 特征在空间中的分布。

Figure 2　Distributions of CICFlowMeter [46] traffic feature extracted from differnet existing datasets.

要系统具备良好的实时性 [15,26]，同时需要应对各种复杂的流量模式 [17,38]，考虑攻击者可能生成的逃逸检测的流量样本 [18]，以及识别隐蔽的加密通信方式发起的攻击 [10,19]。本研究旨在解决高带宽广域网场景下的数据集规模问题。

2.2　研究问题：压缩流量特征训练数据集

高带宽场景下部署攻击流量检测系统的关键挑战之一是：如何在高带宽场景下采集的大规模数据集上进行训练。目前的互联网转发节点的流量模型已经突破了 10Tb/s 级别 [1]，这些节点每分钟转发超过一千万个数据包和上万条流。另一方面，要想构建足够精准的攻击流量识别系统就需要采集数据集的时间足够长，采样的比率足够大 [17]。因此，训练攻击流量检测系统的数据集规模通常是巨大的，并且将伴随因特网流量规模的激增进一步上升。大规模的流量数据集不但为存储带来巨大压力，更使得在其上训练机器学习模型变成不可能完成的计算密集型任务，特别是对于近期被提出的基于深度学习的攻击流量检测系统 [15,27,28]。例如，Kitsune 方法采用堆叠自编码器结构，在高带宽场景下采集的 1 分钟流量数据集上进行训练将消耗超过一小时的时间 [15]，因为其采用包级别的特征，相比流级别的特征具备更大的规模。这使得及时地在网络中部署攻击流量检测系统成为开销巨大的困难任务。

因此，本研究考量这样一个科学问题，能否通过压缩流量数据集，在原始数据集之上构建一个规模显著较小的数据集，其中保留了原始数据集中的关键信息。由此，在压缩后的小规模数据集上进行训练可以达到在原始大规模数据集上训练相似的效果，且显著降低了在大规模数据集上训练模型产生的巨大算力开销。

3　研究动机与方案概要

3.1　设计动机：流量特征空间的稠密分布特性

我们发现，各种流量数据中提取的流量特征向量，在与之对应的流量特征空间中显然不是均匀分布的。相反，从图 2 中可以看出，这些特征向量会稠密地分布在某些区域中。我们在三个有代表性的流量数据集上提取了 CICFlowMeter 特征集中的特征 [46]，并使用 t-SNE 算法将这些特征向量从高维特征空间降维到三维空间。通过观察流量特征空间中流量数据集对应的流形 [16,18,20]，我们发现表示流量的点通常稠密地分布在某部分区域中。其原因是这些流量虽然不同，但均由同一种网络应

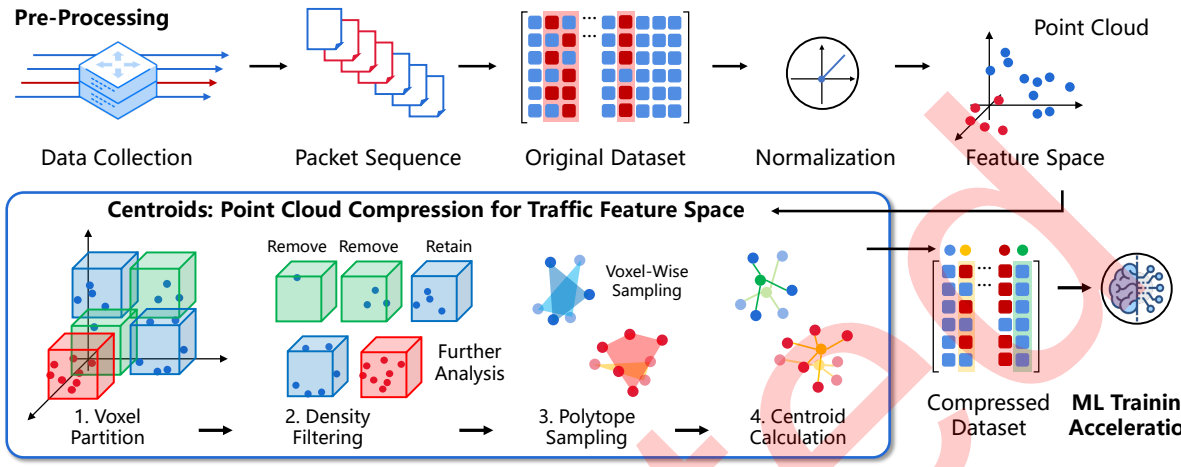


图 3　Centroids 高层次架构。

Figure 3　High-Level architecture of **Centroids** traffic dataset compression system.

用产生，其中必然存在一定的相似性，因而这些表示流量的高维点在空间中必然是邻接的。所以流量数据集存在显著的冗余性，也就是说，大量流量特征是相似的。因此，可以通过空间中的少数点来表示与之存在邻接关系的点，而没有显著的信息量损失。我们抽取了 NetBeacon 方案 [16]、Whisper 方案 [18] 和 FlowLens 方案 [43] 的特征，发现了类似的现象均存在。综上所述，在本项研究揭示了流量特征空间的稠密分布性质。这表明用少数点表示整个流量数据集是可行的。

3.2　设计方案概要

本项研究将流量数据集视为高维流量特征空间中的点云，并拓展三维空间中体素 [39,40] 的概念（Voxel），基于高维立方对点云进行分割。随后，对不同的高维立方测量密度，并以密度为依据分类处理这些立方。而后，对每个高维立方进行采样，最后计算采样点组成的高维立方体的形心（Centroid），这些形心代表了所有高维立方中的全体点。最终，压缩后的数据集将由这些形心组成。与原始数据集相比，由形心组成的压缩后数据集的规模显著下降，因此在其上训练模型具有更低的算力开销。另一方面，这些形心可以有效地保持高维流量特征空间中的流形几何结构，从而确保在其上训练的模型不会存在较大的精度损失。具体而言，这一方法包含了四个关键模块：

- **基于体素的空间分割：**将原始数据集归一化后，将每一个样本视为高维流量空间中的点。同时，将三维空间中体素的概念扩展到高维的流量特征空间，即使用高维立方体来划分整个流量特征空间。如此可以将点云分区域压缩，防止考虑全体点云的拓扑关系产生极大的性能开销。
- **点云密度筛选：**在这一模块中，方案根据点云密度对高维流量空间中的体素进行分类处理。具体地，方案直接丢弃表示离群点的体素，从而增强数据集的质量。同时，保留密度较低的体素中的所有点（流量样本），以防止信息丢失。此外，方案将进一步处理包含高密度点的体素。
- **多面体采样：**对于同一高密度体素中的所有点，本方案认为其中存在显著的冗余性，因此对其进行采样以初步降低其密度。具体而言，该方案进行多次采样得到若干个高维空间中的多面体，有效避免处理局部稠密点造成的高性能开销。
- **形心计算模块：**这一模块对每一个高密度体素中采样得到的高维空间中的多面体进行形心的计算，计算出的形心可以有效表征构成多面体的所有点，因为其在几何关系上同时趋近于这些点。最后，将流量特征空间中形心作为压缩后的样本加入压缩后的数据集中。

https://www.sciengine.com/doi/10.1360/SSI-2025-0025

之后，使用压缩后的数据集对攻击流量检测模型进行训练，可以降低计算开销，并且压缩后的点保持了高维流形中几乎全部的信息，不会对检测准确度造成显著的影响。

4　方案详细设计

在这一节当中，对 **Centroids** 的设计细节进行形式化描述。为方便表述，本文沿用现有流量分析工作中的数学表示方法 [18,41,42]。

4.1　预处理阶段

我们在预处理步骤将特征进行归一化，并将这些特征作为流量特征空间中的点进行处理。设 N 表示特征向量的数量，其等于原始流量数据集当中的样本的总数，是一个很大的数值。这些特征可能是流级别的特征 [16,28,31,41,43,44]，也可能是包级别的特征 [15,28,34]，对他们的处理方式是相同的，我们不关注特征具体表达的含义。

具体而言，使用 $\vec{s}_i = [\mathrm{s}_{i1}, \ldots, \mathrm{s}_{iM}]^{\mathrm{T}} (1 \leqslant i \leqslant N)$ 和 M 分别表示第 i 个特征向量及特征向量的长度。注意，不同方案的特征具有不同的 M 值 [16,31,45]。设矩阵 $\mathbf{S}$ 表示所有的特征，其中 $\mathbf{s}_{ij}$ 定义为第 i 个流量样本的第 j 个特征，例如流完成时间（FCT）[46] 和流中的数据包数量 [16]：

$$\mathbf{S} = [\vec{s}_1, \ldots, \vec{s}_i, \ldots, \vec{s}_N] = \begin{bmatrix} \mathrm{s}_{11} & \cdots & \mathrm{s}_{N1} \\ \vdots & \ddots & \vdots \\ \mathrm{s}_{1M} & \cdots & \mathrm{s}_{NM} \end{bmatrix}. \tag{1}$$

为了使特征在数值上稳定，便于进一步的分析，方案进行对数变换以减少特征范围并防止在点云分析过程中发生算术溢出。我们使用 $\mathbf{E} = \log_2(\mathbf{1} + \mathbf{S})$ 来表示结果。然后，我们通过对矩阵 $\mathbf{E}$ 的每一行进行最小-最大归一化，将 $\mathbf{E} = [\mathrm{e}_{ij}] (1 \leqslant i \leqslant N, 1 \leqslant j \leqslant M)$ 中的每个元素转换到区间 $[0, 1]$。现在，获得了由 N 个点 $\mathbf{P} = [\vec{p}_1, \ldots, \vec{p}_N]$ 表示的归一化特征向量：

$$\begin{cases} \vec{u} = [\mathrm{u}_j], \quad \mathrm{u}_j = \mathsf{min}(\mathrm{e}_{1j}, \ldots, \mathrm{e}_{ij}, \ldots, \mathrm{e}_{Nj}), \\ \vec{v} = [\mathrm{v}_j], \quad \mathrm{v}_j = \mathsf{max}(\mathrm{e}_{1j}, \ldots, \mathrm{e}_{ij}, \ldots, \mathrm{e}_{Nj}), \\ \forall i, j \quad 1 \leqslant j \leqslant M, \quad 1 \leqslant i \leqslant N, \end{cases} \tag{2}$$

$$\vec{p}_i = [\mathrm{p}_{i1}, \ldots, \mathrm{p}_{ij}, \ldots, \mathrm{p}_{iM}]^{\mathrm{T}}, \quad \mathrm{p}_{ij} = \frac{\mathrm{e}_{ij} - \mathrm{u}_j}{\mathrm{v}_j - \mathrm{u}_j}. \tag{3}$$

在此之后，将 $\mathbf{P}$ 视为高维流量特征空间当中的点云，矩阵当中的每一个列向量对应高维空间当中的点，即一个流量样本对应一个点，这些点构成高维流量特征空间当中的流形。 在进行了归一化后这些点可以被基于点云的特征压缩算法进行数值稳定的处理。

4.2　流量特征空间体素划分

在这一步骤中，我们将三维空间中体素的概念推广到高维的流量特征空间中。体素是三维空间中的立方体，用于表示覆盖范围内的全部点，通常被用来降低处理完整点云所造成的超高计算开销 [39]。因此，我们基于此理念构建数据集压缩算法，分析点云的密度，压缩冗余点。

https://www.sciengine.com/doi/10.1360/SSI-2025-0025

定义体素 $\mathscr{V}(\vec{a})$ 为在归一化的流量特征空间 $\mathfrak{F}=[0,1]^M$ 中边长为 ϵ 的 M 维小立方体，其中 $\vec{a}=[\mathrm{a}_1,\ldots,\mathrm{a}_M]^{\mathrm{T}}$ 是体素的索引。因此，该体素所覆盖的空间可以用笛卡尔积来表示：

$$\mathscr{V}(\vec{a})=[\epsilon(\mathrm{a}_1-1),\epsilon\mathrm{a}_1]\times\cdots\times[\epsilon(\mathrm{a}_M-1),\epsilon\mathrm{a}_M], \tag{4}$$

$$\forall\mathrm{a}_j\in\{1,\ldots,\lceil 1/\epsilon\rceil\},\quad (1\leqslant j\leqslant M). \tag{5}$$

此外，显然对于所有的 $\vec{a}$，$\mathscr{V}(\vec{a})\in\mathfrak{F}$ 均成立，并且不同体素的数量等于 $H=\lceil 1/\epsilon\rceil^M$，其中 $\lceil 1/\epsilon\rceil$ 是单个维度可排列的体素数目，其 M 次方是体素总数。

定义函数 $\xi(i,\vec{a};\mathbf{P})$，用于判断点 $\vec{p}_i$ 是否位于体素 $\mathscr{V}(\vec{a})$ 所覆盖的空间内：

$$\xi(i,\vec{a};\mathbf{P})=\begin{cases}\{i\}, & \text{if}\quad \forall 1\leqslant j\leqslant M,\quad \mathrm{p}_{ij}\in[\epsilon(\mathrm{a}_j-1),\epsilon\mathrm{a}_j),\\ \phi, & \text{else.}\end{cases} \tag{6}$$

将点集 $\mathbf{P}$ 聚集到由索引向量的全集 $\mathcal{A}=\{\vec{a}_1,\ldots,\vec{a}_H\}$ 索引的体素中。然后，定义函数 $\zeta(\vec{a};\mathcal{P})$，该函数输出位于由 $\vec{a}$ 索引的体素内的点的索引的集合：

$$\zeta(\vec{a};\mathbf{P})=\bigcup_{i=1}^{N}\xi(i,\vec{a};\mathbf{P}). \tag{7}$$

至此，本方案通过建立高维立方体，在高维流量空间中构建了体素，对表示流量数据集的密集点云进行了划分。通过这种基于体素的方法对点云划分，可以对大规模流量数据集在高维流量特征空间中形成的复杂流形分区域处理。

我们利用点之间的拓扑关系去除重复数据，同时需要防止处理大规模稠密点云之间复杂的拓扑依赖关系所带来的性能问题。由于这个原因，我们在构建点云时限制了点云之间拓扑关系的考虑范围。在同一个体素内，根据点云的密度不同，对其进行分类处理。

4.3　基于点云密度的体素分类处理

首先，对于表示离群点的体素，需要排除它们，因为学习离群点会造成严重的机器学习鲁棒性问题，从而使攻击者可以利用数据集中的噪声样本构造对抗样本[18,26]。具体而言，方案排除了表示少于 E 个点的体素。定义保留体素的索引为 $\mathcal{A}^*=\{\vec{a}_1^*,\ldots,\vec{a}_U^*\}$，满足对于所有 $\vec{a}^*\in\mathcal{A}^*$，$|\zeta(\vec{a}^*;\mathcal{P})|\geqslant E$，设 U 是保留体素的数量，E 是体素中点的最小数量。通过排除孤立点，**Centroids** 初步减少了点的数量，这可以有效减少处理开销。在实验部分，我们将介绍，这一流量压缩方案对检测的鲁棒性几乎无影响，攻击者无法通过向攻击流量中添加噪声的方式来干扰检测过程。

另一方面，定义包含超过 $D>E$ 个点的体素的索引为 $\mathcal{A}^\dagger=\{\vec{a}_1^*,\ldots,\vec{a}_V^*\}$，满足对于所有 $\vec{a}^*\in\mathcal{A}^\dagger$，$|\zeta(\vec{a}^*;\mathcal{P})|\geqslant D$，其中 V 是高密度体素的数量。显然 $\mathcal{A}^\dagger\subseteq\mathcal{A}^*$。随后，本方案将着重处理这些具备高密度的点云，它们是降低数据集规模的关键。

另外，对于点的数量超过 E 但小于 D 的体素 $\mathcal{A}^*-\mathcal{A}^\dagger$，保留其中全部的点。因为这些点既不是需要去除的离群点，也不是需要压缩的稠密点，其密度适中，基本不需要处理。定义这些点的坐标为：

$$\mathcal{E}_1=\bigcup_{\mathcal{A}^*-\mathcal{A}^\dagger}^{\vec{a}}\zeta(\vec{a};\mathbf{P}). \tag{8}$$

这些直接保留的点对应的特征被记做 $\mathbf{P}^{\S}=[\mathbf{P}_{\mathcal{E}_1}]$。

4.4　高维多边形采样

在本步骤中，该方案处理的对象是表征高密度点的体素，并对这些体素进行采样。通过采样，避免直接处理高密度点云造成的过高处理开销。现定义函数 $\mathsf{Sample}(\mathcal{S}, n; K)$，随机从一个包含元素均为整数的集合 $\mathcal{S}$ 中采样一个大小为 K 的子集，其中 n 是随机数种子，该采样函数希望在不同的随机数种子下尽可能地使得采样结果不同。

对于单个高密度体素，对其进行 T 次采样。每次采样构造一个包含 K 个点的集合，形成一个高维多边形。我们定义表示这一采样过程的函数：

$$\mathsf{Polyhedron}(\vec{a}; \mathbf{P}, T, K) = \bigcup_{i=1}^{T} \{\mathcal{U}_i\}, \tag{9}$$

$$\mathcal{U}_i = \mathsf{Sample}\left(\zeta(\vec{a}; \mathbf{P}); i, K\right). \tag{10}$$

这一函数将输出 T 组采样点的索引的集合，接下来方案将对 T 组采样得到的点，也就是 T 个包含 K 个顶点的高维多边形别计算形心，进而用形心来代表全部采样的点。

4.5　点云形心计算

在这一步中，计算每一组采样当中的点构成的多边形在高维流量空间当中的形心 [47]，即对坐标进行平均化计算得到的几何中心 (Centroid)，并采用这些形心表示全部多边形中的点。注意到，形心距离各点均较近，所以可以有效表示全部点，不会显著损失信息。因此，对于一个索引是 $\vec{a}_i^* \in \mathcal{A}^\dagger (1 \leqslant i \leqslant V)$ 体素中的 $|\zeta(\vec{a}_i^*; \mathbf{P})|$ 个点，本方案可以有效地用 T 个形心表示他们，因此可以显著降低点云的规模，即流量数据集的规模。

首先，定义形心计算函数，计算一个矩阵索引集合 $\mathcal{I}$ 指示的全部点的形心：

$$\mathsf{Centroids}(\mathcal{I}; \mathbf{P}) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \vec{p}_i. \tag{11}$$

而后，定义对任意一个体素 $\vec{a}_i \in \mathcal{A}^\dagger, 1 \leqslant i \leqslant V$ 得到的形心在高维流量特征空间的坐标是：

$$\mathbf{P}_i^\ddagger = [\vec{t}_1, \ldots, \vec{t}_T], \quad 1 \leqslant i \leqslant V, \tag{12}$$

$$\begin{aligned} \mathcal{Z}^{(i)} &= \mathsf{Polyhedron}(\vec{a}_i; \mathbf{P}, T, K), \quad 1 \leqslant i \leqslant V, \\ \vec{t}_j &= \mathsf{Centrodis}(\mathcal{Z}_j^{(i)}; \mathbf{P}), \quad 1 \leqslant j \leqslant K. \end{aligned} \tag{13}$$

最后，将全部的形心构成的矩阵进行拼接，得到表示对高密度体素对应的点进行点云压缩后得到的表示点云的矩阵。在这里，函数 $\mathsf{Concatenate}(\cdot)$ 对第一个维度进行拼接操作：

$$\mathbf{P}^\ddagger = \mathsf{Concatenate}(\mathbf{P}_1, \ldots, \mathbf{P}_V). \tag{14}$$

最后输出对原始数据集 $\mathbf{S}$ 进行压缩后的数据集是 $\mathbf{C} = \mathsf{Concatenate}(\mathbf{P}^\S, \mathbf{P}^\ddagger)$。

注意到，输出的数据集是归一化后的数据集，因此在训练机器学习算法时无需重复进行归一化操作。最终，在压缩后的数据集 $\mathbf{C}$ 上进行训练的开销要远远小于在 $\mathbf{S}$ 上进行训练的开销。并且两个数据集输出的格式 M 是完全一致的，压缩后的数据集可以直接用作算法的输入。

https://www.sciengine.com/doi/10.1360/SSI-2025-0025

5　实验验证

在这一节中，验证 **Centroids** 对于智能攻击流量检测系统的训练效率的增益。实验结果显示，这一方法可以对数据集进行有效的压缩，同时提升训练数据集的质量，最终使得 10 种最先进的检测系统在 8 个公开数据集上的训练效率平均提升 75.82 倍。同时，此方法仅对检测的准确度造成微小的影响，即保证了检测模型的准确性和鲁棒性。

5.1　实验设定

5.1.1　软件实现

本研究使用超过两千行的 C++ 17 和 Python 3.10 源代码实现了 **Centroids** 以及其性能分析模块。同时，我们利用 libpcap++（版本 22.05）来实现网络组件，用于数据包解析和特征提取（由 GCC v9.4.0、Ninja v1.10.0 和 CMake v3.16.3 编译）。同时，使用 PyTorch（版本 1.11.0 ，适用于 CUDA v11.3 ）来实现基于神经网络进行检测的模型。此外，对于现有方法采用 mlpack v4.5.3 来实现其中的传统统计机器学习算法。

5.1.2　硬件平台

研究团队在一台配备两个 Intel Xeon E2699 v4 CPU、512GB DDR4 内存、Intel 82599SE 网卡（2 × 10Gb/s SFP+ 收发器）以及 Ubuntu v20.04.2（Linux v5.15.0）的 DELL 服务器上部署了原型。深度学习模型在一块 Tesla V100 GPU（32 GB 内存，驱动程序 v470.103.01）上进行训练和执行。与此同时，使用了光纤电缆将这台服务器与另一台配置类似的服务器连接起来，在两台服务器之间按原始速度重放现有的公共数据集。

5.1.3　数据集

实验使用现有数据集来评估本研究，选取了八个具有代表性的数据集，这些数据集被广泛地应用于评价智能攻击流量检测系统的性能，其涵盖了不同的网络环境：

- **HyperVision 数据集** [19]：从 10Gb/s 的光纤链接收集的数据，包含利用真实漏洞进行攻击时生成的加密流量。该数据集中的加密攻击分为三个小类别：恶意软件、Web 攻击、以及针对加密协议的综合攻击（MISC）。
- **NetBeacon 数据集** [16]：在私有网络环境下收集，涵盖多种高速链路场景下的攻击流量，通常被用于测试高带宽网络下的检测效果。
- **Whisper 数据集** [18]：涵盖洪范攻击及其侦察测量步骤，如链路洪范攻击（Link Flooding Attacks, LFAs [53]）和脉冲 TCP DoS 攻击 [2, 54]。
- **IDS 数据集** [24]：由一个加拿大安全研究机构（Canadian Institute for Cybersecurity，CIC）在 2017 年采集。数据集当中包含了典型的网络攻击，虽然其流量模式相对简单，但这一数据集被广泛地用于验证入侵检测系统的准确度。
- **Kitsune 数据集** [15]：包含针对物联网设备的攻击流量，例如通过 Telnet 的指令注入攻击和从物联网设备发起的洪范攻击等。
- **CTU 数据集** [48]：从校园网络收集的网络流量数据，其中包含了大量真实用户的行为和恶意软件生成的流量，是攻击流量识别领域的经典数据集。

https://www.sciengine.com/doi/10.1360/SSI-2025-0025

• **CIC 收集的其他数据集**[49~51]：是近期公开的数据集，包括 DNS-over-HTTP 隐蔽通道、Android 恶意软件、物联网网络中的隐蔽攻击。

此外，鲁棒性分析实验还重放了 48 个逃避攻击的数据集以进行检测鲁棒性分析。总结以上，上述流量数据涵盖了各种网络环境，可以验证 **Centroids** 在各种网络环境中的性能增益。

5.1.4　基准系统

我们选择 10 个有代表性的现有智能攻击流量检测系统，测量 **Centroids** 对这些系统的训练加速效果，这些系统涵盖了有监督和无监督的方法。具体而言，实验选择了分析数据包特征[15,34,55]、流特征[16,35,43]和主机特征[28]的检测方案。我们对开源方法[15,18,19]直接部署，并实现了闭源方法[26,30]的原型。对只有在特定硬件上才可运行的方法[16,28]，实验对其进行软件模拟。为便于理解实验设定，在此简要介绍代表性系统：

• **CICFlowMeter 特征集**[46]：其中包含了超过 80 个有效的流级别特征，例如流持续时间等。为了实现端到端检测，使用了随机森林算法来学习 CICFlowMeter 特征集，因为随机森林类算法被现有方案大量采用[17,43,52]，现有研究也采用类似方法构建基线算法[19,67]。

• **HorusEye 方案**[64]：这一方案一部分功能在可编程交换机上实现，采用无监督的随机森林对多种数据报的特征进行分类。

• **Taurus 方案**[35]：在 FPGA 芯片上实现了向量计算电路，可在高速网络环境下进行快速推理的机器学习算法。这一系统主要采用了简单的数据包特征，并使用支持向量机算法对其进行分类。

• **N3IC 方案**[28]：在智能网卡上实现了二值化的神经网络，可以在高速网络环境下实时推理。该方法主要采用了粗粒度的主机级统计特征。

• **FlowLens 方案**[43]：这一流量分析方法涵盖了流级别的分布特征，例如一条链接中全部数据包的包长分布，并采用整数进行有效表示以降低存储开销，最终采用决策树算法进行分类。

• **NetBeacon 方案**[16]：这一方法在可编程交换机上实现，提取了流级别的特征，并在数据面上采用决策树算法进行有监督的分类，能有效识别具备洪范模式的攻击流量。

• **RAPIER 方案**[55]：这一方法基于循环神经网络，构建无监督模型以检测未知的网络攻击。这一方法学习细粒度的包级序列特征，同时解决数据集的标签不准确问题。

• **FSC 方案**[18]：是攻击流量识别的著名基线方案，采用简单的流级别粗粒度统计特征，使用无监督的 K-Means 算法进行检测。

• **nPrintML 方案**[34]：是具备代表性的包级别检测方法，将每一个数据包转换成一个由 0、+1、-1 组成的向量，随后采用自动机器学习（AutoML）方法有监督地学习这些特征。

• **Kitsune 方案**[15]：方案采用无监督的堆叠自编码器学习 115 维的数据包特征，其中包含了与 IP 地址相关的统计量。

5.2　训练速度增益

我们在 **Centroids** 方法压缩后的数据集上训练各个攻击流量检测模型，并对训练的速度进行比较。表 1 显示了相比于在未经压缩的数据集上直接训练，在压缩后的数据集上训练对训练时间降低的倍率。

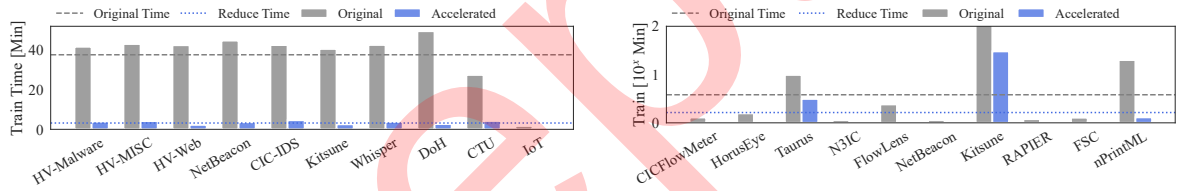
对于不同检测算法的训练速度增益。首先，实验观察到对于不同的检测方法，本文提出的方法能够实现至少 10.51 倍的训练加速。特别地，我们的方法为 nPrintML 方法[34]的训练提供了 275.83 倍的加速，主要原因是该方法采用的是包级别的特征，其特征规模较其他流级别特征更大，因此压

表 1　Centroids 方法对训练速度的提升。

Table 1　Improvements on training speeds.

Methods	HyperVision Datasets			Existing Datasets				CIC Datasets			Overall
	Malware	MISC	Web	NetB.	CIC-IDS	Kitsune	Whisper	DoH	CTU	IoT	
CICFlowMeter	19.48	18.76	28.67	98.01	28.54	37.56	32.71	20.14	15.55	9.78	30.92
HorusEye	11.06	11.63	22.68	40.97	15.65	19.49	13.69	22.68	18.72	4.52	18.11
Taurus	125.59	3.67	182.14	40.00	1.68	4.02	4.23	19.95	2.59	85.69	46.96
N3IC	21.85	51.19	66.76	189.42	79.58	64.17	83.54	2.20	1.18	2.13	56.20
FlowLens	174.66	154.41	510.37	578.56	148.54	240.42	319.05	3.03	1.33	2.10	213.24
NetBeacon	20.77	56.00	63.42	190.32	66.79	58.77	83.21	1.74	1.30	1.61	54.39
Kitsune	11.39	11.87	17.99	12.46	10.27	16.51	12.72	15.00	6.65	5.93	12.08
RAPIER	28.76	23.84	42.18	116.95	33.77	45.45	38.69	27.86	29.89	12.45	39.98
FSC	7.98	9.48	9.63	17.29	10.22	4.53	9.47	8.44	13.72	14.35	10.51
nPrintML	75.25	186.67	598.81	258.29	590.76	131.20	206.32	587.59	16.30	107.14	275.83
Average	49.68	52.75	154.26	154.23	98.58	62.21	80.36	70.86	10.72	24.57	75.82

[1] We highlight the best results in ● and the worst results in ●.



(a) Compare speeds on different datasets.　(b) Compare speeds for different detection methods.

图 4　流量数据集前后的训练时间开销对比。

Figure 4　Comparing speed improvements before and after **Centroids** compression.

缩效果更为明显。此外，还观察到对于 FlowLens 方法 [43]，该方案也实现了超过 200 倍的训练加速效果，原因是 FlowLens 的特征向量长度相较于其他方法更长，从而使得加速效果更为显著。

在不同数据集上的加速效果。另一方面，比较在不同数据集上的加速效果，可以发现，在 NetBeacon [16] 和 CIC-IDS [24] 两个数据集上的加速效果更为明显，这是因为本文提出的方法可以显著压缩其中的洪范流量。具体而言，这些数据集中的洪范流量在流量特征空间中集中在少数若干个体素上，因此可以高效地压缩。相反，在 CTU 数据集上，产生的加速效果相对较弱，这是因为数据集本身规模较小，在其他高带宽网络环境下采集的数据集上，方法产生的加速效果则相对较高。

加速前后的时间开销对比。从图 4(a) 中观察到，对于各个数据集上训练的不同算法，**Centroids** 可以将其平均训练时间从 41.35 分钟降低到 4.59 分钟，从而显著提升了模型部署的效率。例如，对恶意软件数据集，在其上训练各种智能攻击流量检测方案的平均时间高达 249 分钟，这是因为这一数据集采集自高速光纤网络，其流量规模很大，而本文提出的方法压缩原始的数据集，使得这些方法在压缩后的数据集上训练的时间开销仅为平均 23 分钟。另外，从图 4(b) 中还观察到，对于一些算法，其训练开销巨大，例如，Kitsune 基于深度学习的算法在各个数据集上训练的时间开销高达 344 分钟，而 **Centroids** 压缩方法可以将训练的时间降低到大约 29 分钟。综合以上，本研究使得攻击流量检测系统可以更快地部署到各类网络中。

数据压缩比率分析。接下来详细分析对数据的压缩效果。从表 2中可以看出，**Centroids** 可以对各种方案抽取的流量特征达到 66.23 倍的平均压缩比率，因此可以通过显著的数据规模压缩降低训练开销。特别地，实验发现这一方法对于 Kitsune 检测方法 [15] 和 RAPIER 检测方法 [55] 可以压

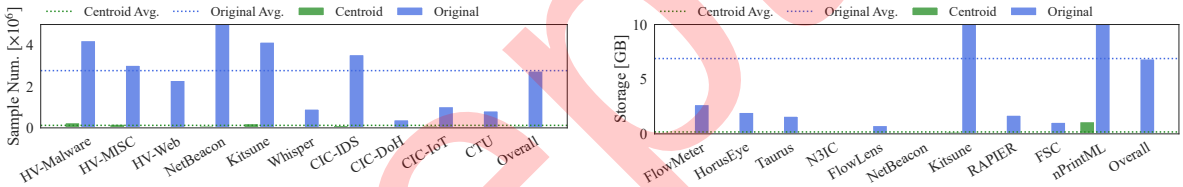
https://www.sciengine.com/doi/10.1360/SSI-2025-0025

表 2　**Centroids** 方案对不同种类特征和数据集的压缩的比率。

Table 2　Ratios of compression by **Centroids** for different kinds of features on various datasets.

Methods	HyperVision Datasets			Existing Datasets				CIC Datasets			Overall
	Malware	MISC	Web	NetB.	CIC-IDS	Kitsune	Whisper	DoH	CTU	IoT	
CICFlowMeter	15.77	15.55	29.06	64.26	18.64	47.78	26.86	25.88	9.61	11.39	26.48
HorusEye	32.31	30.32	56.71	72.16	32.59	89.81	43.66	57.94	20.34	24.63	46.05
Taurus	16.64	16.39	30.30	65.03	19.43	49.09	27.87	26.98	10.17	12.39	27.43
N3IC	19.31	18.90	36.23	73.69	22.18	55.76	32.49	31.68	11.42	14.18	31.58
FlowLens	51.89	49.40	117.32	181.28	53.45	191.66	100.01	100.55	59.75	88.12	99.34
NetBeacon	34.27	33.24	70.74	128.44	36.71	115.33	61.88	59.78	28.30	39.16	60.79
Kitsune	115.92	117.98	197.71	125.59	96.73	171.18	133.33	162.92	40.43	17.42	117.92
RAPIER	80.55	76.75	168.01	284.56	85.83	228.52	147.75	165.32	61.46	90.33	138.91
FSC	54.20	49.41	104.41	117.96	52.02	168.46	77.37	102.38	46.41	57.92	83.05
nPrintML	30.45	30.89	32.12	31.62	30.83	28.13	32.33	29.75	37.02	24.41	30.75
Average	45.13	43.88	84.26	114.46	44.84	114.57	68.36	76.32	32.49	38.00	66.23

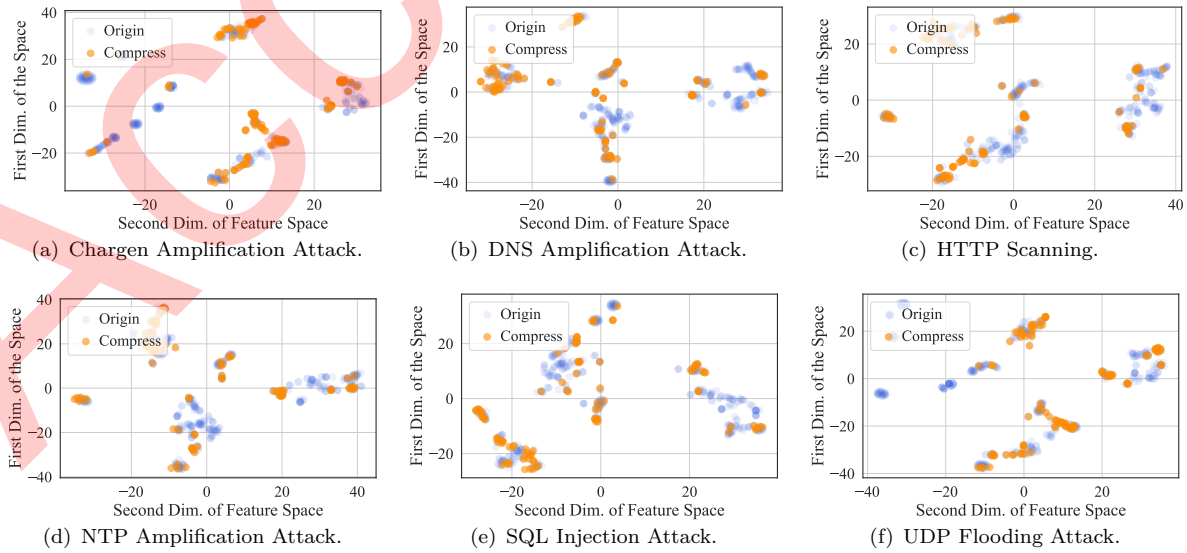
[1] We highlight the best compression ratios in ● and the worst ratios in ●.



(a) Number of samples.　(b) Storage overheads.

图 5　压缩前后的样本数量和存储开销对比。

Figure 5　Comparing numbers of samples and storage overheads after the compression.



(a) Chargen Amplification Attack.　(b) DNS Amplification Attack.　(c) HTTP Scanning.

(d) NTP Amplification Attack.　(e) SQL Injection Attack.　(f) UDP Flooding Attack.

图 6　数据压缩前后流量特征空间对比。

Figure 6　Comparing traffic feature spaces before and after **Centroids** compression.

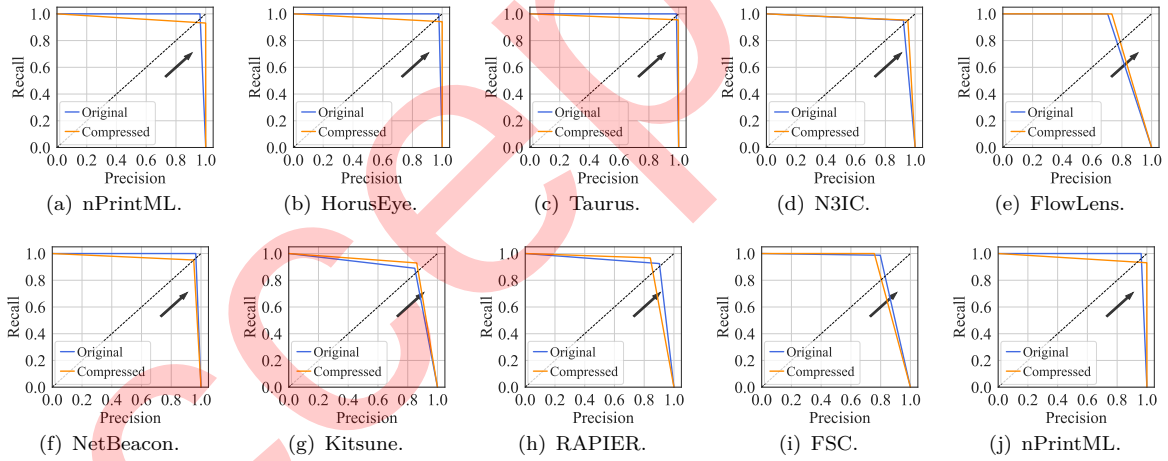
缩超过两个数量级的特征，这是因为这两个方法大量采用包粒度的特征，这些特征的设计相对比较简单，因此在流量特征空间中分布稠密，很容易被点云分析算法压缩。相比之下，对于复杂的流特征压缩效果相对较差。例如，对于 CICFlowMeter 特征 [46] 的压缩比仅为 26.48 倍，因此算法对这些采用复杂特征的算法的训练速度增益也相对较差。从数据集这一维度来看，**Centroids** 的算法在

https://www.sciengine.com/doi/10.1360/SSI-2025-0025

表 3　训练数据集压缩对检测准确度的影响。

Table 3　Impacts on detection accuracy when enabling **Centroids** dataset compression.

Methods	HyperVision Datasets			Existing Datasets				CIC Datasets			Overall
	Malware	MISC	Web	NetB.	IDS	Kitsune	Whisper	DoH	CTU	IoT	
CICFlowMeter	-0.030	-0.000	0.280	-5.730	-0.000	-0.000	-0.000	-0.570	-0.000	-0.000	-0.605
HorusEye	-0.850	3.200	3.910	0.500	-3.060	2.560	1.260	5.860	-5.680	-4.400	0.330
Taurus	-0.230	-0.070	-0.010	-4.720	-0.000	0.180	-0.000	-3.610	-0.170	3.150	-0.548
N3IC	0.060	-13.240	-0.250	0.020	-0.030	-0.000	-0.000	-10.560	-0.000	-1.000	-2.500
FlowLens	0.170	-0.000	-0.000	-0.000	-0.000	-0.300	-0.000	0.450	0.160	-0.000	0.048
NetBeacon	-0.020	-0.160	-0.200	-0.300	-0.010	-0.020	-0.050	9.060	-2.270	0.380	0.641
Kitsune	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000	-0.000
RAPIER	-0.000	-0.000	-0.000	-2.620	-0.000	-0.010	-0.000	0.040	-0.000	-0.000	-0.259
FSC	-0.000	-0.240	-0.000	0.090	0.670	-0.650	0.200	-0.000	3.290	5.700	0.906
nPrintML	-0.000	-0.020	-0.020	-0.030	-0.010	-0.860	-0.000	0.110	8.600	-0.090	0.768
Average	-0.090	-1.053	0.371	-1.279	-0.244	0.090	0.141	0.078	0.393	0.374	-0.122



(a) nPrintML.　(b) HorusEye.　(c) Taurus.　(d) N3IC.　(e) FlowLens.

(f) NetBeacon.　(g) Kitsune.　(h) RAPIER.　(i) FSC.　(j) nPrintML.

图 7　**Centroids** 压缩训练数据前后准确度-召回率曲线对比。

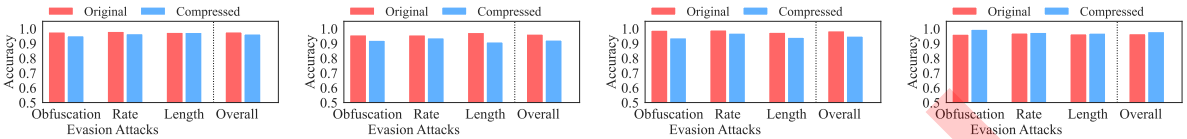
Figure 7　Precision-Recall curves before and after the compression.

NetBeacon 数据集和 Kitsune 数据集上的压缩比率更大，这是因为这些数据集中存在较多的洪范流量，它们之间的流量特征十分相似，很容易被压缩。

样本数量和存储开销降低效果。此外，本章还从样本数量和存储开销角度测量了压缩效果，结果如图 5 所示。首先，对于 CICFlowMeter 特征，各个数据集中的平均样本数在压缩前是 2.76×10^6 个，压缩后是 0.10×10^6 个，因此该方案可显著降低特征规模。此外，图 5(a) 测量了不同方案在处理 HyperVision 恶意软件数据集时的内存消耗量，可以看出压缩前的特征平均存储开销是 6.88GB，在压缩后的平均存储开销是 0.17GB，因此本方案可以将特征的存储开销降低 40.47 倍。

特征压缩可视化分析。在图 6绘制了启用 **Centroids** 进行数据集压缩和启用之前的流量特征空间的分布，其中蓝色的点代表未压缩之前的特征（随机采样 500 个），橙色的点代表数据压缩后的数据集（随机采样 100 个），例如图 6(b) 展示了利用 DNS 实现漏洞欺诈协议向受害者生成大量流量的反射放大攻击 [56]。可以发现，采样后的数据点大致上表示了采样前流量特征的流形，并且压缩后的点准确地出现在未压缩之前的高密度点云区域中。这是因为这些点云区域的点存在更多的冗余。**Centroids** 方法采用了少数若干个点对其进行有效表示，本质上是一个去冗余的过程。通过这一过

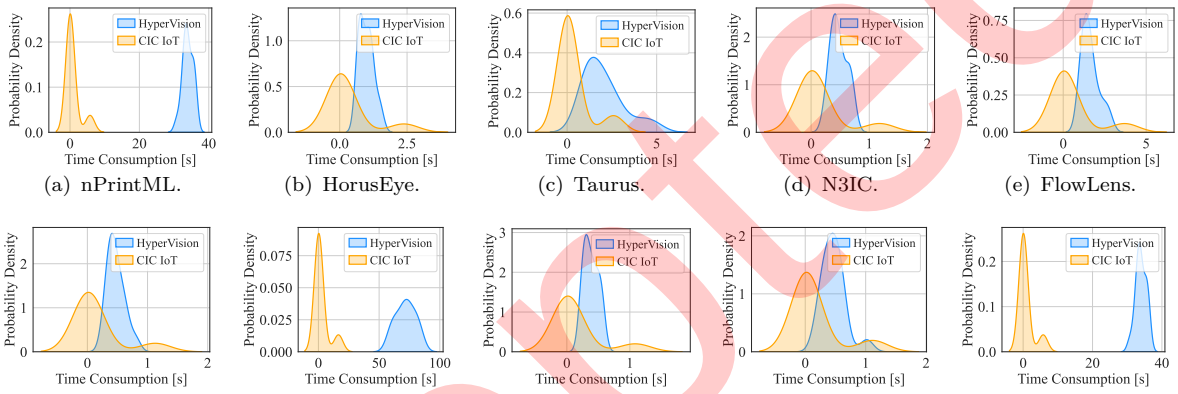
https://www.sciengine.com/doi/10.1360/SSI-2025-0025



(a) AUC. (b) Accuracy. (c) Precision. (d) Recall.

图 8　Centroids 训练加速对检测鲁棒性的影响。

Figure 8　Impcats on robustness when enabling **Centroids** for compresing training datasets.



(a) nPrintML. (b) HorusEye. (c) Taurus. (d) N3IC. (e) FlowLens.

(f) NetBeacon. (g) Kitsune. (h) RAPIER. (i) FSC. (j) nPrintML.

图 9　Centroids 对不同种类流量数据集压缩速度对比分析。

Figure 9　Speeds of **Centroids** traffic dataset compression for various detction methods.

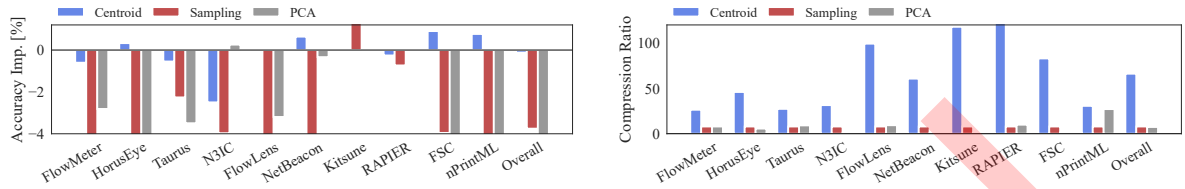
程显著降低了点的数量，即数据集的规模；同时保持了点之间的拓扑关系，即防止原始数据集信息被丢失，因此仍然能保证检测的准确度。

5.3　检测准确度影响

在这一节中，评价数据压缩对检测准确度的影响。因为智能攻击流量检测算法在压缩的数据集上训练，可能会存在精度的下降。表 3中显示，对于一般的情况，**Centroids** 方法对速度提升的负面作用是 1.22% 的准确度下降。对于不同的方法，实验发现本文提出的数据压缩对 N3IC 方法 [28] 的准确度下降最为明显，达到了 2.50% 的准确度下降，但是对于其他方法，准确度下降保持在 1.0% 之内。分析不同的数据集，可以发现对于加密攻击流量数据和 NetBeacon 数据集的准确度下降较为严重。同时还可以观察到，对训练数据进行压缩不绝对会导致准确度下降，在少数情况下甚至可以提升准确度，例如对于 nPrintML 方法 [34] 的准确度存在 0.76% 的微弱提升，此外在 IoT 数据集上的平均表现也有微弱的提升。这是因为数据压缩过程中一定程度地去除了数据中的噪声，这样有利于机器学习模型更好地学习数据。

另一个角度，图 7比较了压缩前后对准确度和召回率的影响。通过对全体方法比较可以看出，压缩前后对准确度-召回率曲线几乎没有影响。但综合大部分情况而言，**Centroids** 数据压缩方法在提升检测速度的同时会使召回率略微上升，而准确度略微下降，其原因是压缩减少了数据集中的噪声，使得检测更加鲁棒，从而倾向于判定流量为异常。综合以上所述，该方法不会对各个方案的准确度造成较大的影响，并且在此过程中保证了对训练速度的显著提升。

https://www.sciengine.com/doi/10.1360/SSI-2025-0025



(a) Accuracy comparison with other methods.　(b) Ratio comparison with other methods.

图 10　和其他压缩数据加速方案对比效果。

Figure 10　Comparing effectiveness with other compression methods.

5.4　检测鲁棒性的影响

我们根据现有研究中的逃避检测策略构建对抗流量样本 [18,26,55]：(i) 流量混淆：攻击者以 1:4 的比例注入良性 TCP/UDP 加密流量来混淆攻击流量；(ii) 发送速率降低：攻击者将其发送速率降低 50%；(iii) 长度操纵：攻击者通过操纵数据包长度来模仿良性加密流，参照随机选取的 5.0% 良性流量。根据这些策略，我们基于 16 个公开的流量数据集 [19] 生成了 48 种逃逸攻击。先前研究证实这些逃逸策略可以成果绕过若干现有检测系统 [18]。

图 8 分析了数据压缩对于检测鲁棒性产生的影响。可以发现，各个检测系统在检测具备逃逸行为的攻击时，相比在未压缩数据集上训练的系统，**Centroids** 数据集压缩使其降低了 0.1378 的 AUC，0.0401 的 Accuracy，0.0356 的 Precision，但可以提升 0.0139 的 Recall。这表明了在各种场景下，我们的方法对检测的鲁棒性几乎不存在影响，检测方案的鲁棒性不会显著下降导致攻击者可以逃避检测。另外，研究还发现操纵包长度对鲁棒性的影响更大，存在 0.0636 的准确度下降，显著高于流量混淆造成的 0.0368 的准确度下降，和降低速度逃逸造成的 0.0196 的准确度下降。

5.5　对比其他训练加速方法

流量特征是一种表格数据 [57] (Tabular Data)，现有的方案对于表格数据的压缩方法主要分为纵向和横向。纵向压缩方法减少数据集中的流量样本数量 [17,26]，例如对数据集进行采样。横向压缩方法通过降维，降低特征的维度，例如采用主成分分析 (PCA) 方法对特征维度进行降维 [20]。实验发现，这些方法均难以在保障准确的条件下有效降低流量数据集规模。仿照现有方法 [20,26]，本实验对比了基于采样的压缩方法，即对数据集随机采样 12.5%；以及基于降维的方法，即用 PCA 将数据集降维到 10 个维度。图 10 中比较了准确度和压缩比率。可以观察到，基于采样和降维的方法准确度损失比 **Centroids** 方案高 30.78 倍和 40.17 倍，但是压缩比比本方案低 8.27 倍和 8.87 倍。因此，对于流量数据 **Centroids** 方案可在保障训练准确性的情况下有效降低数据规模，效果优于传统横向压缩和纵向压缩方法。

5.6　算法的性能开销

最后讨论流量空间点云压缩算法自身的运行效率。请注意，这一阶段可以完全离线地完成，从而显著降低多个模型训练阶段的开销。图 9绘制了压缩各种智能攻击流量检测系统的特征的时间消耗分布。

实验选取了两个数据集，分别是规模最大的 NetBeacon 数据集和规模最小的 IoT 数据集，对其速度进行比较。通过对比可以发现，对于常见的流级别检测算法，压缩他们在 NetBeacon 数据集上的特征产生的时间消耗在 0.4675 秒至 2.050 秒之间。与此同时，对于常见的包粒度的检测算

法，压缩他们在 NetBeacon 训练数据集上提取的特征的时间开销在 33.97 秒至 71.71 秒之间。压缩包级别的特征消耗的时间显著长于压缩流级别的特征，这是因为包级别特征的方法对每一个数据包抽取一个特征，其产生的特征规模更大。虽然压缩特征产生的时间开销更大，但从图 4中可以看见，**Centroids** 方法可以为其节约更多的时间开销。另外，实验发现对于小规模的数据集，本文提出的方法均可在 2.33 秒内完成压缩任务。综上，**Centroids** 方法可以在较少时间完成，压缩后的数据集可以为现有的方法降低大量的的训练时间开销。

6　相关工作

攻击流量检测系统的目的在于将流量分类为正常和异常两大类。近年来，基于人工智能算法的攻击流量识别系统，相比于传统的基于固定规则匹配的检测系统，在实时性、准确度和吞吐量上均有显著提升[21~23]。这些系统借助人工智能算法，自动地分析数据包的特征[15,16]；同时也关注于分析粗粒度的流和主机的特征[17,18,20]。目前，这些系统可以检测很多复杂攻击产生的隐蔽恶意流量[16,19]。本研究旨在实现压缩流量数据集，加速这些算法的训练过程。在这节中我们将相关工作从若干角度进行讨论，包括检测到攻击后启用的攻击流量防御系统、智能驱动的安全研究中的数据集问题，以及基于机器学习的安全系统中的共性问题。

6.1　基于机器学习的恶意流量检测

为了实现通用的攻击流量检测，现有研究工作设计了许多特征：(i) 流级别特征，如离散分布特征[43]、统计特征[46] 和序列特征[55]；(ii) 包级别特征，如 Kitsune[15] 和 nPrintML[34]；(iii) 主机级别特征，如 HyperVision 的图特征[20] 和 Whisper 的频率特征[18]。与通用检测相比，特定任务的检测旨在识别恶意软件[10,17,62]、Web 攻击[32,41,58]、物联网网络中的威胁[59,60] 以及洪攻击[38,61]。本文聚焦于通用检测方法，并为其提升训练效率。

近年来，高速网络设备被用于高效检测。例如，基于可编程交换机的方法支持高速机器学习推断，如决策树[16,63]、随机森林[64] 和循环神经网络[65]。同样，具备多个嵌入 CPU 核心的 SmartNICs 使得部署更加灵活[28,66]。此外，现有方法还在 FPGA 上实现模型[35,67]。需要注意的是，恶意流量检测与流量分类是完全不同的任务，流量分类是判断流量是否由特定应用或用户生成[68~71]。

6.2　基于流量的攻击防御系统

这些系统通过使用固定规则来限制异常流量。早些年，基于 SDN 的方法实现了自动规则部署[44,72,73]。最近的研究在可编程交换设备上实现复杂的防御行为，例如基于寄存器的 Poseidon[29]、基于 Sketch 的 Jaqen[30]、使用多个设备的 Mew[74]、分布式防御的 Ripple[45]、针对脉冲 DoS 攻击的 ACC-Turbo[75] 以及针对隐蔽通道的 NetWarden[31]。此外，研究人员基于传统的转发设备设计了诸多防御方案，例如针对 ISP 的 AS 层次防御[76] 和 IXP 视角下的防御[22,77]。

6.3　安全研究中的数据集问题的

为了消除模拟和现实世界数据集之间的差异，部分研究工作通过人工设计规则来增强 Tor 流量[70] 用于构造攻击和用于应用识别任务[78]。同样，Jin *et al.* [69] 人工构造了多标签 Web 流量数据集。Jan *et al.* [79] 使用 GANs[80] 生成了用于假用户检测的攻击样本。此外，netUnicorn 是一

个用于收集流量数据集的工具[81]，Qing *et al.* [55] 解决了标签错误问题，而 Du *et al.* [82] 实现了高效的重训练。但目前，对于安全智能应用的数据集规模约束研究仍然存在显著空白。

6.4　基于机器学习的安全应用中的问题

Sommer *et al.* [37] 分析了基于机器学习的流量检测系统的低可用性问题，并强调了构建新数据集的高昂成本。Arp *et al.* [36] 研究了将机器学习应用于安全将遇到的实际操作问题。此外，Alahmadi *et al.* [83]、Vermeer *et al.* [23] 发现基于机器学习的安全应用产生了大量误报。Fu *et al.* [42] 等人提出一种假阳性警报识别方案，在智能算法的测试阶段将警报视为点云，并识别关键点；而本研究是一种流量数据集压缩方案，在智能算法的训练阶段将流量视为点云，并压缩点云。因此，两项工作解决不同的问题、部署在不同的阶段、点云建模的对象不同并且算法处理的流程不同。还有研究分析了概念漂移问题[84]。此外，Han *et al.* [85]、Jacobs *et al.* [86] 和 Wei *et al.* [21] 尝试解机器学习安全应用的可解释性问题。本研究认为，**Centroids** 尝试解决的数据集规模问题是部署机器学习安全应用的关键挑战之一。

6.5　其他模型训练效率提升方法

目前有多种方法提升模型训练的效率，其主要从计算和通讯角度优化训练过程。首先，基于计算的方案引入专用计算硬件，例如 GPU[42,87,88] 和 FPGA[35,67]，加速训练过程中的向量和矩阵运算。这些计算加速手段也被用于流量分析任务，例如 Zhao *et al.* [89] 采用 GPU 训练基于 Transformer 流量分析模型，ET-BERT 方案[88] 和 YaTC 方案[87] 也采用了 GPU 训练用于流量分类的大规模注意力模型。类似地，现有若干网站指纹生成方案也采用了 GPU 训练加速，例如 Deng *et al.* [71] 的方案，和 TMWF 方案[69]。另一方面，现有方案从网络通讯的角度优化训练过程，例如 ATP 方案[90] 和 SwitchML 方案[91] 可在编程网络设备上进行梯度聚合，类似的 ByteScheduler 方案[92] 优化了深度学习框架的网络通讯模块。这两大类方法引入专用硬件设备提升训练效果，而 **Centroids** 方案是数据驱动的训练效率优化方法，在不引入新硬件的条件下提升模型训练效率，这一流量数据压缩方案可与基于计算和通讯的训练效率提升方法同时被使用。

7　研究局限性和研究展望

首先，**Centroids** 对检测系统的鲁棒性的影响有待进一步分析。在本项研究中，仅考虑了现有的、已经被披露出来的逃逸攻击[18,20,93,94]。然而，我们无法预料其他潜在的针对基于机器学习的流量检测系统的逃逸攻击，因此我们仅采用了现有的逃逸策略，以验证攻击者在 **Centroids** 启用时是否能逃逸检测系统。在未来，当其他逃逸攻击被发现后，我们方案对鲁棒性的影响需要进行更为细致的评估。

此外，目前我们的流量数据集压缩方法仅能压缩表格数据形式的向量特征。但是，少数方法采用矩阵形式的特征[18]，甚至是异构的图特征[19]，对于这些方法的流量压缩策略仍有待进一步探究。现有的测量实验发现，这些数据在其对应的流量空间中和向量形式的特征类似，均存在稠密分布的特性，这意味着其数据可能存在一定程度的冗余性。在未来的研究工作中，可能通过改造输入形式，例如矩阵展开，使得我们的方案适配到这些不采用向量形式输入的智能攻击流量检测系统。

https://www.sciengine.com/doi/10.1360/SSI-2025-0025

8　结论

本研究工作提出了一种智能攻击流量检测模型的训练加速方法，旨在通过压缩流量数据集，利用有限的计算资源在大规模流量数据集上快速训练攻击流量检测模型。该方法使得各种检测模型可以迅速部署于网络环境中。方案采用的数据压缩技术基于高维流量特征空间的几何结构。具体方法包括将数据集中的流量样本映射为高维特征空间中的点云，并将体素概念扩展到此高维空间来划分点云。通过计算体素中点的形心，以此来代表整个区域中的点云，从而显著减小数据集的规模。实验结果验证了该方法在八个不同的数据集上对当前最先进的十种检测技术可以提升达 75.82 倍的训练效率，同时基本不影响检测的准确度和鲁棒性。

参考文献

1 WIDE MAWI Working Group Traffic Archive. (http://mawi.wide. ad.jp/mawi/)
2 Luo, X. & Chang, R. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. *ISOC NDSS*. (2005)
3 Jonker, M., et al. Measuring the Adoption of DDoS Protection Services. *ACM IMC*. pp. 279-285 (2016)
4 Jonker, M., et al. A First Joint Look at DoS Attacks and BGP Blackholing in the Wild. *ACM IMC*. pp. 457-463 (2018)
5 Moura, G., et al. When the Dike Breaks: Dissecting DNS Defenses During DDoS. *ACM IMC*. pp. 8-21 (2018)
6 Griffioen, H., et al. Scan, Test, Execute: Adversarial Tactics in Amplification DDoS Attacks. *ACM CCS*. pp. 940-954 (2021)
7 Kopp, D., et al. DDoS Hide Seek: On the Effectiveness of a Booter Services Takedown. *ACM IMC*. pp. 65-72 (2019)
8 Durumeric, Z., et al. An Internet-Wide View of Internet-Wide Scanning. *USENIX Security*. pp. 65-78 (2014)
9 Merget, R., et al. Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities. *USENIX Security*. pp. 1029-1046 (2019)
10 Dodia, P., et al. Exposing the Rat in the Tunnel: Using Traffic Analysis for Tor-based Malware Detection. *ACM CCS*. pp. 875-889 (2022)
11 Chen, Y., et al. On Training Robust PDF Malware Classifiers. *USENIX Security*. pp. 2343-2360 (2020)
12 AKamai Prolexic. (https://www.akamai.com/products/prolexic-solutions)
13 AKamai Akamai Unveils Machine Learning That Intelligently Automates Application And API Protections And Reduces Burden On Security Professionals. (https://www.akamai.com/newsroom/press-release/-akamai-unveils-machine-learning-that-intelligently-automates-ap)
14 Cisco Encrypted Traffic Analytics. (https://www.cisco.com/c/en/us/solutionsenterprise-networks/enterprise-network-security/eta.html)
15 Mirsky, Y., et al. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *ISOC NDSS*. (2018)
16 Zhou, G., et al. NetBeacon: An Efficient Design of Intelligent Network Data Plane. *USENIX Security*. pp. 6203 - 6220 (2023)
17 Bilge, L., et al. Disclosure: detecting botnet command and control servers through large-scale NetFlow analysis. *ACM ACSAC*. pp. 129-138 (2012)
18 Fu, C., et al. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. *ACM CCS*. pp. 3431-3446 (2021)
19 Fu, C., et al. Detecting Unknown Encrypted Malicious Traffic in Real Time via Flow Interaction Graph Analysis. *ISOC NDSS*. (2023)
20 Fu, C., et al. Flow Interaction Graph Analysis: Unknown Encrypted Malicious Traffic Detection. *IEEE/ACM Trans. Netw.*. **33**, 2972-2987 (2024)
21 Wei, F., et al. XNIDS: Explaining Deep Learning-based Network Intrusion Detection Systems for Active Intrusion Responses. *USENIX Security*. pp. 4337-4354 (2023)
22 Wichtlhuber, M., et al. IXP scrubber: learning from blackholing traffic for ML-driven DDoS detection at scale. *SIGCOMM*. pp. 707-722 (2022)
23 Vermeer, M., et al. Alert Alchemy: SOC Workflows and Decisions in the Management of NIDS Rules. *ACM CCS*. pp. 2770-2784 (2023)
24 CIC Intrusion Detection Evaluation Datasets (CIC-IDS2017). (https://www.unb.ca/cic/datasets/ids-2017.html)
25 CIC DDoS Evaluation Datasets (CIC-DDoS2019). (https:// www.unb.ca/cic/datasets/ddos-2019.html)
26 Fu, C., et al. Frequency Domain Feature Based Robust Malicious Traffic Detection. *IEEE/ACM Trans. Netw.*. **31**, 452-467 (2023)
27 King, I. & Huang, H. Euler: Detecting Network Lateral Movement via Scalable Temporal Graph Link Prediction. *ISOC NDSS*. (2022)
28 Siracusano, G., et al. Re-architecting Traffic Analysis with Neural Network Interface Cards. *NSDI*. pp. 513-533 (2022)
29 Zhang, M., et al. Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches. *ISOC NDSS*. (2020)
30 Liu, Z., et al. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. *USENIX Security*. pp. 3829-3846 (2021)
31 Xing, J., et al. NetWarden: Mitigating Network Covert Channels while Preserving Performance. *USENIX Security*. pp. 2039-2056 (2020)
32 Tang, R., et al. ZeroWall: Detecting Zero-Day Web Attacks through Encoder-Decoder Recurrent Neural Networks. *INFOCOM*. pp. 2479-2488 (2020)
33 Du, M., et al. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. *ACM CCS*. pp. 1285-1298 (2017)
34 Holland, J., et al. New Directions in Automated Traffic Analysis. *ACM CCS*. pp. 3366-3383 (2021)
35 Swamy, T., et al. Taurus: a data plane architecture for per-packet ML. *ASPLOS*. pp. 1099-1114 (2022)
36 Arp, D., et al. Dos and Don'ts of Machine Learning in Computer Security. *USENIX Security*. (2022)
37 Sommer, R. & Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE*

SP. pp. 305-316 (2010)
38 Xu, Z., et al. Xatu: boosting existing DDoS detection systems using auxiliary signals. *IEEE CoNEXT*. pp. 1-17 (2022)
39 Gobbetti, E. & Marton, F. Far voxels: a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. *ACM Trans. Graph.*. **24**, 878-885 (2005)
40 Liu, L., et al. Neural Sparse Voxel Fields. *NIPS*. (2020)
41 Bartos, K., et al. Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants. *USENIX Security*. pp. 807-822 (2016)
42 Fu, C., et al. Point Cloud Analysis for ML-Based Malicious Traffic Detection: Reducing Majorities of False Positive Alarms. *ACM CCS*. pp. 1005-1019 (2023)
43 Barradas, D., et al. FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications. *ISOC NDSS*. (2021)
44 Zheng, J., et al. Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Trans. Inf. Forensics Secur.*. **13**, 1838-1853 (2018)
45 Xing, J., et al. Ripple: A Programmable, Decentralized Link-Flooding Defense Against Adaptive Adversaries. *USENIX Security*. pp. 3865-3880 (2021)
46 CIC CICFlowMeter: a network traffic flow generator and analyser. (https://www.unb.ca/cic/research/applications.html)
47 Wikipedia Centroid. (https://en.wikipedia.org/wiki/Centroid)
48 University, C. The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background traffic. (https://www.stratosphereips.org/datasets-ctu13)
49 Canadian Institute for Cybersecurity. DNS-over-HTTP datasets. (https://www.unb.ca/cic/datasets/dohbrw-2020.html)
50 Canadian Institute for Cybersecurity. A real-time dataset and benchmark for large-scale attacks in IoT environment. (https://www.unb.ca/cic/datasets/iotdataset-2023.html)
51 Canadian Institute for Cybersecurity. Android malware dataset. (https://www.unb.ca/cic/datasets/andmal2017.html)
52 Anderson, B. & McGrew, D. Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity. *KDD*. pp. 1723-1732 (2017)
53 Kang, M., et al. The Crossfire Attack. *IEEE SP*. pp. 127-141 (2013)
54 Kuzmanovic, A. & Knightly, E. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. *SIGCOMM*. pp. 75-86 (2003)
55 Qing, Y., et al. Low-Quality Training Data Only? A Robust Framework for Detecting Encrypted Malicious Network Traffic. *ISOC NDSS*. (2024)
56 Rossow, C. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. *ISOC NDSS*. (2014)
57 Wang, K., et al. BARS: Local Robustness Certification for Deep Learning based Traffic Analysis Systems. *ISOC NDSS*. (2023)
58 Li, P., et al. Learning from Limited Heterogeneous Training Data: Meta-Learning for Unsupervised Zero-Day Web Attack Detection across Web Domains. *ACM CCS*. pp. 1020-1034 (2023)
59 Sharma, R., et al. Lumen: a framework for developing and evaluating ML-based IoT network anomaly detection. *CoNEXT*. pp. 59-71 (2022)
60 Tekiner, E., et al. A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks. *ISOC NDSS*. (2022)
61 Samra, R. & Barcellos, M. DDoS2Vec: Flow-Level Characterisation of Volumetric DDoS Attacks at Scale. *PACMNET*. **1**, 13:1-13:25 (2023)
62 Tegeler, F., et al. BotFinder: finding bots in network traffic without deep packet inspection. *CoNEXT*. pp. 349-360 (2012)
63 Jafri, S., et al. Leo: Online ML-based Traffic Classification at Multi-Terabit Line Rate. *NSDI*. (2024)
64 Dong, Y., et al. HorusEye: A Realtime IoT Malicious Traffic Detection Framework using Programmable Switches. *USENIX Security*. pp. 571-588 (2023)
65 Yan, J., et al. Brain-on-Switch: Towards Advanced Intelligent Network Data Plane via NN-Driven Traffic Analysis at Line-Speed. *NSDI*. pp. 419-440 (2024)
66 Panda, S., et al. SmartWatch: accurate traffic analysis and flow-state tracking for intrusion prevention using SmartNICs. *CoNEXT*. pp. 60-75 (2021)
67 Fu, C., et al. Detecting Tunneled Flooding Traffic via Deep Semantic Analysisof Packet Length Patterns. *ACM CCS*. pp. 3659 - 3673 (2024)
68 Shen, M., et al. Subverting Website Fingerprinting Defenses with Robust Traffic Representation. *USENIX Security*. (2023)
69 Jin, Z., et al. Transformer-based Model for Multi-tab Website Fingerprinting Attack. *ACM CCS*. pp. 1050-1064 (2023)
70 Bahramali, A., et al. Realistic Website Fingerprinting By Augmenting Network Traces. *ACM CCS*. pp. 1035-1049 (2023)
71 Deng, X., et al. Robust Multi-tab Website Fingerprinting Attacks in the Wild. *IEEE SP*. pp. 1005-1022 (2023)
72 Fayaz, S., et al. Bohatei: Flexible and Elastic DDoS Defense. *USENIX Security*. pp. 817-832 (2015)

73 Kang, M., et al. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks. *ISOC NDSS*. (2016)

74 Zhou, H., et al. Mew: Enabling Large-Scale and Dynamic Link-Flooding Defenses on Programmable Switches. *IEEE SP*. pp. 3178-3192 (2023)

75 Alcoz, A., et al. Aggregate-based congestion control for pulse-wave DDoS defense. *ACM SIGCOMM*. pp. 693-706 (2022)

76 Tran, M., et al. On the Feasibility of Rerouting-Based DDoS Defenses. *IEEE SP*. pp. 1169-1184 (2019)

77 Wagner, D., et al. United We Stand: Collaborative Detection and Mitigation of Amplification DDoS Attacks at Scale. *ACM CCS*. pp. 970-987 (2021)

78 Xie, R., et al. Rosetta: Enabling Robust TLS Encrypted Traffic Classification in Diverse Network Environments with TCP-Aware Traffic Augmentation. *USENIX Security*. pp. 625-642 (2023)

79 Jan, S., et al. Throwing Darts in the Dark? Detecting Bots with Limited Data using Neural Data Augmentation. *IEEE SP*. pp. 1190-1206 (2020)

80 Yin, Y., et al. Practical GAN-based synthetic IP header trace generation using NetShare. *ACM SIGCOMM*. pp. 458-472 (2022)

81 Beltiukov, R., et al. In Search of netUnicorn: A Data-Collection Platform to Develop Generalizable ML Models for Network Security Problems. *ACM CCS*. pp. 2217-2231 (2023)

82 Du, M., et al. Lifelong Anomaly Detection Through Unlearning. *ACM CCS*. pp. 1283-1297 (2019)

83 Alahmadi, B., et al. 99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms. *USENIX Security*. pp. 2783-2800 (2022)

84 Yang, L., et al. CADE: Detecting and Explaining Concept Drift Samples for Security Applications. *USENIX Security*. pp. 2327-2344 (2021)

85 Han, D., et al. DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications. *ACM CCS*. pp. 3197-3217 (2021)

86 Jacobs, A., et al. AI/ML for Network Security: The Emperor has no Clothes. *ACM CCS*. pp. 1537-1551 (2022)

87 Zhao, R., et al. Yet Another Traffic Classifier: A Masked Autoencoder Based Traffic Transformer with Multi-Level Flow Representation. *AAAI*. pp. 5420-5427 (2023)

88 Lin, X., et al. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. *WWW*. pp. 633-642 (2022)

89 Zhao, R., et al. MT-FlowFormer: A Semi-Supervised Flow Transformer for Encrypted Traffic Classification. *KDD*. pp. 2576-2584 (2022)

90 Lao, C., et al. ATP: In-network Aggregation for Multi-tenant Learning. *NSDI*. pp. 741-761 (2021)

91 Sapio, A., et al. Scaling Distributed Machine Learning with In-Network Aggregation. *NSDI*. pp. 785-808 (2021)

92 Peng, Y., et al. A generic communication scheduler for distributed DNN training acceleration. *SOSP*. pp. 16-29 (2019)

93 Xu, S., et al. "One Model Fits All Nodes": Neuron Activation Pattern Analysis-Based Attack Traffic Detection Framework for P2P Networks. *IEEE/ACM Trans. Netw.*. pp. to appear (2025)

94 Gao, L., et al. Wedjat: Detecting Sophisticated Evasion Attacks via Real-time Causal Analysis. *KDD*. pp. to appear (2025)

Accelerate Training of AI-Powered Attack Traffic Detection: Dataset Compression Based on Centroids of Point Clouds in Traffic Feature Spaces

Chuanpu Fu[1], Qi Li[2,3] & Ke Xu[1,3*]

1. *Tsinghua University, Department of Computer Science and Technology, Beijing* 100084, *China*;
2. *Tsinghua University, Institute of Network Sciences and Cyberspace, Beijing* 100084, *China*;
3. *Zhongguancun Lab, Beijing* 100194, *China*
* Corresponding author. E-mail: xuke@tsinghua.edu

Abstract Attack traffic detection systems identify malicious traffic within vast amounts of internet traffic to protect legitimate users. In recent years, artificial intelligence algorithms have been widely applied in the detection of attack traffic. These algorithms learn traffic patterns from traffic datasets and are capable of detecting stealthy threats with significantly higher accuracy compared to traditional rule-based detection methods. However, with the rapid increase in internet traffic, the scale of traffic datasets has also grown significantly, meaning that training models for attack traffic detection on large-scale traffic datasets consumes substantial computational power and involves considerable deployment delays.

In this paper, we propose a method to accelerate the training of attack traffic detection models by compressing traffic data, allowing for rapid training on large-scale datasets using limited computational resources and enabling swift deployment of the detection system to networks. The specific data compression method is based on the geometric structure of the high-dimensional traffic feature space. We map the entire set of sample traffic to a point cloud in the high-dimensional traffic feature space, and extend the concept of voxels to this space. By calculating the centroids of points within these voxels, we represent the entire set of points (traffic samples), thus significantly reducing the size of the dataset. Experimental analysis shows that this method can improve training efficiency by 75.82 times across eight datasets for the ten state-of-the-art detection methods currently available, without significantly affecting the accuracy and robustness of detection, and ensuring real-time processing of compressed traffic data.

Keywords Network Security, Artificial Intelligence, Attack Traffic Detection, Point Cloud, Voxel