基于值导数 GRU 的移动恶意软件流量检测方法

周翰逊[1]，陈晨[1]，冯润泽[1]，熊俊坤[1]，潘宏[2]，郭薇[3]

（1. 辽宁大学信息学院，辽宁 沈阳 110036；2. 辽宁大学数字经济研究院，辽宁 沈阳 110036；3. 沈阳航空航天大学计算机学院，辽宁 沈阳 110135）

摘 要：针对移动恶意软件数量和种类的急剧增加给移动用户的信息安全带来的巨大挑战，提出了一种基于值导数 GRU 的移动恶意软件流量检测方法，旨在解决基于 RNN 的移动恶意软件流量检测方法难以捕获网络异常流量的动态变化和关键信息的问题。值导数 GRU 算法通过引入“累计状态变化”的概念，可以同时描述移动网络恶意流量的低阶和高阶动态变化信息。此外，通过增设池化层使算法可以捕获移动恶意流量的关键信息。最后，通过仿真实验分析累计状态变化、隐藏层和池化层对于值导数 GRU 算法性能的影响。实验表明，基于值导数 GRU 的移动恶意软件流量检测方法拥有较高的检测准确率。

关键词：网络安全；移动恶意软件；RNN；值导数 GRU；流量检测

中图分类号：TP393.4

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2020005

Mobile malware traffic detection approach based on value-derivative GRU

ZHOU Hanxun[1], CHEN Chen[1], FENG Runze[1], XIONG Junkun[1], PAN Hong[2], GUO Wei[3]

1. Information Academy, LiaoNing University, Shenyang 110036, China
2. Digital Economy Academy, LiaoNing University, Shenyang 110036, China
3. Computer Academy, Shenyang Aerospace University, Shenyang 110135, China

Abstract: For the dramatic increase in the number and variety of mobile malware had created enormous challenge for information security of mobile network users, a value-derivative GRU-based mobile malware traffic detection approach was proposed in order to solve the problem that it was difficult for a RNN-based mobile malware traffic detection approach to capture the dynamic changes and critical information of abnormal network traffic. The low-order and high-order dynamic change information of the malicious network traffic could be described by the value-derivative GRU approach at the same time by introducing the concept of “accumulated state change”. In addition, a pooling layer could ensure that the algorithm can capture key information of malicious traffic. Finally, simulation were performed to verify the effect of accumulated state changes, hidden layers, and pooling layers on the performance of the value-derivative GRU algorithm. Experiments show that the mobile malware traffic detection approach based on value-derivative GRU has high detection accuracy.

Key words: network security, mobile malware, RNN, value-derivative GRU, traffic detection

1 引言

移动网络的快速发展，使接入移动网络的移动设备的数量猛增。与此同时，移动用户越来越多地面临来自恶意软件等的严重威胁[1]。现代移动设备与电子商务、个人支付和社交通信等相关应用的关

收稿日期：2019–06–04；**修回日期**：2019–11–13

通信作者：郭薇，155381296@qq.com

基金项目：国家自然科学基金资助项目（No.61300233, No.61402298, No.61472169, No.51704138）；辽宁省教育厅基金资助项目（No.JYT19053）；辽宁省自然科学基金资助项目（No.2019-MS-149）

Foundation Items: The National Natural Science Foundation of China (No.61300233, No.61402298, No.61472169, No.51704138), Liaoning Provincial Department of Education Project (No.JYT19053), The Natural Science Foundation of Liaoning Province (No.2019-MS-149)

系日益密切，因此关注移动设备的安全问题变得尤为重要[2]。

移动恶意软件检测是构建移动网络防御体系的重要一环[3-4]。近年来，针对移动恶意软件检测的研究方法不断涌现。Khatri 等[5]开发出一种基于移动网络的恶意软件检测系统，用于检测网络中的恶意活动，保护最终用户免受移动恶意软件的攻击。Adeel 等[6]提出了基于感染节点行为的移动恶意软件分类方法。Moghaddam[7]将基于分类的 Android 恶意软件检测技术的静态特征划分为相关的类别，并研究这些静态特征对 Android 恶意软件检测效率的影响。Tripp 等[8]为 Android 创建了一个恶意软件检测系统—— MASSDROID，通过可扩展的数据流分析形式从调用图中收集安全相关操作的痕迹，然后传送到监督学习引擎以获得恶意/良性分类。Nguyen 等[9]提出了一种基于网络行为的 SDN 移动设备恶意软件检测系统，该系统由 IP 黑名单、连接成功率、连接速率 3 种算法组成，以更有效和灵活的方式开发恶意软件检测方法。但是，这些工作的根本问题在于无法摆脱对于特征工程的依赖。也就是说，这些工作所提出的移动恶意软件检测方法的优劣很大程度上取决于特征提取技术的好坏。

为了解决特征工程的依赖问题，基于深度学习的移动恶意软件检测方法逐渐引起学术界的关注。Li 等[10]使用基于深度学习的方法来检测 Android 恶意软件，并开发了自动检测引擎来检测恶意应用程序。Yuan 等[11]提出了一个基于深度学习的在线 Android 恶意软件检测引擎（DroidDetector），它可以自动检测应用程序是否是恶意软件，并使用数千款 Android 应用测试了 DroidDetector。Kim 等[12]提出了一种多模态深度学习方法来检测移动恶意软件，它使用基于相似性的特征提取方法来细化特征，以便在恶意软件检测上进行有效的特征表示。Su 等[13]提出一种基于深度学习模型的 Android 平台的恶意软件检测方法——DroidDeep，它可以从 Android 应用程序的静态分析中构建深度学习模型。然而，这些工作仅仅照搬已经成功应用在其他领域（例如计算机视觉、自然语言处理）的深度学习理论，没有考虑移动恶意软件攻击的特点，例如恶意软件在移动流量方面通常会在感染后的 5 min 突然激增[14]，导致这些工作所提出的移动恶意软件检测方法的检测效果无法取得显著提高。

为了结合深度学习模型和移动恶意软件流量方面的特点，本文提出基于值导数 GRU（value-derivative gated recurrent unit）的移动恶意软件流量检测方法来捕获移动恶意流量的动态变化信息和关键信息。通过引入“累计状态变化”，值导数 GRU 算法能够同时定量地描述移动网络恶意流量的低阶和高阶变化信息。此外，通过增设池化层使值导数 GRU 算法获取流量的重要信息。实验结果表明，基于值导数 GRU 的移动恶意软件流量检测方法比 GRU 算法拥有更高的准确率。

2　GRU 算法

循环神经网络（RNN, recurrent neural network）[15]是一种处理序列数据的深度神经网络。通过在相邻时间步的隐层单元之间引入循环连接，RNN 能够有效地利用历史信息来执行当前决策。但是，经过多阶段传播的 RNN 梯度倾向于消失或爆炸，导致 RNN 丧失学习长期依赖的能力，这种现象被称作梯度消失与爆炸[16]。

与 RNN 不同，长短期记忆（LSTM, long short-term memory）[17]不仅具有外部的隐层单元循环，而且具有内部的记忆细胞循环。此外，LSTM 是一种拥有 3 个特殊门控系统的循环神经网络，3 个门控系统分别被称作遗忘门、输入门、输出门。然而，复杂的门控系统要求 LSTM 拥有大量的网络参数，导致 LSTM 的训练代价居高不下。

为了解决 RNN 的梯度消失与爆炸问题、降低 LSTM 居高不下的训练代价，Cho 等[18]提出了一种新型循环神经网络——门控循环单元（GRU, gated recurrent unit）。相比于 RNN 和 LSTM，GRU 拥有相似而不同的设计模式。

1) 每个时间步的隐层单元拥有 2 个输入：前一个时间步隐层单元的输出和当前时间步的输入。

2) 相邻时间步的隐层单元之间存在循环连接，隐层状态从前一个时间步的隐层单元流入后一个时间步的隐层单元。

3) 每个时间步的隐层单元拥有 2 个门控系统，分别被称作更新门和重置门。

GRU 隐层单元结构如图 1 所示。其中，x_t 表示时间步 t 的输入，s_{t-1} 表示时间步 t–1 的隐层状态，s_t 表示时间步 t 的隐层状态，u_t 表示时间步 t 的更新门，r_t 表示时间步 t 的重置门。对于与时间步 t 关联的隐层单元而言，它的输入不仅包括当前时间步 t

的输入 x_t，还包括上一个时间步 t–1 的隐层单元的输出 s_{t-1}。

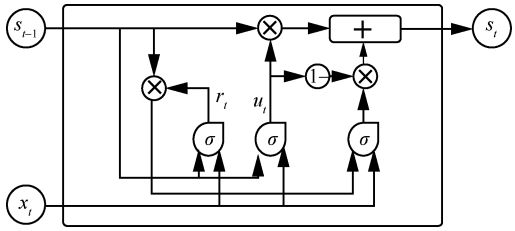


图 1 门控循环单元的隐层单元

在 GRU 隐层单元中，更新门 u_t 和重置门 r_t 共同决定隐层状态 s_t 的更新和重置，其计算式为

$$s_t = u_t \odot s_{t-1} + (1-u_t) \odot c_t \tag{1}$$

其中，c_t 表示时间步 t 的候选隐层状态，其计算式为

$$c_t = \tanh(\boldsymbol{W}_{xc} x_t + \boldsymbol{W}_{sc}(r_t \odot s_{t-1}) + b_c) \tag{2}$$

更新门 u_t 和重置门 r_t 能够独立地忽略候选隐层状态 c_t 和上一隐层状态 s_{t-1}。隐层状态 s_{t-1} 和输入 x_t 共同控制更新门 u_t 和重置门 r_t，经过激活函数 sigmoid 压缩之后输出一个[0,1]内的值，用于表示更新门 u_t 和重置门 r_t 的激活程度。特别地，当激活函数 sigmoid 输出 0 时，更新门 u_t 选择完全保留候选隐层状态 c_t，重置门 r_t 选择完全忽略上一隐层状态 s_{t-1}；当激活函数 sigmoid 输出 1 时，更新门 u_t 选择完全忽略候选隐层状态 c_t，重置门 r_t 选择完全保留上一隐层状态 s_{t-1}。更新门 u_t 和重置门 r_t 的计算式分别为

$$u_t = \text{sigmoid}(\boldsymbol{W}_{xu} x_t + \boldsymbol{W}_{su} s_{t-1} + b_u) \tag{3}$$

$$r_t = \text{sigmoid}(\boldsymbol{W}_{xr} x_t + \boldsymbol{W}_{sr} s_{t-1} + b_r) \tag{4}$$

式(1)~式(4)中，$\boldsymbol{W}_{xu}$、$\boldsymbol{W}_{xr}$、$\boldsymbol{W}_{xc}$ 分别表示输入单元到更新门、重置门以及隐层单元的权重矩阵，$\boldsymbol{W}_{su}$、$\boldsymbol{W}_{sr}$、$\boldsymbol{W}_{sc}$ 分别表示隐层单元到更新门、重置门以及隐层单元的权重矩阵，b_u、b_r、b_c 分别表示更新门、重置门以及隐层单元的偏置，⊙表示 2 个变量按照对应元素相乘。

3 值导数 GRU 算法

在数学中，导函数（或称导数）用于描述某一函数在定义域上每一点的变化趋势。也就是说，导函数可以定量地反映函数在定义域上每一点的局部变化程度。因此，函数与导函数的乘积（本文称之为值导数）可以反映函数在定义域上每一点的绝对变化程度。正是基于值导数这个数学概念，本文提出值导数 GRU 算法。

3.1 值导数 GRU 的隐层单元结构

基于 GRU 的移动恶意软件流量检测算法通过利用循环连接的 GRU 隐层单元可以记忆移动网络流量的完整静态信息。正如前文所述，隐层单元结构决定了 GRU 算法只能控制移动网络流量的流动，无法捕获移动网络流量的动态变化信息。由于恶意软件的移动流量通常会在感染后的 5 min 内突然激增，因此，本文引入的值导数 GRU 算法既要考虑移动网络流量的静态信息，又要考虑移动网络流量的动态变化信息。通过捕获移动网络流量的静态和动态信息，提高值导数 GRU 算法对于移动网络恶意流量的检测准确率。

值导数 GRU 隐层单元结构如图 2 所示。其中，x_t 表示时间步 t 的输入，s_{t-1} 表示时间步 t–1 的隐层状态，s_t 表示时间步 t 的隐层状态，u_t 表示时间步 t 的更新门，r_t 表示时间步 t 的重置门。通过在隐层单元内部增设更新门 u_t 和重置门 r_t 这 2 个门控系统，值导数 GRU 算法可以选择性地记忆或遗忘移动网络流量。也就是说，当移动网络流量流入隐层单元时，更新门 u_t 和重置门 r_t 可以独立地控制是否可以通过以及可以通过多少移动网络流量。为了保证更新门 u_t 和重置门 r_t 能够定量地控制可以通过的移动网络流量，隐层单元引入一阶状态变化 $s_{t-1} \odot \dfrac{\mathrm{d}s_{t-1}}{\mathrm{d}t}$，用于描述时间步 t 时移动网络流量的一阶动态变化。当网络空间遭受攻击时，伴随着移动网络异常流量的持续增加，一阶状态变化的值将显著增长，更新门 u_t 和重置门 r_t 的激活程度随之增大，进而能够定量地保留大部分移动网络流量；当网络空间正常运行时，由于内部几乎不存在移动网络异常流量，一阶状态变化的值将趋于平稳且接近于 0，更新门 u_t 和重置门 r_t 的激活程度随之减小，此时仅仅定量地保留小部分移动网络流量。

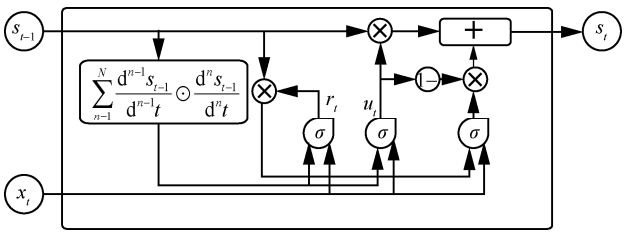


图 2 值导数 GRU 的隐层单元

本文把一阶状态变化 $s_{t-1} \odot \dfrac{\mathrm{d}s_{t-1}}{\mathrm{d}t}$ 作为一个核心因素来控制移动网络流量的信息流动，能够保证基

于值导数 GRU 的移动恶意软件流量检测算法可以有效地捕获移动网络流量的动态变化信息。此外，n阶状态变化$\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t}\odot\frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$可以描述时间步$t$时移动网络流量的高阶动态变化，通过引入 n 阶状态变化，值导数 GRU 算法能够有效地捕获移动网络流量的高阶动态变化信息。更进一步，累计状态变化$\sum\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t}\odot\frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$可以同时描述时间步 t 时移动网络流量的低阶和高阶动态变化，通过引入累计状态变化，值导数 GRU 算法可以同时捕获移动网络流量的低阶和高阶动态变化信息。

在值导数 GRU 隐层单元中，时间步 t 时更新门 u_t 和重置门 r_t 的计算式分别为

$$u_t=\mathrm{sigmoid}\left(\sum_{n=1}^{N}\boldsymbol{W}_{su}^{(n)}\left(\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t}\odot\frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}\right)+\boldsymbol{W}_{xu}x_t+b_u\right) \tag{5}$$

$$r_t=\mathrm{sigmoid}\left(\sum_{n=1}^{N}\boldsymbol{W}_{sr}^{(n)}\left(\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t}\odot\frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}\right)+\boldsymbol{W}_{xr}x_t+b_r\right) \tag{6}$$

更新门 u_t 和重置门 r_t 均选取函数 sigmoid 作为激活函数，输出一个[0,1]内的值，用于表示更新门 u_t 和重置门 r_t 的激活程度。也就是说，当函数 sigmoid 输出 0 时，更新门 u_t 和重置门 r_t 不允许任何移动网络流量信息通过；当函数 sigmoid 输出 1 时，更新门 u_t 和重置门 r_t 允许所有移动网络流量信息通过；当函数 sigmoid 输出一个(0,1)内的值时，更新门 u_t 和重置门 r_t 选择记忆一部分移动网络流量信息，遗忘另一部分移动网络流量信息。

在时间步 t 时，隐层状态 s_t 的计算式为

$$s_t=u_t\odot s_{t-1}+(1-u_t)\odot c_t \tag{7}$$

其中，c_t 表示时间步 t 的候选隐层状态，其计算式为

$$c_t=\tanh(\boldsymbol{W}_{sc}(r_t\odot s_{t-1})+\boldsymbol{W}_{xc}x_t+b_c) \tag{8}$$

重置门 r_t 侧重于保留上一隐层状态 s_{t-1}，更新门 u_t 侧重于忽略上一隐层状态 s_{t-1} 以及保留候选隐层状态 c_t，更新门 u_t 和重置门 r_t 共同决定隐层状态 s_t。

在式(5)~式(8)中，$\boldsymbol{W}_{xu}$、$\boldsymbol{W}_{xr}$、$\boldsymbol{W}_{xc}$ 分别表示输入单元到更新门、重置门以及隐层单元的权重矩阵，$\boldsymbol{W}_{su}$、$\boldsymbol{W}_{sr}$、$\boldsymbol{W}_{sc}$ 分别表示隐藏单元到更新门、重置门以及隐层单元的权重矩阵，b_u、b_r、b_c 分别表示更新门、重置门以及隐层单元的偏置。

此外，在式(5)和式(6)中，本文分别设置 n 个矩阵 $\boldsymbol{W}^{(1)}{}_{xu},\boldsymbol{W}^{(2)}{}_{xu},\cdots,\boldsymbol{W}^{(n)}{}_{xu}$ 和 $\boldsymbol{W}^{(1)}{}_{xr},\boldsymbol{W}^{(2)}{}_{xr},\cdots,\boldsymbol{W}^{(n)}{}_{xr}$ 用于控制一阶状态变化 $s_{t-1}\odot\frac{\mathrm{d}s_{t-1}}{\mathrm{d}t}$，二阶状态变化 $\frac{\mathrm{d}s_{t-1}}{\mathrm{d}t}\odot\frac{\mathrm{d}^2s_{t-1}}{\mathrm{d}^2t},\cdots$，$n$ 阶状态变化 $\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t}\odot\frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$ 的权重。在值导数 GRU 算法的训练过程中，权重矩阵 $\boldsymbol{W}^{(1)}{}_{x*},\boldsymbol{W}^{(2)}{}_{x*},\cdots,\boldsymbol{W}^{(n)}{}_{x*}$ 随之动态更新，进而动态控制更新门 u_t 和重置门 r_t 的激活程度，保证值导数 GRU 算法能够动态捕获移动网络异常流量的低阶或高阶动态变化信息。通过捕获这些不同的累计状态变化，值导数 GRU 算法能够准确地判定相应的移动网络异常流量。综上所述，值导数 GRU 算法正是基于动态更新的权重矩阵 $\boldsymbol{W}^{(1)}{}_{x*}$，$\boldsymbol{W}^{(2)}{}_{x*},\cdots,\boldsymbol{W}^{(n)}{}_{x*}$ 和不尽相同的累计状态变化 $\sum\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t}\odot\frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$ 完成移动恶意软件流量检测。

3.2　池化层

由于网络入侵攻击的行为与正常网络行为不同，而且最大的不同之处在于网络攻击的某些关键步骤。例如，扫描蠕虫首先通过扫描网络发现漏洞主机，然后进行漏洞攻击。因此，这些关键信息能够作为一个核心因素来区分正常的移动网络流量以及不同类型的攻击。为了捕获移动网络异常流量的关键信息，本文添加池化层来调节值导数 GRU 算法。

池化层结构如图 3 所示。其中，左侧区域表示值导数 GRU 隐层单元结构，中间区域表示隐层单元的输出，右侧区域表示池化层。对于时间步 t 而言，与之关联的值导数 GRU 隐层单元的输出向量完全输入池化层。基于值导数 GRU 算法的池化层使用每个时间步 t 的输出向量的总体特征代替最后一个时间步的输出向量的局部特征。

通过保留所有时间步的输出向量对应位置的最大元素，池化层可以获得最佳匹配结果，本文选取 max 函数作为池化层函数。池化层向量计算式为

$$\boldsymbol{V}=\begin{bmatrix}v_1\\v_2\\\vdots\\v_s\end{bmatrix}=\begin{bmatrix}\max\left(s_{11},\cdots,s_{t1},\cdots,s_{T1}\right)\\\max\left(s_{12},\cdots,s_{t2},\cdots,s_{T2}\right)\\\vdots\\\max\left(s_{1s},\cdots,s_{ts},\cdots,s_{Ts}\right)\end{bmatrix} \tag{9}$$

其中，v_s 表示所有时间步的输出向量的第 s 个位置的最大元素，s_{ts} 表示时间步 t 的输出向量的第 s 个位置的元素。

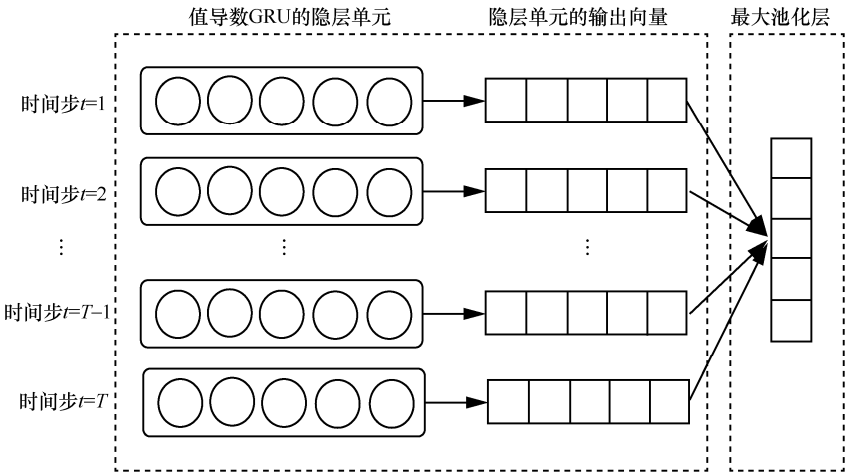


图 3 池化层结构

3.3 值导数 GRU 的训练过程

累计状态变化 $\sum \frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t} \odot \frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$ 作为值导数 GRU 模型的核心内容，其计算式为

$$\sum_{n=1}^{N} \frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t} \odot \frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t} = s_{t-1} \odot \frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} + \frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} \odot \frac{\mathrm{d}^{2}s_{t-1}}{\mathrm{d}^{2}t} + \cdots + \frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t} \odot \frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t} \tag{10}$$

下面仅以一阶状态变化 $s_{t-1} \odot \frac{\mathrm{d}s_{t-1}}{\mathrm{d}t}$ 和二阶状态变化 $\frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} \odot \frac{\mathrm{d}^{2}s_{t-1}}{\mathrm{d}^{2}t}$ 为例，说明累计状态变化 $\sum \frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t} \odot \frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$ 的计算方法。由于高阶状态变化 $\frac{\mathrm{d}^{n-1}s_{t-1}}{\mathrm{d}^{n-1}t} \odot \frac{\mathrm{d}^{n}s_{t-1}}{\mathrm{d}^{n}t}$ 的计算算法类似，本文不再赘述。

对于连续问题而言，微分能够准确地描述单变量函数（本文仅仅考虑隐层状态这一个因素）的函数值 y 随变量 x 的变化而变化的关系，导函数是 $\frac{\mathrm{d}y}{\mathrm{d}x}$。然而，计算机硬件的物理特性决定了计算机无法解决连续问题，只能够处理离散数据。因此，本文借助微分的离散形式，即差分来计算一阶状态变化和二阶状态变化，具体计算式为

$$s_{t-1} \odot \frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} = s_{t-1} \odot \frac{\Delta s_{t-1}}{\Delta t} = s_{t-1} \odot \frac{s_{t-1} - s_{t-2}}{\Delta t} \tag{11}$$

$$\frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} \odot \frac{d^{2}s_{t-1}}{d^{2}t} = \frac{\Delta s_{t-1}}{\Delta t} \odot \frac{\Delta s_{t-1} - \Delta s_{t-2}}{\Delta t^{2}} = \frac{s_{t-1} - s_{t-2}}{\Delta t} \odot \frac{s_{t-1} - 2s_{t-2} + s_{t-3}}{\Delta t^{2}} \tag{12}$$

通过把时间与时间步联系起来，GRU 算法已经成功应用在计算机视觉、自然语言处理等领域。例如，对于视频分类任务而言，GRU 算法通常将视频的一帧与一个时间步关联。此时，相邻时间步之间存在等长时间间隔，即相邻帧之间的时间差值。对于基于值导数 GRU 的移动恶意软件流量检测算法而言，本文将移动网络流量数据分组的间隔时间与时间步关联起来，此时相邻时间步之间存在变长时间间隔，即相邻移动网络流量数据分组之间的时间差值。因此，一阶状态变化和二阶状态变化的具体计算式为

$$s_{t-1} \odot \frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} = s_{t-1} \odot \frac{\Delta s_{t-1}}{\Delta t_1} = s_{t-1} \odot \frac{s_{t-1} - s_{t-2}}{\Delta t_1} \tag{13}$$

$$\frac{\mathrm{d}s_{t-1}}{\mathrm{d}t} \odot \frac{d^{2}s_{t-1}}{d^{2}t} = \frac{\Delta s_{t-1}}{\Delta t_1} \odot \frac{\Delta s_{t-1} - \Delta s_{t-2}}{\Delta t_1 \Delta t_2} = \frac{s_{t-1} - s_{t-2}}{\Delta t_1} \odot \frac{s_{t-1} - 2s_{t-2} + s_{t-3}}{\Delta t_1 \Delta t_2} \tag{14}$$

其中，Δt_1 表示 s_1、s_2 代表的移动网络流量数据分组的时间差值，Δt_2 表示 s_2、s_3 代表的移动网络流量的时间差值。

在值导数 GRU 训练过程中，本文采用信息论中的交叉熵来刻画预测向量 $\hat{\boldsymbol{y}}$ 和真实向量 $\boldsymbol{y}$ 之间的距离。交叉熵代价函数可以量化预测向量 $\hat{\boldsymbol{y}}$ 和真实向量 $\boldsymbol{y}$ 之间偏差，它是一个非负实值函数，代价函数值越小，值导数 GRU 算法的稳健性就越好。交叉熵代价函数表示为

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,i} \log \hat{y}_{n,i} \tag{15}$$

其中，N 表示样本数目。本文选择截断 BPTT 算法

对于 GRU 算法进行训练：如果值导数 GRU 算法样本预测值与样本实际值之间的误差越大，那么在随时间反向传播的过程中，各种参数调整的幅度就要更大，从而使训练更快收敛。训练过程如算法 1 所示。

算法 1　值导数 GRU 算法的训练过程

输入　data, label

from t=1 to t=T:

1) 执行式(10)、式(13)和式(14)，计算累计状态变化；

2) 执行式(5)，计算更新门 u_t；

3) 执行式(6)，计算重置门 r_t；

4) 执行式(7)和式(8)，计算隐层状态 s_t；

5) output = max_pooling(s_1,⋯,s_T)

6) predict = softmax(output)

7) loss = cross entropy(predict, label)，交叉熵计算式为式(15)

8) gradient descent(loss)

抽象而言，GRU 和值导数 GRU 均用于模拟人脑记忆细胞。人脑拥有数以亿计的神经元细胞，一部分神经元细胞拥有记忆静态实体的功能，另一部分神经元细胞拥有记忆动态实体的功能。相比于 GRU 模拟记忆静态实体的神经元细胞而言，值导数 GRU 主要用于模拟记忆动态实体的神经元细胞。因此，值导数 GRU 能够有效地捕获移动网络异常流量的动态变化信息。

4　实验与分析

为了准确评估值导数 GRU 算法捕获移动恶意软件流量的动态变化信息和关键信息的能力，本文基于深度学习框架 TensorFlow 实现值导数 GRU 算法，并且选取 CICAndMal2017 数据集进行仿真实验。此外，本文将 CICAndMal2017 数据集划分为训练集、验证集和测试集 3 个独立数据集，并且进行十折交叉验证。

CICAndMal2017 数据集分为移动恶意软件流量和移动良性软件流量 2 类，其中移动恶意软件流量包括 42 种不同的恶意软件流量。为了准确地评估值导数 GRU 算法检测移动恶意软件流量的能力，本文进行二分类的实验。在实验中，移动恶意软件流量（包括 Dowgin，Ewind，⋯，Zsone 共 42 种）的具体行为是“abnormal”，移动良性软件流量（Benign）的具体行为是“normal”。此外，为了准确评估值导数 GRU 算法检测未知移动恶意软件流量的能力，本文进行的算法仿真实验包括已知移动恶意软件流量检测和未知移动恶意软件流量检测。对于已知移动恶意软件流量检测实验而言，训练集、验证集和测试集均拥有移动恶意软件流量（42 种移动恶意软件流量）和移动良性软件流量；对于未知移动恶意软件流量检测实验而言，训练集、验证集拥有随机的 32 种移动恶意软件流量和移动良性软件流量，测试集拥有 42 种移动恶意软件流量和移动良性软件流量。此外，本文分别独立地进行累计状态变化、隐藏层以及池化层的实验，以此验证这 3 个因素对于值导数 GRU 算法性能的影响。

通过引入准确率 Accuracy 这个衡量指标，本文可以定量地评估值导数 GRU 算法检测移动恶意软件流量的能力。准确率的计算式为

$$\text{Accuracy}=\frac{\text{TP+TN}}{\text{TP+FN+FP+TN}} \tag{16}$$

其中，TP 表示正确检测的移动恶意软件流量的数目，TN 表示正确检测的移动良性软件流量的数目，FN 表示错误检测的移动恶意软件流量的数目，FP 表示错误检测的移动良性软件流量的数目。

4.1　累计状态变化

本节主要研究累计状态变化对于值导数 GRU 算法性能的影响。在单层隐藏层和忽略池化层的条件下，分别对一阶、二阶和三阶累计状态变化的值导数 GRU 算法进行相关实验。

4.1.1　已知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测已知移动恶意软件流量的能力，实验结果如表 1 所示。对于二分类实验而言，RNN 的测试集准确率大约是 86%，LSTM、GRU 的测试集准确率大约是 91%，值导数 GRU 算法的测试集准确率超过 95%。此外，对于值导数 GRU 算法本身而言，一阶累计状态变化的值导数 GRU 算法的测试集准确率分别是 95.46%，二阶累计状态变化的值导数 GRU 算法的测试集准确率分别是 95.91%，三阶累计状态变化的值导数 GRU 算法的测试集准确率分别是 96.89%。因此，对于已知移动恶意软件流量检测而言，值导数 GRU 算法相比于 RNN、LSTM、GRU 拥有较高的验证集和测试集准确率，同时三阶累计状态变化的值导数 GRU 算法拥有最高的验证集和测试集准确率。

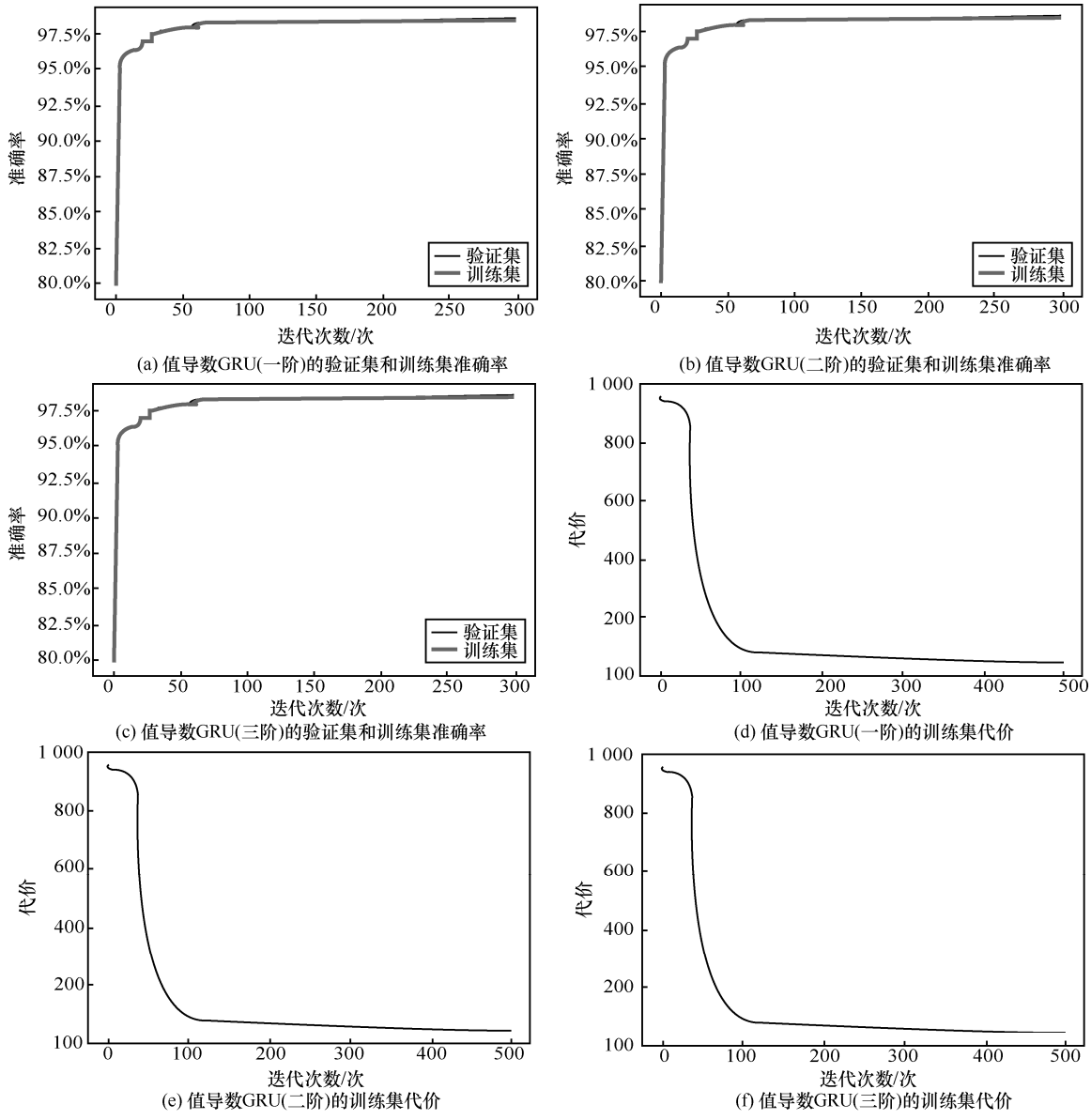
表 1　已知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	87.17%	86.14%
LSTM	91.21 %	91.23%
GRU	91.17%	91.12%
值导数 GRU(一阶)	95.47%	95.46%
值导数 GRU(二阶)	95.78%	95.91%
值导数 GRU(三阶)	**96.79%**	**96.89%**

已知移动恶意软件流量检测实验的训练结果如图 4 所示。其中，图 4 中的横坐标表示模型训练的迭代次数，图 4(a)、图 4(b)、图 4(c)分别是二分类实验的一阶、二阶、三阶累计状态变化的值导数 GRU 算法的验证集和训练集准确率曲线；图 4(d)、图 4(e)、图 4(f)分别是相应的训练集代价曲线。对于二分类实验而言，一阶、二阶、三阶累计状态变化的值导数 GRU 算法的初始验证集和训练集的准确率均是 80%，初始训练集代价均是 1 000。经过大约 100 次迭代之后，算法的验证集和训练集的准确率上升到 97.5%以上，训练集代价下降至 150。经过大约 500 次迭代之后，算法趋于稳定。

4.1.2　未知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测未知移动恶意软件流量的能力，实验结果如



(a) 值导数GRU(一阶)的验证集和训练集准确率
(b) 值导数GRU(二阶)的验证集和训练集准确率
(c) 值导数GRU(三阶)的验证集和训练集准确率
(d) 值导数GRU(一阶)的训练集代价
(e) 值导数GRU(二阶)的训练集代价
(f) 值导数GRU(三阶)的训练集代价

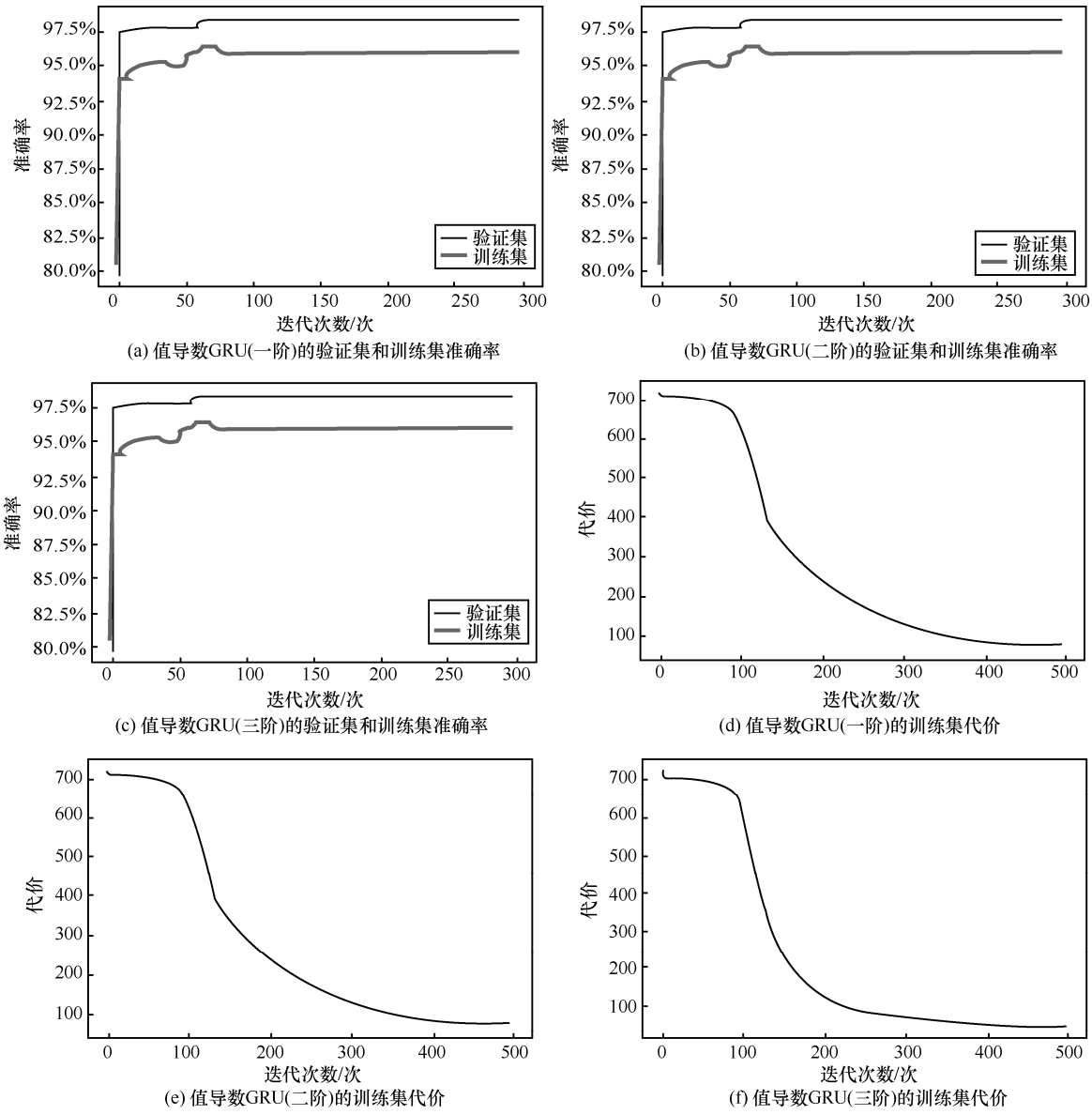
图 4　已知移动恶意软件流量检测的训练结果

表 2 所示。RNN 的测试集准确率是 79.51%，LSTM 的测试集准确率是 90.70%，GRU 的测试集准确率是 90.07%，值导数 GRU 算法的测试集准确率超过 94%。此外，对于值导数 GRU 算法本身而言，一阶累计状态变化的值导数 GRU 算法的测试集准确率是 94.26%，二阶累计状态变化的值导数 GRU 算法的测试集准确率是 95.73%，三阶累计状态变化的值导数 GRU 算法的测试集准确率是 96.03%。因此，对于未知移动恶意软件流量检测而言，值导数 GRU 算法相比于 RNN、LSTM、GRU 拥有较高的验证集和测试集准确率，同时三阶累计状态变化的值导数 GRU 算法拥有最高的验证集和测试集准确率。

表 2　未知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	81.23%	79.51%
LSTM	—	90.70%
GRU	—	90.07%
值导数 GRU(一阶)	96.64%	94.26%
值导数 GRU(二阶)	96.83%	95.74%
值导数 GRU(三阶)	**96.87%**	**96.03%**

未知移动恶意软件流量检测实验的训练结果如图 5 所示。其中，图 5 中的横坐标表示模型训练的迭代次数。一阶、二阶、三阶累计状态变化的值导数 GRU 算法的初始验证集和训练集的准确率均



(a) 值导数GRU(一阶)的验证集和训练集准确率
(b) 值导数GRU(二阶)的验证集和训练集准确率
(c) 值导数GRU(三阶)的验证集和训练集准确率
(d) 值导数GRU(一阶)的训练集代价
(e) 值导数GRU(二阶)的训练集代价
(f) 值导数GRU(三阶)的训练集代价

图 5　未知移动恶意软件流量检测的训练结果

是 83%，初始训练集代价均是 700。经过大约 150 个迭代之后，算法的验证集准确率上升到 96%，训练集代价均下降至 350。经过大约 400 个迭代之后，算法趋于稳定。

由于下文实验的验证集、测试集准确率曲线与训练集代价曲线和上述实验曲线相似，鉴于篇幅所限不再赘述。

4.2 隐藏层

本节主要研究隐藏层对于值导数 GRU 算法性能的影响。在一阶累计状态变化和忽略池化层的条件下，分别对一层、三层和五层隐藏层的值导数 GRU 算法进行相关实验。

4.2.1 已知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测已知移动恶意软件的能力，实验结果如表 3 所示。对于值导数 GRU 算法本身而言，一层隐藏层的值导数 GRU 算法的测试集准确率是 95.46%，三层隐藏层的值导数 GRU 算法的测试集准确率是 96.53%，五层隐藏层的值导数 GRU 算法的测试集准确率是 96.62%。因此，对于已知移动恶意软件流量检测而言，值导数 GRU 算法相比于 RNN、LSTM、GRU 拥有较高的验证集和测试集准确率，同时五层隐藏层的值导数 GRU 算法拥有最高的验证集和测试集准确率。

表 3 已知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	87.17%	86.14%
LSTM	91.21%	91.23%
GRU	91.17%	91.12%
值导数 GRU(一层)	95.47%	95.46%
值导数 GRU(三层)	96.50%	96.53%
值导数 GRU(五层)	**96.55%**	**96.62%**

4.2.2 未知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测未知移动恶意软件流量的能力，实验结果如表 4 所示。对于值导数 GRU 算法本身而言，一层隐藏层的值导数 GRU 算法的测试集准确率是 94.26%，三层隐藏层的值导数 GRU 算法的测试集准确率是 95.76%，五层隐藏层的值导数 GRU 算法的测试集准确率是 96.13%。因此，对于未知移动恶意软件流量检测而言，值导数 GRU 算法相比于 RNN、LSTM、GRU 拥有较高的验证集和测试集准确率，同时五层隐藏层的值导数 GRU 算法拥有最高的验证集和测试集准确率。

表 4 未知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	81.23%	79.51%
LSTM	—	90.70%
GRU	—	90.07%
值导数 GRU(一层)	96.64%	94.26%
值导数 GRU(三层)	97.84%	95.76%
值导数 GRU(五层)	**98.20%**	**96.13%**

4.3 池化层

本节主要研究池化层对于值导数 GRU 算法性能的影响。在一阶累计状态变化和单层隐藏层的条件下，分别对于增设、忽略池化层的值导数 GRU 算法进行相关实验。

4.3.1 已知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测已知移动恶意软件流量的能力，实验结果如表 5 所示。对于值导数 GRU 算法本身而言，增设池化层的值导数 GRU 算法的测试集准确率是 94.28%，忽略池化层的值导数 GRU 算法测试集准确率是 95.46%。因此，对于已知移动恶意软件流量检测而言，值导数 GRU 算法相比于 RNN、LSTM、GRU 拥有较高的验证集和测试集准确率，但是，池化层不能提高值导数 GRU 算法检测已知移动恶意软件流量的准确率。

表 5 已知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	87.17%	86.14%
LSTM	91.21%	91.23%
GRU	91.17%	91.12%
值导数 GRU(无池化层)	**95.47%**	**95.46%**
值导数 GRU(有池化层)	95.13%	94.28%

4.3.2 未知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测未知移动恶意软件流量的能力，实验结果如表 6 所示。对于值导数 GRU 算法本身而言，增设池化层的值导数 GRU 算法的测试集准确率是 95.34%，忽略池化层的值导数 GRU 算法的测试集准确率是 94.26%。因此，对于未知移动恶意软件流量检测而言，值导数 GRU 算法相比于 RNN、LSTM、GRU 拥有较高的验证集和测试集准确率，同时增设

池化层的值导数 GRU 算法拥有最高的验证集和测试集准确率。

表 6　未知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	81.23%	79.51%
LSTM	—	90.70%
GRU	—	90.07%
值导数 GRU(无池化层)	96.64%	94.26%
值导数 GRU(有池化层)	**96.65%**	**95.34%**

4.4　综合分析

本节主要研究综合累计状态变化、隐藏层和池化层 3 个因素对于值导数 GRU 算法性能的影响。在三阶累计状态变化、五层隐藏层和增设池化层的条件下，本文对值导数 GRU 算法进行已知和未知移动恶意软件流量检测的相关实验。

4.4.1　已知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法对于已知移动恶意软件流量的检测能力，实验结果如表 7 所示。对于值导数 GRU 算法本身而言，三阶值导数 GRU 算法的测试集准确率是 96.89%，五层隐藏层的值导数 GRU 算法测试集准确率是 96.62%，增设池化层的值导数 GRU 算法测试集准确率是 94.28%，忽略池化层的值导数 GRU 算法测试集准确率是 95.46%，三阶、五层隐藏层和增设池化层条件下的值导数 GRU 算法测试集准确率是 96.93%，三阶、五层隐藏层和忽略池化层条件下的值导数 GRU 算法测试集准确率是 97.28%。因此，对于已知移动恶意软件流量检测而言，三阶、五层隐藏层和忽略池化层条件下的值导数 GRU 算法拥有最高的验证集和测试集准确率。

4.4.2　未知移动恶意软件流量检测

本节主要通过二分类实验评估值导数 GRU 算法检测未知移动恶意软件流量的能力，实验结果如表 8 所示。对于值导数 GRU 算法本身而言，三阶值导数 GRU 算法的测试集准确率是 96.03%，五层隐藏层的值导数 GRU 算法的测试集准确率是 96.13%，增设池化层的值导数 GRU 算法的测试集准确率是 95.34%，忽略池化层的值导数 GRU 算法的测试集准确率是 94.26%，三阶、五层隐藏层和增设池化层条件下的值导数 GRU 算法测试集准确率是 96.38%，三阶、五层隐藏层和忽略池化层条件下的值导数 GRU 算法测试集准确率是 96.14%。因此，对于未知移动恶意软件流量检测而言，三阶、五层隐藏层和增设池化层条件下的值导数 GRU 算法拥有最高的测试集准确率。

表 7　已知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	87.17%	86.14%
LSTM	91.21%	91.23%
GRU	91.17%	91.12%
值导数 GRU(三阶)	96.79%	96.89%
值导数 GRU(五层)	96.55%	96.62%
值导数 GRU(无池化层)	95.47%	95.46%
值导数 GRU(有池化层)	95.13%	94.28%
值导数 GRU(三阶、五层，有池化层)	96.80%	96.93%
值导数 GRU(三阶、五层，无池化层)	**97.13%**	**97.28%**

表 8　未知移动恶意软件流量检测准确率

算法	验证集	测试集
RNN	81.23%	79.51%
LSTM	—	90.70%
GRU	—	90.07%
值导数 GRU(三阶)	96.87%	96.03%
值导数 GRU(五层)	98.02%	96.13%
值导数 GRU(有池化层)	96.65%	95.34%
值导数 GRU(无池化层)	96.64%	94.26%
值导数 GRU(三阶，五层，有池化层)	**96.98%**	**96.38%**
值导数 GRU(三阶，五层，无池化层)	96.90%	96.14%

4.5　讨论

综合上述实验结果，本文讨论关于累计状态变化、隐藏层以及池化层对于值导数 GRU 算法性能的影响如下。

1) 对于累计状态变化而言，三阶累计状态变化的值导数 GRU 算法相比于一阶、二阶而言拥有较高的验证集和测试集准确率。此外，实验结果显示更高阶累计状态变化的值导数 GRU 算法拥有类似结论，由于篇幅限制，本文不再赘述。因此，高阶累计状态变化的值导数 GRU 算法能够更为有效地检测移动恶意软件流量。

2) 对于隐藏层而言，五层隐藏层的值导数 GRU 算法相比于一层、三层而言拥有较高的验证集和测试集准确率。同样，实验结果显示更多层隐藏层的值导数 GRU 算法拥有相似结论，由于篇

幅限制本文不再赘述。因此，多层隐藏层的值导数 GRU 算法可以更为有效地检测移动恶意软件流量。

3) 对于池化层而言，增设池化层的值导数 GRU 算法拥有较高的未知移动恶意软件流量检测的准确率，但是拥有较低的已知移动恶意软件流量检测的准确率。因此，增设池化层的值导数 GRU 算法仅能够更有效地检测未知移动恶意软件流量。

4) 综合累计状态变化、隐藏层以及池化层来看，在检测已知移动恶意软件流量的时候，三阶、五层隐藏层和忽略池化层条件下的值导数 GRU 算法拥有最高的验证集和测试集准确率；在检测未知移动恶意软件流量的时候，三阶、五层隐藏层和增设池化层条件下的值导数 GRU 算法拥有最高的验证集和测试集准确率。因此，综合三阶、五层隐藏层能够更为有效地检测已知移动恶意软件流量，综合三阶、五层隐藏层和增设池化层能够更为有效的检测未知移动恶意软件流量，

综上所述，基于值导数 GRU 的移动恶意软件流量检测算法相比于基于 RNN、LSTM、GRU 的移动恶意软件流量检测算法拥有较高的验证集和测试集准确率。也就是说，值导数 GRU 算法相比于 RNN、LSTM、GRU 算法能够更有效地检测移动恶意软件流量。

5 结束语

本文提出一种基于值导数 GRU 的移动恶意软件流量检测方法，旨在解决基于 RNN 的移动恶意软件流量检测算法难以捕获移动网络异常流量的动态变化和关键信息的问题。值导数 GRU 算法通过引入“累计状态变化”概念可以定量地描述移动网络异常流量的低阶和高阶动态变化信息。此外，通过增设池化层使值导数 GRU 算法获取流量的重要信息。最后，本文选取 CICAndMal2017 数据集进行了一系列仿真实验，并且研究累计状态变化、隐藏层以及池化层 3 个因素对于值导数 GRU 算法性能的影响。实验结果表明，基于值导数 GRU 的移动恶意软件流量检测算法能够更为有效地检测移动网络异常流量。

参考文献：

[1] HE D, CHAN S, GUIZANI M. Mobile application security: malware threats and defenses[J]. IEEE Wireless Communications, 2015, 22(1): 138-144.

[2] 冯勇, 张丽颖, 顾兆旭, 等. 面向高校多源异构数据环境的元数据集成方法[J]. 辽宁大学学报(自然科学版), 2019, 46(2): 135-141.
FENG Y, ZHANG L Y, GU Z X, et al. A metadata integration method for multi-source heterogeneous data environment in universities[J]. Journal of Liaoning University: Natural Science, 2019, 46(2): 135-141.

[3] SAYYAR S. Enhanced TWOACK based AODV protocol for intrusion detection system[C]//International Conference on Computing, Mathematics and Engineering Technologies.2018: 1-4.

[4] MIMURA M, TANAKA H. Long-term performance of a generic intrusion detection method using Doc2vec[C]//2017 Fifth International Symposium on Computing and Networking (CANDAR). 2017: 456-462.

[5] KHATRI V, ABENDROTH J. Mobile guard demo: network based malware detection[C]//IEEE International Conference on Trust, Security and Privacy in Computing and Communications. 2015: 1177-1179.

[6] ADEEL M, TOKARCHUK L N. Analysis of mobile P2P malware detection framework through cabir & commwarrior families[C]//IEEE Third International Conference on Privacy, Security, Risk and Trust. 2011: 1335-1343.

[7] MOGHADDAM S H. Sensitivity analysis of static features for Android malware detection[C]//22nd Iranian Conference on Electrical Engineering. 2014: 920-924.

[8] TRIPP O, PISTOIA M, FERRARA P, et al. Pinpointing mobile malware using code analysis[C]//IEEE/ACM International Conference on Software Engineering and Systems. 2016: 275-276.

[9] NGUYEN TRI-HAI, YOO M. A behavior-based mobile malware detection model in software-defined networking[C]//International Conference on Information Science and Communications Technologies. 2017: 1-3.

[10] LI D F, WANG Z G, XUE Y B. Fine-grained Android malware detection based on deep learning[C]//IEEE Conference on Communications and Network Security. 2018: 1-2.

[11] YUAN Z L, LU Y Q, XUE Y B. Droiddetector: Android malware characterization and detection using deep learning[J]. Tsinghua Science and Technology, 2016, 22(1): 114-123.

[12] KIM T G, KANG B J, RHO M, et al. A multimodal deep learning method for Android malware detection using various features[J]. IEEE Transactions on Information Forensics and Security, 2019, 14(3): 773-788.

[13] SU X, ZHANG D F, LI W J, et al. A deep learning approach to Android malware feature learning and detection[C]//IEEE International Conference on Trust, Security and Privacy in Computing and Communications. 2016: 244-251.

[14] CHEN Z X, YAN Q B, HAN H B. Machine learning based mobile malware detection using highly imbalanced network traffic[C]//IEEE International Conference on Computational Science and Engineering

(CSE) and IEEE International Conference on Embedded and Ubiquitous Computing. 2017: 588-595.

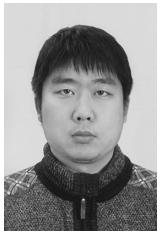
[15] ZHANG L, FAN X P. A fusion financial prediction strategy based on RNN and representative pattern discovery[C]//International Conference on Parallel and Distributed Computing, Applications and Technologies. 2017: 92-97.

[16] BENGIO Y. Learning long-term dependencies with gradient descent is difficult[J]. IEEE Transactions on Neural Networks, 2002, 5(2): 157-166.

[17] BEAUFAYS S S. Long short-term memory recurrent neural network architectures for large scale acoustic modeling[J]. Computer Science, 2014, 15(3): 338-342.

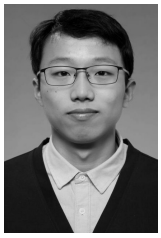
[18] CHO K, MERRIENBOER VAN B, GULCEHRE C. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. Computer Science, 2014, 12(1): 236-248.

[作者简介]



周翰逊（1981− ），男，辽宁沈阳人，博士，辽宁大学副教授、硕士生导师，主要研究方向为网络安全、图像处理、深度学习、恶意代码分析。

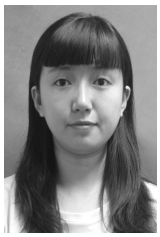
陈晨（1995− ），女，辽宁鞍山人，辽宁大学硕士生，主要研究方向为网络安全、深度学习、恶意代码分析。



冯润泽（1994− ），男，山东临沂人，辽宁大学硕士生，主要研究方向为网络安全、深度学习、恶意代码分析。

熊俊坤（1996− ），男，湖北天门人，辽宁大学硕士生，主要研究方向为深度学习、网络安全。

潘宏（1979– ），男，辽宁盘锦人，博士，辽宁大学副教授，主要研究方向为数字经济、大数据、区块链、网络安全、深度学习等。



郭薇（1983– ），女，辽宁沈阳人，博士，沈阳航空航天大学副教授，主要研究方向为网络安全和图像处理。