

基于 TensorFlow 的恶意代码片段自动取证检测算法

李炳龙, 佟金龙, 张宇, 孙怡峰, 王清贤, 常朝稳

(信息工程大学密码工程学院, 河南 郑州 450001)

摘 要: 针对数字犯罪事件调查, 在复杂、异构及底层的海量证据数据中恶意代码片段识别难的问题, 通过分析 TensorFlow 深度学习模型结构及其特性, 提出一种基于 TensorFlow 的恶意代码片段检测算法框架; 通过分析深度学习算法训练流程及其机制, 提出一种基于反向梯度训练的算法; 为解决不同设备、不同文件系统的证据源中恶意代码片段特征提取问题, 提出一种基于存储介质底层的二进制特征预处理算法; 为进行反向传播训练, 设计并实现了一个代码片段数据集制作算法。实验结果表明, 基于 TensorFlow 的恶意代码片段检测算法针对不同存储介质以及证据存储容器中恶意代码片段的自动取证检测, 综合评价指标 F_1 达到 0.922, 并且和 CloudStrike、Comodo、FireEye 等杀毒引擎相比, 该算法在处理底层代码片段数据方面具有绝对优势。

关键词: 自动取证; 深度学习; 全连接神经网络; 恶意代码片段

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2021048

Auto forensic detecting algorithms of malicious code fragment based on TensorFlow

LI Binglong, TONG Jinlong, ZHANG Yu, SUN Yifeng, WANG Qingxian, CHANG Chaowen

College of Cryptographic Engineering, Information Engineering University, Zhengzhou 450001, China

Abstract: In order to auto detect the underlying malicious code fragments in complex, heterogeneous and massive evidence data about digital forensic investigation, a framework for malicious code fragment detecting algorithm based on TensorFlow was proposed by analyzing TensorFlow model and its characteristics. Back-propagation training algorithm was designed through the training progress of deep learning. The underlying binary feature pre-processing algorithm of malicious code fragment was discussed and proposed to address the problem about different devices and heterogeneous evidence sources from storage media and such as AFF forensic containers. An algorithm which used to generate data set about code fragments was designed and implemented. The experimental results show that the comprehensive evaluation index F_1 of the method can reach 0.922, and compared with

收稿日期: 2020-11-16; 修回日期: 2021-02-01

通信作者: 李炳龙, lbl2017@163.com

基金项目: 国家自然科学基金 (60903220)

Foundation Item: The National Natural Science Foundation of China (60903220)

论文引用格式: 李炳龙, 佟金龙, 张宇, 等. 基于 TensorFlow 的恶意代码片段自动取证检测算法[J]. 网络与信息安全学报, 2021, 7(4): 154-163.

LI B L, TONG J L, ZHANG Y, et al. Auto forensic detecting algorithms of malicious code fragment based on TensorFlow[J]. Chinese Journal of Network and Information Security, 2021, 7(4): 154-163.

CloudStrike, Comodo, FireEye antivirus engines, the algorithm has obvious advantage in dealing with the underlying code fragment data from heterogeneous storage media.

Keywords: auto forensics, deep learning, full connected network, malicious code fragment

1 引言

随着移动互联网技术的迅猛发展, 数字犯罪事件频繁发生, 磁盘介质容量不断增大以及存储犯罪事件信息数字设备数量的不断增加, 造成司法机构在进行事件调查过程中需要处理的数字证据量剧增。据德克萨斯州司法机构 2019 年数字取证能力分析报告称: 美国联邦调查局 (FBI, Federal Bureau of Investigation) 拥有最好的取证实验室, 但却积压了长达超过 9 个月的数字证据量, 并且因为大量的数字证据不能进行有效分析, 从而导致结案量不得不减少^[1-2]。此外, 由于犯罪事件证据来源于计算机、智能手机、平板电脑, 甚至物联网设备及可穿戴设备等不同类型的设备, 这些海量证据因具有不同操作系统和文件系统元数据信息, 导致了犯罪事件证据分析的极大差异性^[3-4]。另外, 为确保数字犯罪证据分析的完整性和可重复性, 数字犯罪证据需要通过存储介质映像技术将不同设备中的证据保存在 AFF、E01、RAW 等证据容器中^[5-7], 以底层二进制格式存储在取证容器中的证据数据导致取证分析越来越复杂。因此, 为解决数字犯罪事件中数字证据分析的大数据性、证据差异性以及复杂性, 自动取证分析技术成为数字取证领域的重点研究问题, 目前, 该技术已经取得了初步研究成果。有学者探讨了高度自动化数字取证的必要性和重要性, 并分析了自动化取证的优点^[8]。此外为提升取证分析的自动性, 在经典的取证套件中增加按钮式自动取证功能, 如 EnCase、Forensic ToolKit、Autopsy Forensic Browser 等具有全功能的取证工具套件允许取证调查人员仅通过知道要按哪一个按钮就能进行初步的, 甚至一些复杂的调查分析任务^[8-10]。这些受欢迎的工具让取证调查人员的工作更加容易, 并提升了自动化取证能力。而且功能较为单一的取证软件 TraceHunter^[11]也能够提供关联、解释以及 Windows 注册表分析的自动取证功能。此外数字取证领域中证据分类技术是一个快速增长的且自动化程度较高的研究

方向, 许多研究^[12-16]已经支持在计算机和移动电话中, 相关证据自动分类的功能, 从而便于取证人员快速、自动、实时获取证据。根据美国 FBI 发布的 2019 年互联网犯罪调查报告分析: 自动取证技术有利于快速、自动进行犯罪事件分析, 其已经成为数字调查中数字证据深度分析约减的关键技术^[17]。然而, 有学者针对手工调查和证据自动分类进行了比较研究, 研究结果表明, 在较为复杂的网络攻击犯罪调查中, 如犯罪分子将恶意代码以分片的形式存储在网络硬盘或者在对等网络存储系统中, 因缺乏恶意代码的整体知识而导致自动分类取证技术检测漏掉潜在的证据。

此外, 恶意软件威胁日益增加, 已经成为数字取证检测的难点。根据 McAfee 实验室的报告, 2019 年第一季度该实验室新增超过 6.5×10^7 个新的恶意软件^[18]。传统的恶意软件检测机制依赖于在恶意软件样本中提取签名特征, 并将这些特征存储在数据库中。然而, 提取恶意软件样本特征需要进行大量的手工分析, 并且基于签名特征的恶意代码检测技术难以有效跟上恶意软件数量的快速增长, 其根本原因是恶意代码签名扫描技术仅对已知恶意软件样本有效, 而对新增未知恶意软件无效。另外一个经典方法是根据恶意软件的运行行为进行检测, 该方法涉及运行恶意软件样本, 并且观察其运行行为。尽管该方法能够改善未知恶意软件的检测, 但这种方法容易受到虚拟机逃逸技术恶意代码的阻扰。此外, 可疑恶意代码运行还需要耗费大量时间和计算资源。由于这两种恶意代码检测技术方面的局限性, 加上犯罪分子在大容量存储介质中不断增加的通过分片、加密等反取证手段, 造成恶意代码检测难度更大。

研究人员利用机器学习模型, 训练学习恶意软件特征, 从而增强检测精确度、提升速度^[19-21]。深度学习 (deep learning) 是机器学习领域的分支, 是一种试图使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的算法。深度学习在图像分类、语音识别、机器翻译

等领域获得了突破性的进展, 产生了大量的研究成果^[22-24]。然而, 要获得一个好的深度学习模型需要针对每个具体问题(如图像分类)研究其深度学习框架, 并进行长期调优(即通过训练使模型参数最优), 这使深度神经网络学习方法应用具有局限性。为此, Lukasz 等^[25]研究探索了统一的深度学习模型, 即通过构建一个模型自适应地解决图像分类、语音识别、机器翻译等不同领域、不同数据模态下的多个不同类型的任务, 且在特定任务上的性能没有明显损失或接近于现有的主流方法。该模型主要适用于图像分类、语音识别及机器翻译等问题。

TensorFlow 是一个采用数据流图, 用于数值计算的开源软件库^[26], 是谷歌公司开发的深度学习框架, 也是深度学习的主流框架之一^[22], 它可以实现卷积神经网络(CNN)、循环神经网络(RNN)和深度神经网络(DNN)等经典算法, 并应用于语音识别、自然语言处理、计算机视觉等方面^[23]。此外, TensorBoard 是与 TensorFlow 配套的可视化工具, 能够实现深度学习程序运行过程的可视化, 包括对训练过程中参数变化、损失及准确率等的可视化^[26]。TensorFlow 平台已经被广泛应用于工业界和学术界, 许多提供源代码的深度学习文章使用 TensorFlow 来实现其模型。

针对数字犯罪事件调查, 在复杂、异构及底层的海量证据数据中恶意代码片段识别问题, 本文采用深度学习模型和理论, 从证据数据存储的底层特征出发, 探索恶意代码片段自动取证检测问题。尽管 Yara 规则能够针对磁盘等存储介质进行恶意软件检测自动化^[27]。然而, 该方法的本质与签名检测类似, 即该方法仍需要恶意软件样本的对应规则, 才能够检测到对应的恶意软件。本文提出的方法受统一模型^[25]的启发, 但和文献[25]方法的不同之处在于, 不是通过在现有深度学习框架中增加不同操作系统类型的恶意代码处理模块, 而是根据恶意代码在存储介质底层将“扇区”或者“簇”的二进制片段数据转换为深度学习模型的高维特征数据, 并进而利用 TensorFlow 深度学习模型网络结构进行训练调参, 获得适合恶意代码片段处理的深度学习模型。为此, 本文首先通过分析 TensorFlow 深度学习模型结构及其特

性, 提出一种基于 TensorFlow 的恶意代码片段检测算法框架; 其次, 通过分析深度学习算法训练流程及其机制, 提出一种基于反向梯度训练的算法; 最后, 为自适应 TensorFlow 模型输入数据要求, 提出一种恶意代码片段特征转化算法。

2 恶意代码片段识别算法

2.1 基于 TensorFlow 的恶意代码片段识别算法框架

图 1 给出了本文提出的恶意代码片段识别算法框架, 该框架包括 3 个主要模块。第一个模块是基于 TensorFlow 恶意代码片段自动识别框架。该模块为解决磁盘敏感扇区识别问题使用的深度学习网络是全连接网络(FCN), 即每个神经元与前后相邻层的每一个神经元都有连接关系, 输入是 4 096 维特征向量, 输出为正常或恶意的预测结果。第二个模块是基于恶意代码训练集的训练。利用恶意代码片段数据训练集训练深度学习模型, 微调深度学习模型相关参数, 学习并得到恶意代码片段的抽象特征。第三个模块是利用生成的 FCN 模型进行恶意代码片段的自动识别与检测。通过训练的分层深度学习模型对待检测代码片段进行检测与分类。本文采用 TensorFlow 深度学习开源框架^[8]构建恶意代码片段自动取证算法, 其中, 输入节点数为 4 096, 模型输出节点数为 2 的分类输出。

2.2 基于反向传播的训练策略机制

基于反向传播的训练策略机制是恶意代码片段识别算法框架的主要模块, 其流程如图 2 所示。该算法流程为训练过程开始后, 如果存在模型则恢复模型, 否则直接开始进入训练循环, 设定每 1 000 轮次训练保存一次模型参数, 并计算、打印当前损失值。该算法中增加了训练模型保存功能, 其目的在于实现断点续训, 可以在损失值趋于稳定后手动停止, 或者运行直到给定轮次的训练。反向传播训练算法实施过程中需要考虑两点: 一是采用随机参数初始化方法, 其目的在于使参数服从正态分布或均匀分布, 确保网络层不同神经元对不同的输入有不同的输出, 并确保网络训练过程中有好的收敛效果; 二是训练优化方法, 即在深度学习模型中采用交叉熵(cross entropy)损

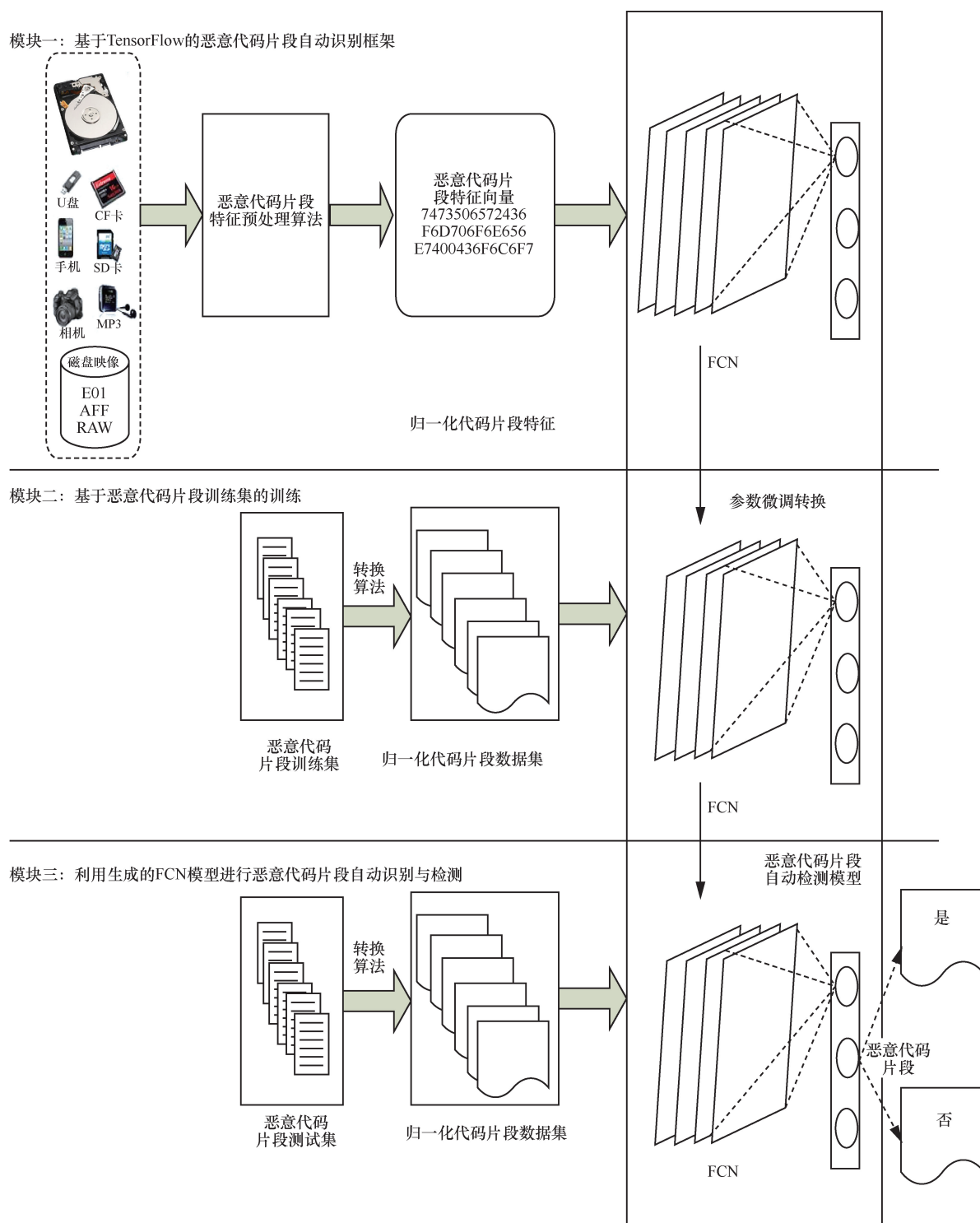


图1 恶意代码片段识别算法框架
Figure 1 Forensic algorithm framework for malicious code fragment

失函数寻找模型的最优解，TensorFlow 根据输入模型预测值和实际值得到损失函数，再计算损失函数的梯度，并根据梯度调整模型参数。此外，为提升恶意代码片段自动识别算法框架的泛化能

力引入正则化机制。在损失函数中引入模型复杂指标，给每个权重参数加上权重，抑制训练数据中的噪声，注意一般不对模型中的偏置参数使用。另外，学习率的设置对训练有很大影响，本文选

择指数衰减学习率，在训练过程中动态调整学习率，每隔一段轮次，计算学习率衰减率，并更新学习率：新学习率=学习率初值×学习率衰减率。滑动平均的作用是记录每个参数过去一段时间内的平均值，像影子一样缓慢变化，也能增加模型的泛化性。滑动平均针对所有参数优化。

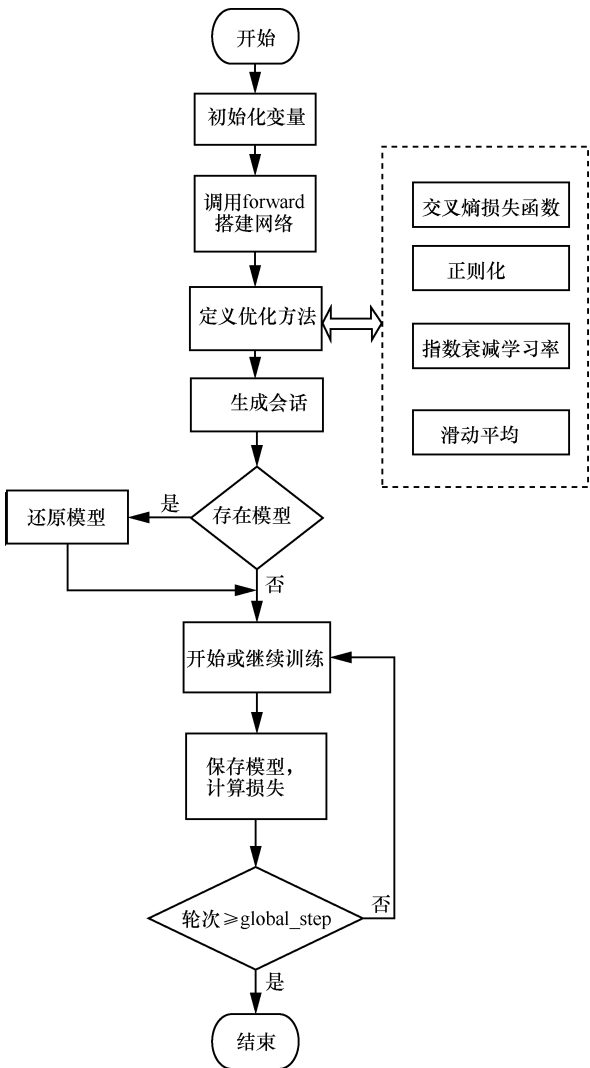


图 2 反向训练传播算法流程
Figure 2 Back training propagation algorithm process

2.3 恶意代码片段特征预处理算法

数字事件调查的证据来源于不同设备和不同文件系统类型的存储介质，为此，本文设计并实现了恶意代码片段特征预处理算法，其流程如图 3 所示。恶意代码特征预处理算法主要过程如下。

(1) 根据原始证据进行文件系统类型和证据存储容器类型的识别，确定原始证据的文件系统

类型或者证据存储容器类型。

(2) 依据存储介质文件系统特征，或者 AFF、E01 等存储容器原理，解析出存储介质中文件数据存储的起始与结束位置，以及文件数据存储的簇大小（“簇”由多个扇区构成，每个扇区大小为 512 byte），本文中记录文件数据存储的起始与结束位置为恶意代码片段的起始与结束位置，文件数据存储的簇大小为恶意代码片段大小。

(3) 从恶意代码片段的起始位置开始，以恶意代码片段大小为读取单元，以十六进制格式读取恶意代码数据，将这个恶意代码片段的十六进制数据称为恶意代码的预处理特征，这个预处理特征将作为深度学习模型框架的直接输入特征。

恶意代码片段大小对应的是存储介质中的“文件簇”存储单元，其本身是存储介质中扇区的整数倍，因此在恶意代码片段特征预处理过程中，不存在片段数据少或者多的问题，不需要考虑恶意代码片段特征数据补齐或者裁减的问题。但是，恶意代码片段大小对深度学习网络模型训练和实际检测有一定的影响。

2.4 代码片段数据集制作算法

代码片段训练集和数据集对深度学习网络模型的构建和评估至关重要，基于恶意代码特征预处理算法对磁盘等介质处理完成后，会得到大量的数据样本文件，其结果都是大小为 4 kB 的包含某类代码片段的二进制数据文件。目前没有基于磁盘等存储介质的代码片段数据集，因此在预处理阶段，本文选择训练用的正常代码、恶意代码数据集各约 150 MB，并确保正常代码和恶意代码分别来自 Android、Linux 及 Windows 平台，且数量均衡。分别保存于两个文件夹（normal、malware），数据集制作完成后，经过调整，最终包含 39 944 个正常代码片段文件和 40 056 个恶意代码片段文件信息，共计 80 000 个样本文件。

测试集可以用于评估已训练模型的效果，在本文实验中，测试集用于检验算法，没有单独设计验证集。制作测试集时随机选择了两类训练数据各 2 500 个，共 5 000 个，另外重新用上述方法制作了 5 000 个新的代码片段文件，共计 10 000 个。

在上述处理的基础上为数据样本代码片段文

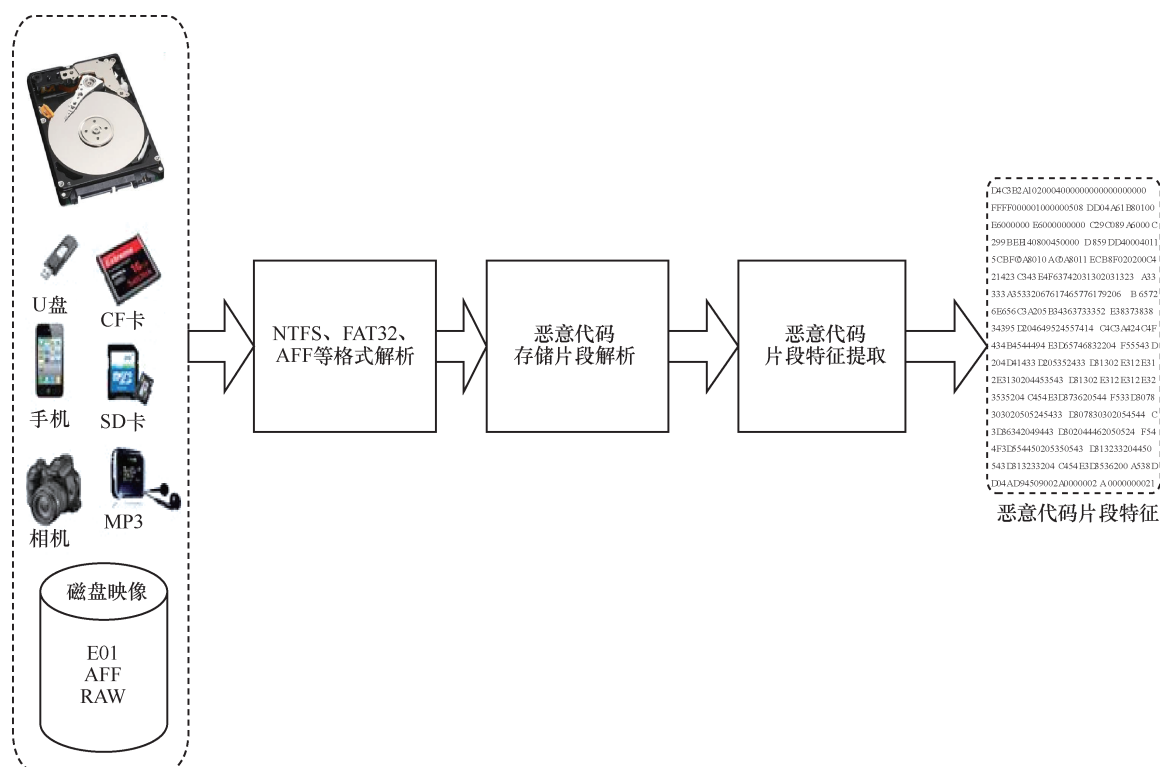


图3 恶意代码片段特征预处理算法流程

Figure 3 Feature preprocessing algorithm of malicious code fragment process

件打标签。得到分好类的数据样本代码片段文件后, 利用批处理方式对数据代码片段文件添加标签, 具体算法过程如下。

(1) 遍历 `normal` 目录下的所有文件名, 每个文件名一行, 并在文件名后添加标签“0”, 保存至 `label0.txt`, 该文件表示正常代码片段标签。

(2) 遍历 `malware` 目录下的所有文件名, 每个文件名一行, 并在文件名后添加标签“1”, 最终得到 `label1.txt`, 该文件表示恶意代码片段标签。

(3) 将正常代码片段标签和恶意代码片段标签文件内容进行合并, 并利用伪随机方法将合并的标签内容打乱, 使标签“0”和标签“1”数据随机分布, 得到共计 80 000 行的随机顺序文本, 将其保存为 `train_labels.txt`。使用上述算法制作测试集标签文件 `test_labels.txt`。

此外, 为提高基于深度学习的恶意代码片段自动取证检测算法的运行效率, 减少训练过程读取文件耗费的时间, 利用 `tfrecords` 文件对训练数据集及其标签 (包括测试集及其标签) 进行处理, 具体算法过程如图 4 所示。

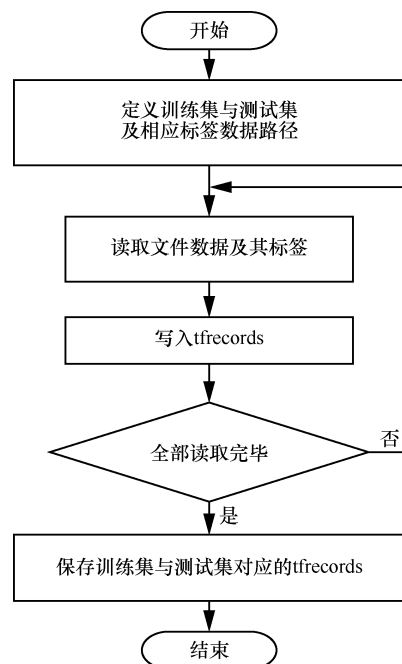


图4 数据集处理算法流程

Figure 4 Data set processing algorithm process

根据图 4 所示的数据集处理算法, 得到如图 5 的两个 `tfrecords` 文件。

名称	修改日期	类型	大小
disk_test.tfrecords	2019/4/8 星期一 11:21	TFRECORDS 文件	40 557 kB
disk_train.tfrecords	2019/4/8 星期一 11:21	TFRECORDS 文件	324 454 kB

图 5 Tfrecords 文件
Figure 5 Tfrecords file

3 实验结果及分析

3.1 数据集

基于深度学习的恶意代码片段的自动取证检测算法所需的数据集需要根据 Windows 系统中 FAT32 文件系统格式, 选择恶意代码片段大小为 4 000 byte, 这也是大容量磁盘中默认的文件存储基本单位 (4 000 byte 对应于 8 个扇区的大小)。结合恶意代码片段预处理算法和数据集制作算法, 具体步骤如下。

(1) 准备正常代码和恶意代码, 二者数量 (总大小) 相当。

(2) 将上述两类代码分别写入干净的磁盘, 磁盘已经利用 WinHex 工具进行全 0 填充。

(3) 根据磁盘文件系统目录表, 定位单个程序。

(4) 以 4 000 byte 为单位读取程序 (正常程序或者恶意程序) 数据, 并另存为 4 000 byte 大小的文件, 文件名为程序的唯一标识 ID+序号。

(5) 逐一读取每个代码数据, 并将正常代码和恶意代码分开存放, 方便制作数据集。

(6) 共生成 4 个目录的文件, 分别为正常代码片段训练 (train_positive)、恶意代码片段训练 (train_virus)、正常代码片段测试 (test_positive)、恶意代码片段测试 (test_virus)。

3.2 不平衡样本处理方法

数据集质量会直接影响全连接深度学习网络功能效果, 因为网络训练学习过程是根据训练数据进行的。如果全连接深度学习网络训练了很久, 损失降低很慢甚至不收敛, 或者准确率一直很低, 在全连接深度学习网络没有错误的情况下, 可能是数据集出现了问题, 问题可能包括但不限于: 恶意代码片段和正常代码片段数量比例失衡, 某类数据比例过大; 数据集标签没有打乱顺序, 如存在前半全是 “0” 的情况, 即前半全是正常代码片段, 后半全是 “1” 的情况, 即后半全是恶意代码片段; 收集的数据不合格或标签制作错误。

为提升全连接深度学习网络训练效果, 针对数据集中恶意代码和正常代码数量上的比例失衡问题, 以及代码片段标签没有打乱顺序等不平衡问题, 本文提出了数据集样本不平衡处理机制, 重新制作了相对平衡的数据集: 正常和恶意的程序样本数量相当, 都是 40 000 左右; 标签文件打乱了顺序, 进行了随机重排列。

3.3 实验结果

3.3.1 FCN 训练过程及准确率

经过测试与调整, 本文发现 3 层全连接神经网络能够达到最佳的取证检测结果。根据经验设置的全连接神经网络反向传播超参数值为: 学习率初值为 0.1, 学习率衰减率为 0.99, 正则化系数为 0.000 1, 参数的滑动平均为 0.99。同时利用训练数据集进行训练, 每次输入 200 组数据, 每 1 000 轮次保存训练结果, 共训练 80 000 轮次。生成的反向训练算法数据流程如图 6 所示。

另外, FCN 模型在平滑参数设置为 0.6 时训练过程中的损失值变化趋势如图 7 所示, 准确率变化趋势如图 8 所示, 可以看出, 在训练初始期间, 训练数据的损失值 (绿色线)、准确率 (绿色线) 与 FCN 模型的目标函数和准确率未完全拟合, 但随着训练次数不断增加, 两者完全拟合并趋于稳定, 准确率达到 99%。

此外, 本文还进行了 5 层全连接神经网络 (其中包含 3 个隐藏层) 构建, 利用数据集进行训练发现: 训练开始后仅第一次计算出一个较小的损失值, 之后训练得到的损失值是 ‘nan’, 也就是损失值过小无法计算, 该实验结果从另一方面证实了: 理论上, 当全连接神经网络规模达到 5 层时, 训练过程就会出现梯度消失问题。虽然将激活函数调整为 relu 后训练能够正常开展, 但网络结构的增加使训练时间大幅增长, 因为参数呈指数增长, 且经训练生成的网络模型检测结果并不如 3 层全连接神经网络。可能的原因是网络输入参数数量太大, 4 096 个输入从一定意义上来说限制了网络规模。

3.3.2 算法具体应用及比较

本文采用精确度 (P)、召回率 (R) 以及 F_1 指标评价恶意代码片段的分类检测性能。假设 TP 表示恶意代码片段样本预测为恶意代码片段类型的

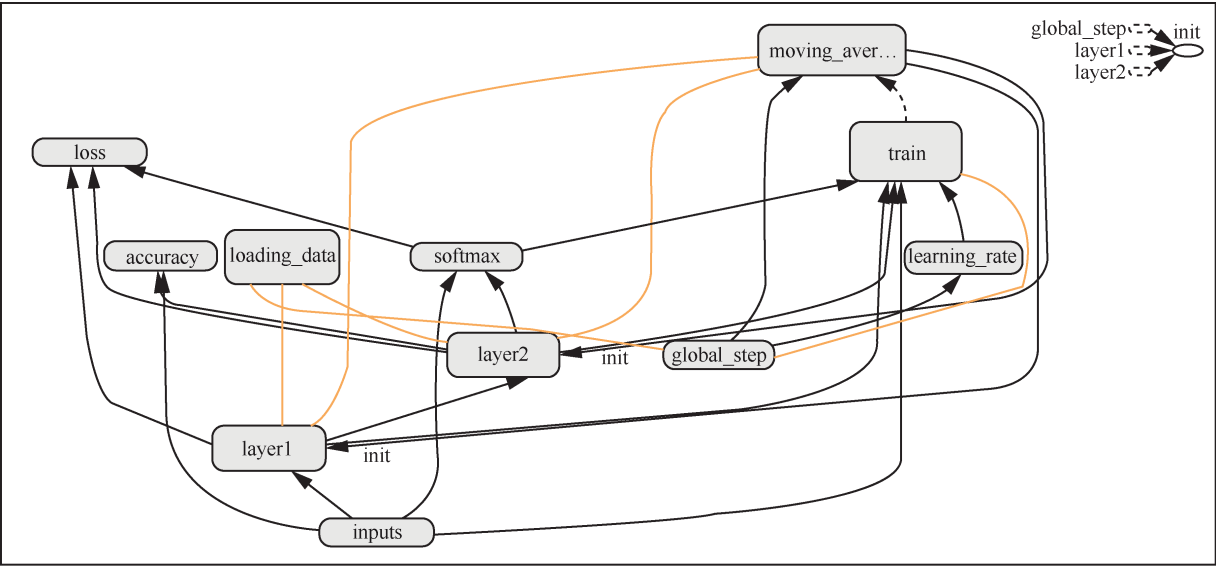


图 6 Tensorboard 自动生成的数据流
Figure 6 The data flow based on Tensorboard

个数，FP 表示正常代码片段样本预测为恶意代码片段类型的个数，TN 表示正常代码片段样本预测为正常代码片段类型的个数。其公式定义如下：

$$P=\frac{TP}{TP+FP}$$

$$R=\frac{TP}{TP+FN}$$

$$F_1=\frac{2PR}{P+R}$$

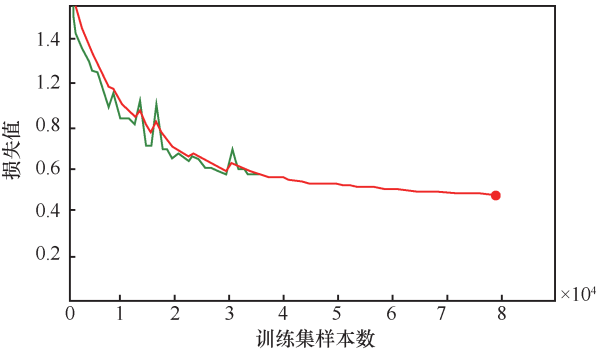


图 7 损失值随训练过程变化趋势
Figure 7 The losses change with the training process

VirusTotal 是一款由独立的 IT 安全实验室 Hispasec Sistemas 所开发的服务，它使用了多种反病毒引擎。利用 VirusTotal 针对测试集中的恶意代码片段进行查杀，并和本文方法进行对比，实验结果如表 1 所示。

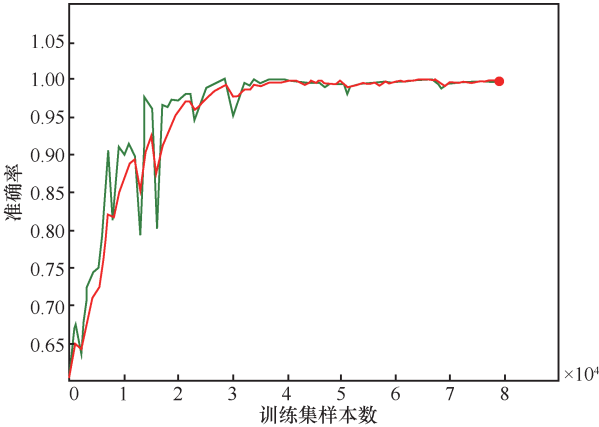


图 8 准确率随训练集样本数变化趋势
Figure 8 The accuracy trends with the training process

表 1 基于 VirusTotal 网站测试集数据取证检测结果
Table 1 Forensic results about the test data set by VirusTotal

方法	<i>P</i>	<i>R</i>	<i>F</i> ₁
CrowdStrike	0.667	0.096	0.168
Comodo	0.474	0.086	0.146
FireEye	1.000	0.096	0.175
ClamAV	1.000	0.102	0.185
McAfee	1.000	0.092	0.168
Kaspersky	1.000	0.094	0.172
本文方法	0.896	0.950	0.922

从表 1 可以看出，FireEye、ClamAV、McAfee 和 Kaspersky 针对测试集中恶意代码片段的检测

精确度达到 1.000, 而 CrowdStrike 和 Comodo 的精确度则分别为 0.667 和 0.474, 这些结果表明后两个杀毒引擎将部分正常代码片段误预测为恶意代码片段, 而前者杀毒引擎则没有将正常代码片段误预测为恶意代码片段。此外, VirusTotal 杀毒引擎的 F_1 综合指标及召回率结果都不理想, 可能原因是这些杀毒引擎通常用于恶意文件整体特征进行检测, 即能够检测出在恶意文件中的局部恶意代码片段, 但检测不出属于该恶意文件的其他代码片段。本文提出的基于 TensorFlow 的恶意代码片段自动取证检测算法的综合评价指标 F_1 达到 0.922, 这说明该方法能够有效检测出数字取证调查过程涉及的存储介质 (包括 RAW、AFF 等取证容器) 的底层恶意代码片段。

4 结束语

本文基于 TensorFlow 深度学习模型, 提出了一个恶意代码片段自动识别算法框架; 基于该框架设计了一种基于后向传播的训练机制; 从二进制存储底层角度, 提出一种针对磁盘、U 盘、智能手机, 以及 RAW、E01、AFF 等证据容器的不同存储介质的恶意代码片段预处理算法; 设计并实现了一个代码片段数据集制作算法; 算法实验结果表明该方法能够针对不同存储介质以及证据存储容器中恶意代码片段的自动取证检测, 综合评价指标 F_1 达到 0.922, 并且和 CloudStrike、Comodo、FireEye 等相比, 该算法在处理底层代码片段数据方面具有绝对优势。未来工作将针对内存取证中的内存映像进行分析。

参考文献:

- [1] TEPE A N, BOYLAN A A, DAVIS D W. Analysis of digital forensic capabilities in texas law enforcement agencies[R]. The Bush School of Government & Public Service, 2019.
- [2] CAVIGLIONE L, WENDZE S, MAZURCZKY W. The future of digital forensics: challenges and the road ahead[J]. IEEE Security & Privacy, 2017, 15(6): 12-17.
- [3] LIN X D, CHEN T, ZHU T, et al. Automated forensic analysis of mobile applications on Android devices[J]. Digital Investigation, 2018, 26: S59-S66.
- [4] 高元照, 李炳龙, 陈性元. 基于 MapReduce 的 HDFS 数据窃取随机检测算法[J]. 通信学报, 2018, 39(10): 11-21.
- [5] GAO Y Z, LI B L, CHEN X Y. Stochastic algorithm for HDFS data detection based on MapReduce[J]. Journal on Communications, 2018, 39(10): 11-21.
- [6] ZAWOAO S, HASAN R. Digital forensics in the age of big data: challenges, approaches, and opportunities[C]//IEEE Big Data Security 2015. 2015: 1-7.
- [7] SERVIDA F, CASEY E. IoT forensic challenges and opportunities for digital traces[J]. Digital Investigation, 2019, 28: S22-S29.
- [8] 韩宗达, 李炳龙. 基于证据库的数据证据转换模型[J]. 计算机应用研究, 2015, 32(7): 2140-2143.
- [9] HAN Z D, LI B L. Evidence conversion model based on evidence database[J]. Application Research of Computers, 2015, 32(7): 2140-2143.
- [10] JAMES J I, PAVEL G. Challenges with automation in digital forensic investigations[J]. arXiv:1303.4498, 2013.
- [11] Guidanc[EB].
- [12] AccessData[EB].
- [13] ZHU Y, JAMES J, GLADYSHEV P. A consistency study of the windows registry[C]//6th IFIP WG 11.9 International Conference on Digital Forensics(DF). 2010: 77-90.
- [14] ROGERS, M, GOLDMAN J, et al. Computer forensics field triage process model[J]. Journal of Digital Forensics, Security and Law, 2006, 1(2): 27-40.
- [15] MISLAN R P, CASEY E, et al. The growing need for on-scene triage of mobile devices[J]. Digital Investigation, 2010, 6(3-4): 112-124.
- [16] MARZIALE L, RICHARD G G, ROUSSEV V. Massive threading: using GPUs to increase the performance of digital forensics tools[J]. Digital Investigation, 2007, 4S: S73-S81.
- [17] GARFINKEL S, NELSON A, WHITE D, et al. Using purpose-built functions and block hashes to enable small block and sub-file forensics[J]. Digital Investigation, 2010, 7: S13-S23.
- [18] QUICK D, CHOO K K R. Impacts of increasing volume of digital forensic data: a survey and future research challenges [J]. Digital Investigation, 2014, 11 (4): 273-294.
- [19] Federal Bureau of Investigation(FBI). 2019 Internet Crime Report[R]. 2020.
- [20] McAfee Labs. 2019 Threats Report[EB]. 2019.
- [21] PIRSCOVEANU R, HANSEN S, CZECH A. Analysis of malware behavior: type classification using machine learning[C]//Proceedings of the 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA). 2015.
- [22] MOHAISEN A, ALRAWI O, MOHAISEN M. Amal: High-fidelity, behavior-based automated malware analysis and classification[J]. Computers & Security, 2015, 52: 251-266.
- [23] LIN Y D, LAI Y C, LU C N, et al. Three-phase behavior-based detection and classification of known and unknown malware[J]. Security and Communication Networks, 2015, 8(11): 2004-2015.
- [24] KRIS C, MAREK R. A Joint model for word embedding and word morphology[C]//Proc of the 1st Workshop on Representation Learning for NLP, August 11th. 2016: 18-26.
- [25] 陈翠平. 基于深度信念网络的文本分类算法[J]. 计算机系统应

用, 2015, 24(2): 121-126.

CHEN C P. Text Categorization based on deep belief network[J]. Computer System & Application, 2015, 24(2): 121-126.

- [24] 黎亚雄, 张坚强, 潘登, 等. 基于 RNN-RBM 语言模型的语音识别研究[J]. 计算机研究与发展, 2014, 51(9): 1936-1944.

LI Y X, ZHANG J Q, PAN D, et al. A study of speech recognition based on RNN-RBM language model[J]. Journal of Computer Research and Development, 2014, 51(9): 1936-1944.

- [25] LUKASZ K, AIDAN N. G, NOAM S, et al. One model to learn them all[J]. arXiv:1706.05137v1, 2017.06:1-10.

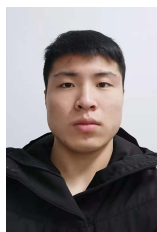
- [26] Ponemon Institute. 2017 Cost of Data Breach Study[R].

- [27] RAFF E, ZAK R, MUNOZ G, et al. Automatic yara rule generation using biclustering[C]//Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security. 2020: 71-82.

[作者简介]



李炳龙 (1974-), 男, 河南卫辉人, 博士, 信息工程大学副教授, 主要研究方向为数字调查与取证、网络入侵溯源追踪与取证、云计算取证、智能手机取证等。



佟金龙 (1997-), 男, 河北保定人, 信息工程大学助理工程师, 主要研究方向为信息智能安全。



张宇 (1996-), 男, 江苏连云港人, 信息工程大学硕士生, 主要研究方向为智能手机取证。



孙怡峰 (1976-), 男, 河南新乡人, 博士, 信息工程大学副教授, 主要研究方向为人工智能与信息安全。



王清贤 (1960-), 男, 河南卫辉人, 信息工程大学教授、博士生导师, 主要研究方向为网络与信息安全。



常朝稳 (1966-), 男, 河南滑县人, 博士, 信息工程大学教授、博士生导师, 主要研究方向为网络信息防御。