基于字节序列频域采样的恶意软件分类

蒋永康，孙　逊，杨玉龙
（贵州航天计量测试技术研究所，贵州　贵阳　550009）

摘　要：近年来，利用机器学习直接从文件字节序列中提取特征并进行恶意软件分类的方法受到了广泛关注。但恶意软件字节序列较长，直接输入模型进行训练，时间和空间开销巨大，难以适用大数据场景下的海量文件样本。针对该问题，提出基于字节序列频域采样的恶意软件分类方法，通过离散傅里叶变换设计频域采样策略保留字节序列中的主要低频成分，合成新的短信号，实现训练效率的提高。公开数据集上的实验结果表明，与最先进的基于原始字节序列的恶意软件分类方法相比，所提出的方法与其分类效果相当，且将模型的训练时间和 GPU 显存占用分别降低了 90% 和 50% 以上。

关键词：恶意软件分类；字节序列；频域采样；机器学习
中图分类号：TP393.08　　**文献标识码**：A　　**DOI**：10.19358/j.issn.2097-1788.2025.01.003
引用格式：蒋永康，孙逊，杨玉龙. 基于字节序列频域采样的恶意软件分类［J］. 网络安全与数据治理，2025，44(1)：15-20.

Frequency domain sampling of byte sequences for malware classification

Jiang Yongkang，Sun Xun，Yang YuLong
（Institute of Guizhou Aerospace Measuring and Testing Technology，Guiyang 550009，China）

Abstract：Recently，methods of using machine learning to directly extract features from byte sequences and classify malware have received widespread attention. However，byte sequences of malware are long，directly inputting them into models for training will involve large time and space overheads，making it difficult to adapt to massive samples in big data scenarios. To address this problem，this paper proposes a malware classification method based on frequency domain sampling of byte sequences. A frequency domain sampling strategy is designed through discrete Fourier transform to retain main low-frequency components in byte sequence，synthesize new short signals，and achieve the purpose of improving training efficiency. Experimental results show that compared with the state-of-the-art malware classification method based on raw byte sequences，the proposed method has comparable accuracy and can reduce the model training time and GPU memory usage by more than 90% and 50% respectively.

Key words：malware classification；byte sequences；frequency domain sampling；machine learning

0　引言

恶意软件分类致力于研究如何识别恶意软件以及区分不同的恶意软件家族，作为网络安全研究领域中的一个重要分支，对于理解和防御不同类型的恶意软件以及溯源网络攻击具有重要意义。

恶意软件分类方法大致可以分为：基于静态特征[1]和动态特征[2]的传统方法，以及引入机器学习[3]的新式方法。基于静态特征的方法[4-6]依赖于复杂的特征工程，难以应对恶意软件的快速演化；基于动态特征的方法[7-9]涉及耗时的行为特征监控[10]，难以规模扩展。

近年来，利用机器学习直接从文件字节序列中提取特征并进行恶意软件分类的方法受到了广泛关注[11-12]。该方法的框架如图 1 所示，其研究核心是设计一个分类模型，将输入样本 x 的字节序列映射到一个范围为［0，1］的概率分布 $c=[c_0, c_1, \cdots, c_M]$ 上，其中 $\sum c_m=1$。测试时，计算类别 $m=\text{argmax}(c)$，$m=0$ 表示良性软件，$m\geqslant 1$ 表示相应的恶意软件家族。如果 $M=1$，分类模型实现面向良性软件与恶意软件的二分类；如果 $M\geqslant 2$，则分类模型实现面向恶意软件家族的多分类，此时良性软件被看作一类特殊的家族。

通过机器学习模型自动地从序列中提取和编码特征的技术路线能更好地适应当今恶意软件的动态变化，也

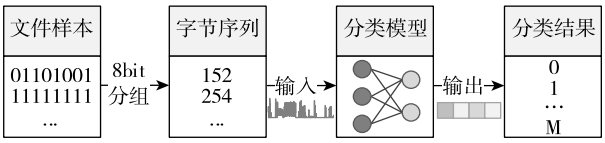

图 1　基于字节序列分析的恶意软件分类框架

能更好地实现各类型和跨平台的恶意软件分类。然而，当面对大数据场景下的海量文件样本时，该路线面临因恶意软件的字节序列较长，直接输入模型进行训练，导致时间和空间开销巨大的挑战。从形式上看，字节序列分类任务类似于时间序列分类任务，借鉴其研究成果已被证明极具挑战[12]。恶意软件通常包含数十万到数百万长度的字节序列，以 100 万长度的字节序列为例，其相当于 $T=1\ 000\ 000$ 步长的时间序列，而已知的最长时间序列分类任务步长 $\leqslant 16\ 000$[13]，这限制了现有时间序列分类模型的直接应用。目前为止，能处理这种极端长度字节序列分类任务的实现是 MalConv[11]，其通过简洁的模型设计，可以处理 $T=2\ 000\ 000$ 步长的字节序列。遗憾的是，MalConv 的训练开销极大，例如在 Ember[14] 数据集 60 万样本上训练该模型，128 GB 显存的 DGX-1 需要消耗一个月的时间。尽管 MalConv2[12] 通过优化池化降低了训练的显存开销，但训练的时间开销依然很大。

本文针对上述如何提高字节序列分类模型的训练效率展开研究。通过引入离散傅里叶变换[15]分析文件字节序列的频率分量发现，字节序列中的能量主要集中在低频部分。本文通过截取低频分量来缩短输入字节序列的长度，进而提出基于字节序列频域采样的恶意软件分类方法。核心的设计思路为：设计频域采样策略，保留字节序列中的主要低频分量，合成新的短信号，从而实现训练效率的提高。Windows 和 Android 公开恶意软件数据集上的实验结果表明，与最先进的基于原始字节序列的 MalConv2[12] 相比，本文提出的方法与其分类效果相当，且将模型的训练时间和 GPU 显存占用分别降低了 90% 和 50% 以上。

综上，本文的主要贡献如下：

（1）提出了一种基于字节序列频域采样的恶意软件分类方法，通过设计频域采样策略，减小输入字节序列的长度，实现模型训练效率的提高。

（2）在公开数据集上进行了验证，结果表明，提出的恶意软件分类方法与最先进的基于原始字节序列的方法分类效果相当，且能将模型的训练时间和 GPU 显存占用大幅降低。

（3）分析了字节序列频域采样策略中采样长度的影响，并对未来的研究方向进行了讨论。

1　设计思路

恶意软件，无论其文件格式如何，都以二进制的形式存储在计算机当中。每个二进制值（0 或 1）用一个比特（bit）表示，8 个比特构成一个字节（byte），每个字节对应［0，255］的数值范围。所以，恶意软件的字节序列可以表示为

$$x=[x(0),x(1),\cdots,x(N-1)],\ x(n)\in[0,255]. \tag{1}$$

其中，$x(n)$ 是文件第 n 个位置上的十进制值，N 是文件的长度。如果将 $x(n)$ 看成文件在第 n 个时刻的响应值，字节序列形式上即为一维的时间序列。

图 2 给出了 zbot 和 wannacry 两个恶意软件家族样本的原始字节序列示例（前 65 536 个点）。可以看出，同一家族样本的字节序列分布相似，不同家族样本的字节序列分布不同。这是基于字节序列分析的恶意软件分类方法的基础。

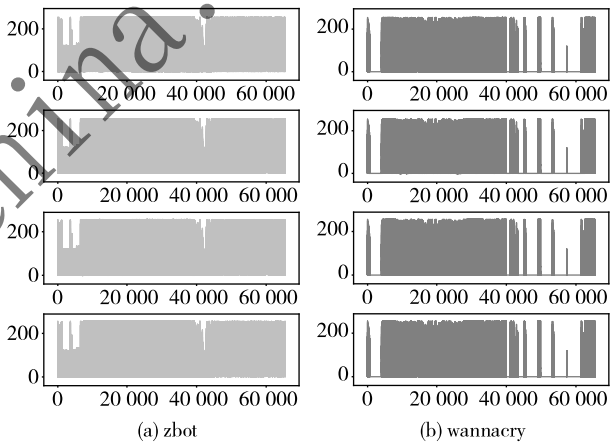

图 2　恶意软件家族样本原始字节序列示例

如何提高训练效率，一直是困扰研究人员的问题。如图 3 所示，通过离散傅里叶变换提取上述样本原始字节序列中的频域分量时发现，原始字节序列中的能量主

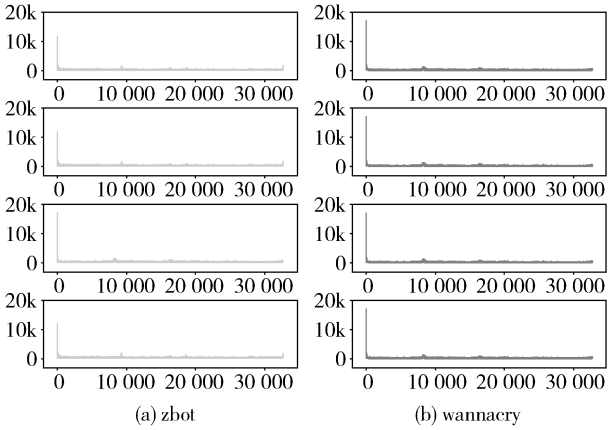

图 3　恶意软件家族样本原始字节序列频谱示例

要集中在低频部分，低频分量包含了样本的主要信息。本文通过截取低频分量合成新的采样序列，来缩短输入字节序列的长度。

图 4 给出了 zbot 和 wannacry 样本原始字节序列的频域采样结果，采样长度设定为 2 048。可以发现，同一家族样本的采样字节序列分布极其相似，不同家族样本的采样字节序列分布明显不同。

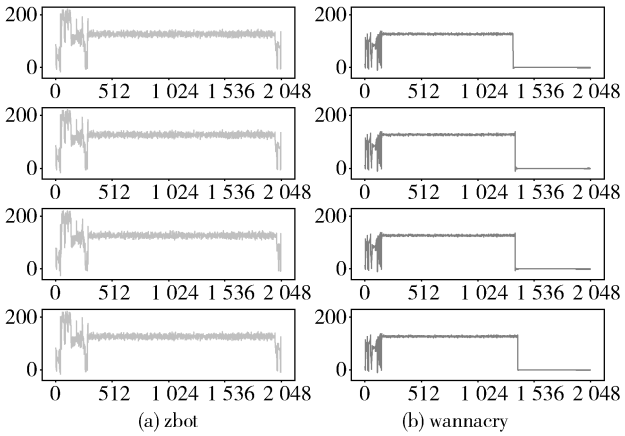

图 4 恶意软件家族样本采样字节序列示例

鉴于上述分析，本文字节序列频域采样的核心设计思路为：设计频域采样策略，保留字节序列中的主要低频分量，合成新的短信号，以实现提高训练效率的目的。

2 方法描述

本文提出基于字节序列频域采样的恶意软件分类方法。该方法包括两部分：字节序列频域采样和字节序列分类，如图 5 所示。字节序列频域采样部分实现将原始字节序列采样为短信号，字节序列分类部分利用机器学习模型对采样后的序列进行分类，输出分类结果。

鉴于字节序列中的低频分量包含了样本的主要能量，如图 5 所示，字节序列频域采样的主要步骤为：（1）利用实数离散傅里叶正变换（Real Fast Fourier Fouvier Transform，rFFT）计算字节序列的复频域分量；（2）根据采样长度截断高频分量，保留主要低频分量；（3）利用低频分量和实数离散傅里叶逆变换（Inverse Real Fast Transform，irFFT）合成新的采样信号。具体地，根据离散傅里叶正变换，字节序列 $x\ (n)$ 的复频域分量 $X\ (k)$ 的计算公式为：

$$X\ (k)=\sum_{n=0}^{N-1}x\ (n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn} \tag{2}$$

由于实数序列的 FFT 具有共轭对称性，即 $X\ (k)=X^*\ (N-k)$，实际上只需要计算前 $N/2+1$ 个正频率分量。考虑设置的采样长度为 L，则需要保留的正频率分量集为 $\{X\ (0), X\ (1), \cdots, X\ (L'-1)\}$，其中 $L'=L/2+1$。利用共轭对称性，可以补充保留的正频率分量的负频率部分，得到相应的正负频率分量集 X'。最后通过如下离散傅里叶逆变换，来合成时域的采样信号 $\hat{x}$，即：

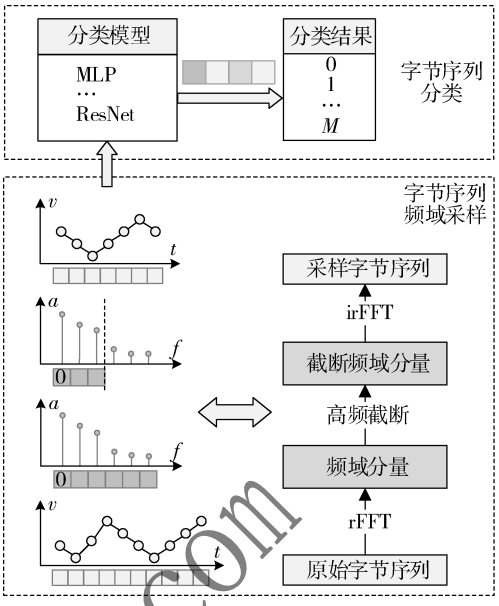

图 5 基于字节序列频域采样的恶意软件分类方法

$$\hat{x}\ (n)=\frac{1}{L}\sum_{k=0}^{L-1}X'(k)\mathrm{e}^{\mathrm{j}\frac{2\pi}{L}kn} \tag{3}$$

上述频域采样的伪代码描述如算法 1 所示。

算法 1 字节序列频域采样算法

输入：原始字节序列 $x=[x(0), x(1), \cdots, x(N-1)]$，采样长度 L

输出：采样字节序列 $\hat{x}=[\hat{x}\ (0), \hat{x}(1), \cdots, \hat{x}(L-1)]$

// 快速傅里叶正变换

$N'=N/2+1$

$X=\{X\ (0), X\ (1), \cdots, X\ (N'-1)\}=\mathrm{rFFT}\ [x]$

// 高频截断

$L'=L/2+1$

$X'=\{X\ (0), X\ (1), \cdots, X\ (L'-1)\}$

// 共轭对称补充负频率

$X'=\{X', X^*\ (0), \cdots, X^*\ (L'-1)\}$

// 快速傅里叶逆变换

$\hat{x}=\{\hat{x}\ (0), \hat{x}\ (1), \cdots, \hat{x}\ (L-1)\}=\mathrm{irFFT}\ [X']$

return $\hat{x}$

图 6 给出了原始字节序列在不同采样长度下的示例。可以发现，在 1 024、2 048 和 4 096 三个不同的采样长度下，采样字节序列的整体分布极其相似；差异在于，随着采样长度的增加（包含的高频分量增多），合成字节序

列的细节更丰富。这个示例直观地说明了，原始字节序列中的高频分量对于其整体分布影响不大，字节序列的主要信息集中在低频分量中。

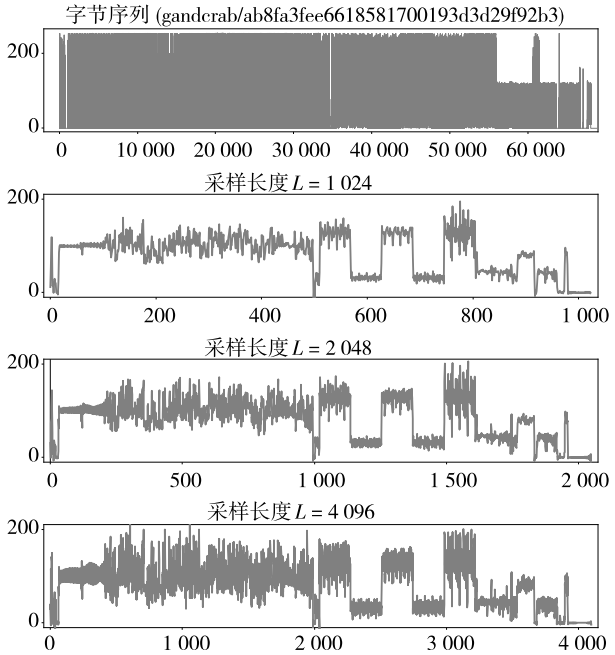

图 6　不同采样长度下的采样字节序列示例

经过上述采样过程，数十万到数百万长度的原始字节序列被浓缩为数百到数万长度的采样字节序列。如果基于采样字节序列能训练有效的恶意软件分类模型，那么这种技术路线将能更好地适应大数据场景下海量文件样本的分析，也能为今后借鉴时间序列分类领域的最新研究成果铺平道路。

3　实验分析

本文的实验分析需要解答如下问题：（1）利用采样后的字节序列能否训练有效的恶意软件分类模型；（2）采样长度对分类效果有何影响。

3.1　实验设置

3.1.1　数据集

实验采用三个公开的恶意软件数据集：Drebin[16]、Ember[14]和 BenchMFC[17]。同时在 Windows 平台和 Android 平台上验证提出的字节序列频域采样方法的有效性。Drebin 是 Android 平台的经典恶意软件家族数据集，包含 179 个家族的 5 560 个恶意软件样本，由于绝大多数家族只包含≤100 样本，选择 Drebin 中的前 8 个家族，共计 3 317 个样本作为 Android 平台的实验数据，如表 1 所示。

Ember 和 BenchMFC 是 Windows 平台的常用恶意软件数据集。Ember 中的恶意软件并没有标注具体的恶意软件家族；BenchMFC 只包含恶意软件，每个样本都有明确的家族标签。为此，本文从 Ember 中随机选择 4 000 个良性样本（Benign），再从 BenchMFC 中随机选择 8 个恶意软件家族，每个家族 500 个样本，共计 8 000 个样本，作为 Windows 平台的实验数据，如表 2 所示。

表 1　Android 恶意软件分类实验数据

ID	家族名称	样本数量
1	FakeInstaller	925
2	DroidKungFu	667
3	Plankton	625
4	GingerMaster	339
5	BaseBridge	330
6	Iconosys	152
7	Kmin	147
8	FakeDoc	132
	总计	3 317

表 2　Windows 恶意软件分类实验数据

ID	家族名称	样本数量
0	Benign	4 000
1	Fareit	500
2	Gandcrab	500
3	Hotbar	500
4	Parite	500
5	Simda	500
6	Upatre	500
7	Yuner	500
8	Zbot	500
	总计	8 000

3.1.2　实验环境

实验在一台配备 NVIDIA RTX 2080Ti GPU，显存为 12 GB 的服务器上进行。服务器的 CPU 为 AMD Ryzen 78845H，内存为 32 GB，操作系统为 Ubuntu 20.04。所有实验均使用 PyTorch 2.0 深度学习框架进行实现。

3.1.3　分类模型

本文的实验目的是验证基于采样字节序列训练恶意软件分类模型的有效性。为此，原始字节序列上的分类模型考虑最先进的面向原始字节序列的 MalConv[11]和 MalConv2[12]，采样字节序列上的分类模型考虑简单的 ResNet18[18]。上述模型均按原论文进行参数配置。

3.1.4　评估指标

准确率是指模型正确分类的样本数占总样本数的比

例。对于模型的分类效果，实验选择准确率和受试者工作特征曲线下面积（Area Under the Receiver Operating Characteristic Curve，AUROC）作为评估指标。AUROC 通过绘制不同阈值下的假阳性率和真阳性率来评估模型的分类能力，AUROC 值介于 0 到 1 之间，值越高表示分类效果越好。

实验样本被按照 6 ∶ 2 ∶ 2 的比例划分为训练集、验证集和测试集。训练 MalConv 和 MalConv2 模型时，样本字节序列被统一截断或填充到 1 048 576。实验过程中，设定训练 batch 大小为 32，每个分类模型在训练集上训练 50 个 epoch，记录模型的训练时间开销和 GPU 显存占用，基于验证集选择最优模型，在测试集上进行分类效果评估。

3.1.5 超参数

采样长度 L 对模型分类效果的影响会在第二个实验问题中详细讨论。在第一个实验问题的评估过程中将使用 $L=1\ 024$，因为实验结果表明，模型在这个采样长度下的分类准确率并不理想。

3.2 分类性能评估

基于第一个实验问题，采样字节序列能否训练有效的恶意软件分类模型。实验分别针对 Windows 和 Android 两个平台的恶意软件分类任务展开讨论。Windows 平台的实验结果如表 3 所示。分析可以发现，基于采样字节序列的 ResNet18 实现了与基于原始字节序列的 MalConv 和 MalConv2 相当的分类效果，且在训练时间开销和 GPU 显存占用上优势明显。具体来说，在训练时间开销和 GPU 显存占用上，ResNet18（采样字节序列）相较于 MalConv（原始字节序列）分别降低了 94.13% 和 88.01%，相较于 MalConv2（原始字节序列）分别降低了 94.09% 和 56.41%。

表 3 Windows 恶意软件分类结果

分类模型	字节序列	准确率/%	AUROC/%	训练时间/min	GPU 显存/GB
MalConv	原始	95.44	99.47	34.26	5.67
MalConv2	原始	95.38	99.12	34.01	1.56
ResNet18	采样	94.56	98.35	2.01	0.68

Android 平台的实验结果如表 4 所示。分析可以发现，在训练时间开销和 GPU 显存占用上，ResNet18（采样字节序列）相较于 MalConv（原始字节序列）分别降低了 90.91% 和 88.56%，相较于 MalConv2（原始字节序列）分别降低了 90.55% 和 57.24%。值得注意的是，基于采样字节序列的 ResNet18 在分类效果上显著领先，可能的原因在于，Android 和 Windows 平台软件的文件结构存在差异，1 048 576 的字节序列截断长度包含的 Android 恶意软件信息对于上述两个模型来说还不够丰富，而采样字节序列是原始字节序列整体分布的体现，保留了更多的信息。

表 4 Android 恶意软件分类结果

分类模型	字节序列	准确率/%	AUROC/%	训练时间/min	GPU 显存/GB
MalConv	原始	71.98	64.37	32.80	5.68
MalConv2	原始	77.35	82.72	31.52	1.52
ResNet18	采样	87.92	85.90	2.98	0.65

综上，本文提出的基于字节序列频域采样的恶意软件分类方法，与最先进的基于原始字节序列的方法分类效果相当，且能将模型的训练时间和 GPU 显存占用分别降低 90% 和 50% 以上。这对于大数据场景下海量样本的恶意软件分类具有实际意义。

3.3 采样长度分析

由于本文在不同的平台和分类模型上观察到的现象类似，针对采样长度对于模型分类效果的影响，本轮实验聚焦于 Windows 恶意软件的 ResNet18 模型（如表 3 所示）。依次设置采样长度 L 为 256、512、1 024、2 048、4 096、8 192、16 384、32 768、65 536，训练模型并记录测试集上的评估结果，其余设置与前述实验相同。

图 7 给出了不同采样长度下的实验结果。随着采样长度的增加，模型的分类效果整体呈波动上升趋势。本轮实验在采样长度 L 为 512、8 192 和 65 536 处呈现三个峰值，特别是当 $L=65\ 536$ 时，准确率为 96.25%，AUROC 为 99.49%，已超过该数据集上目前最先进的基于原始字节序列的分类模型 MalConv（如表 3 所示）。

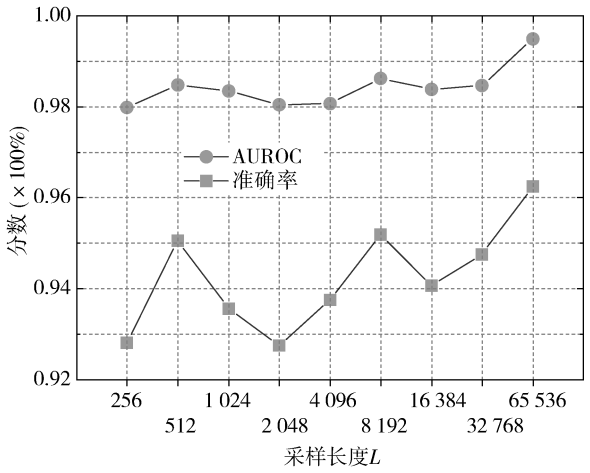

图 7 不同采样长度的实验结果

随着采样长度的增加，整体来看模型的分类效果更好。但过大的采样长度会导致模型的训练时间和 GPU 显存占用显著增加，因此需要在分类效果和训练效率之间进行权衡。本文的实验结果表明，当采样长度 $L \in [512, 8\,192]$ 时，模型的分类效果和训练效率能达到较好的平衡。

未来的研究将关注加壳恶意软件[19-20]对于采样字节序列的影响，以及评估时间序列分类领域的最新研究成果[21]在采样字节序列上的适用程度。

4 结论

本文提出基于字节序列频域采样的恶意软件分类方法，通过引入离散傅里叶变换和设计频域采样策略，保留字节序列中的主要低频分量，实现模型训练效率的提高。公开数据集上的实验分析表明，与最先进的基于原始字节序列的恶意软件分类方法相比，本文提出的方法与其分类效果相当，且能将模型的训练时间和 GPU 显存占用分别降低 90% 和 50% 以上，这对于大数据场景下的恶意软件分类任务具有实际意义。

参考文献

[1] AL-ASLI M. Review of signature-based techniques in antivirus products [C]//International Conference on Computer and Information Sciences, 2019: 1 - 6.

[2] OR-MEIR O, NISSIM N, ELOVICI Y, et al. Dynamic malware analysis in the modern era—a state of the art survey [J]. ACM Computing Surveys, 2019, 52 (5): 1 - 48.

[3] UCCI D, ANIELLO L, BALDONI R. Survey of machine learning techniques for malware analysis [J]. Computers & Security, 2019, 81: 123 - 147.

[4] RAFF E, ZAK R, COX R, et al. An investigation of byte n-gram features for malware classification [J]. Journal of Computer Virology and Hacking Techniques, 2018 (14): 1 - 20.

[5] RAFF E, SYLVESTER J. Learning the pe header, malware detection with minimal domain knowledge [C]// ACM Workshop on Artificial Intelligence and Security, 2017: 121 - 132.

[6] KI Y, KIM E, KIM H K. A novel approach to detect malware based on API call sequence analysis [J]. International Journal of Distributed Sensor Networks, 2015, 11 (6): 659101.

[7] RIECK K, HOLZ T. Learning and classification of malware behavior [C]// International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2008: 108 - 125.

[8] AARAJ N, RAGHUNATHAN A. Dynamic binary instrumentation-based framework for malware defense [C]//Detection of Intrusions and Malware, and Vulnerability Assessment, 2008: 64 - 87.

[9] RHEE J, RILEY R, XU D, et al. Kernel malware analysis with untampered and temporal views of dynamic kernel memory [C]// Recent Advances in Intrusion Detection, 2010: 178 - 197.

[10] AGARWAL S, RAJ G. framework for real time analysis of malware [C]// International Conference on Cloud Computing, Data Science & Engineering, 2018: 14 - 15.

[11] RAFF E, BARKER J, SYLVESTER J, et al. Malware detection by eating a whole exe [C]//Workshop of AAAI Conference on Artificial Intelligence, 2018.

[12] RAFF E, FLESHMAN W, ZAK R, et al. Classifying sequences of extreme length with constant memory applied to malware detection [C]//AAAI Conference on Artificial Intelligence, 2021: 9386 - 9394.

[13] VAN DEN OORD A, DIELEMAN S, ZEN H, et al. Wavenet: a generative model for raw audio [J]. arXiv preprint arXiv: 1609. 03499, 2016.

[14] ANDERSON H S, ROTH P. Ember: an open dataset for training static PE malware machine learning models [J]. arXiv preprintarXiv: 1804. 04637, 2018.

[15] BRIGHAM E O, MORROW R. The fast Fourier transform [J]. IEEE Spectrum, 1967, 4 (12): 63 - 70.

[16] ARP D, SPREITZENBARTH M. Drebin: effective and explainable detection of android malware in your pocket [C]//Network and Distributed System Security, 2014: 23 - 26.

[17] JIANG Y, LI G, LI S, et al. BenchMFC: a benchmark dataset for trustworthy malware family classification under concept drift [J]. Computers & Security, 2024, 139: 103706.

[18] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition [C]//IEEE Conference on Computer Vision and Pattern Recognition, 2016: 770 - 778.

[19] AGHAKHANI H, GRITTI F, MECCA F. Whenmalware is packin′ heat; limits of machine learning classifiers based on static analysis features [C]//Network and Distributed Systems Security, 2020.

[20] UGARTE-PEDRERO X, BALZAROTTI D. SoK: deep packer inspection: a longitudinal study of the complexity of run-time packers [C]//IEEE Symposium on Security and Privacy, 2015: 659 - 673.

[21] NIE Y, NGUYEN N H. Atime series is worth 64 words: long - term forecasting with transformers [C]//International Conference on Learning Representations, 2023.

（收稿日期：2024 - 11 - 24）

作者简介：

蒋永康（1996 - ），男，博士，工程师，主要研究方向：网络安全与人工智能。

孙逊（1987 - ），男，博士，高级工程师，主要研究方向：电子信息技术。

杨玉龙（1988 - ），男，硕士，高级工程师，主要研究方向：密码学与网络安全。

版权声明

凡《网络安全与数据治理》录用的文章，如作者没有关于汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权等版权的特殊声明，即视作该文章署名作者同意将该文章的汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权授予本刊，本刊有权授权本刊合作数据库、合作媒体等合作伙伴使用。同时，本刊支付的稿酬已包含上述使用的费用，特此声明。

《网络安全与数据治理》编辑部