车载通信 21（2020)100198

ScienceDirect提供的目录列表

车载通讯

www.elsevier.com/locate/vehcom

使用深度卷积神经网络的车载网络入侵检测

Hyun Min Song a, Jiyoung Woo b, Huy Kang Kim a,*

[A] 高丽大学信息安全研究生院,韩国首尔b
韩国牙山顺天乡大学

文章信息

文章历史：
2018 年 12 月 30 日收到
2019 年 7 月 31 日以修订形式收到
2019 年 9 月 23 日接受
2019 年 10 月 3 日在线发售

关键词：
车载网络
控制器区域网络 (CAN)
入侵检测
卷积神经网络 (CNN)

抽象的

现代车辆中电子设备的实施导致针对车辆网络的攻击增加;因此,攻击检测模型引起了汽车行业及其研究人员的关注。车辆网络安全是一个紧迫而重大的问题,因为车辆的故障会直接影响到人和道路的安全。用作车载网络事实上的标准的控制器局域网 (CAN) 没有足够的安全功能,例如消息加密和发件人身份验证,无法保护网络免受网络攻击。在本文中,我们提出了一种基于深度卷积神经网络 (DCNN) 的入侵检测系统 (IDS) 来保护车辆的 CAN 总线。 DCNN 学习网络流量模式并检测恶意流量,无需手动设计的功能。我们设计了针对 CAN 总线的数据流量进行优化的 DCNN 模型,以实现高检测性能,同时减少 Inception-ResNet 模型架构中不必要的复杂性。我们使用我们用真实车辆构建的数据集进行了一项实验研究,以评估我们的检测系统。实验结果表明,与传统的机器学习算法相比,所提出的 IDS 具有显着较低的漏报率和错误率。

一、简介

在过去的几十年中,汽车电子的实施经历了快速增长[1]。这种趋势导致了车辆生态系统的一些变化。线控驱动 (DBW) 技术,例如,在控制系统中使用电子或电气系统,例如油门、制动器和转向,这些传统上使用机械连杆进行控制 [2] 。控制车辆系统的电子设备称为电子控制单元 (ECU)。控制器局域网 (CAN) 是作为消息广播系统开发的,旨在为 ECU 之间提供高效、可靠和经济的通信通道。 CAN 是与本地互连网络 (LIN) 和 FlexRay [3] 一样部署最广泛的车载网络之一。与常见的以太网网络不同,CAN 广播多个包含车辆状态信息的短消息,以保持系统所有 ECU 中数据的一致性[4]。

但是,CAN 没有诸如身份验证和加密之类的安全功能来保护其通信免受网络攻击。研究人员已经证明了车辆网络中的安全漏洞[5]。 CAN 协议中缺乏保护机制允许攻击者通过注入伪造的消息来控制车辆系统。例如,攻击者可以通过注入带有与档位功能相关的特定 CAN 标识符的消息来更改汽车的档位。伪造的消息可以直接通过车载诊断 (OBD-II) 端口注入,也可以通过信息娱乐系统或无线通信系统注入[6]。近年来,在车辆系统中发现了各种易受攻击的特征,例如电动车窗升降器、警示灯、安全气囊[7] 和胎压监测系统 (TPMS) [8]。

有几项研究试图通过添加基于成对对称密钥的数字签名来提高 CAN 协议的安全性[9]、 [10]。然而,数字签名具有较高的通信开销,CAN 的带宽限制在 500 kbps。除了开销之外,现有车辆的所有设备都很难改变。因此,需要研究不会引起通信开销并且可以添加到现有系统的入侵检测系统（IDS）。

在本文中,我们提出了一种使用深度卷积神经网络 (DCNN) 来保护 CAN 总线免受网络攻击的 IDS,

* 通讯作者。
电子邮件地址： cenda@korea.ac.kr （香港金）。

https://doi.org/10.1016/j.vehcom.2019.100198

例如拒绝服务 (DoS) 和欺骗攻击,显着降低漏报率和错误率。

我们创建了一个简单的数据组装模块,称为框架构建器,它允许我们将 CAN 总线的比特流数据直接用于卷积神经网络 (CNN)。框架构建器将 CAN 总线数据转换为适合 DCNN 的网格状结构。通过使用网格数据帧而不是特征向量,基于 CNN 的分类器无需额外的数据预处理即可有效地学习 CAN 交通数据中的时间序列模式。实验结果表明,与传统的机器学习算法相比,我们的 IDS 实现了较高的检测性能。

本文提出了以下贡献:

- 首先,据我们所知,这项工作是第一项将 DCNN 应用于 IDS 以实现车载网络安全的研究。我们采用了最近的 DCNN 架构,称为 Inception ResNet,它在自然图像分类任务中展示了最先进的性能[11]。此外,我们还评估了该网络在车载网络中进行入侵检测的有效性。

- 其次,我们使用真实车辆通过注入和记录 CAN 消息构建了完全标记的车载网络攻击数据集。我们的数据集是公开的,以促进进一步的研究[12]。

- 第三,通过使用我们构建的真实数据集进行的大量实验,我们证明了与传统机器学习算法相比,我们的检测模型显示出更高的检测性能。此外,所提出的数据预处理方法成功地用于车内入侵检测。

本文的其余部分安排如下。第2节介绍了涉及特定 CAN 和基于顺序数据的 IDS 的研究。第 3节和第 4节分别提供了有关车载网络安全和人工神经网络（包括 CNN)的背景材料。在第5 节中,介绍了建议的 IDS 和方法步骤。在第6 节中,介绍了评估指标和实验结果。最后,我们在第 7 节中总结了这项研究。

二、相关工作

在本节中,我们从领域和数据性质的角度介绍了一些研究工作,分别是专门针对 CAN 和时序数据的 IDS。

2.1.控制器局域网（CAN)中的入侵检测模型

车载网络承载着对车辆有效运行至关重要的关键信息;因此,网络的安全漏洞与安全问题密切相关。

一些研究侧重于根据 CAN 协议的特点通过各种方法保护车载网络。每个 CAN ID 都有一个特定的频率,ECU 以该频率定期发送消息。 Miller 和 Valasek 引入了基于频率分布的入侵检测的概念。由于特定 CAN ID 的频率不会波动,他们断言可以通过监控频率水平轻松检测到攻击[13]。同样,Muter 和 Asaj 提出了一种基于熵的异常检测方法[14]。他们定义了 CAN 总线上的熵,并通过将熵与参考集进行比较来检测攻击。相反,拉森等人。提出了一种基于规范的攻击检测方法,如果消息不遵循协议级别或 ECU 行为规范,则将其视为攻击[15]。宋等。介绍了一种基于 CAN 消息发生时间分析的轻量级 IDS [16]。 CAN 消息的周期性特性被用作一个显着特征。

通过监视消息之间异常缩短的间隔的发生,将注入的消息与正常消息区分开来。 Cho 和 Kang [17]使用真实时钟和真实时钟频率之间的偏差来分析 ECU 的行为,并提出了基于时钟的 IDS。上述入侵检测方法仅对某些威胁模型或在某些前提下有效,例如 CAN 消息的周期性、具有特定 CAN ID 的消息的大量注入以及规范违规。最近克服该问题的尝试导致提出了基于深度学习的技术。 Kang 和 Kang [18]使用深度信念网络 (DBN) 结构构建分类器并在模拟数据集上对其进行测试。泰勒等。 [19]使用基于长短期记忆 (LSTM) 网络的模型,并在从真实车辆收集的 CAN 交通日志上对其进行了测试。 DBN 和 LSTM 通常被认为在训练模型时比 CNN 成本更高。徐等。 [20]提出了一种使用生成对抗网络训练异常检测模型的新方法。他们使用正常的 CAN 流量数据训练检测模型,并生成类似于 CAN 流量数据的噪声数据。王等。 [21]提出了一种使用分层时间记忆（HTM)的分布式异常检测系统。作者设计了基于 HTM 算法和对数损失函数的预测器来计算异常分数。在其他研究中,Levi 等人。 [22]试图解决考虑网络流量数据的顺序性质的问题。他们训练了一个隐马尔可夫模型 (HMM) 来学习车辆的正常行为。

2.2.时序数据的入侵检测模型

顺序数据中的异常或入侵检测是重要的研究领域之一。钱多拉等。 [23]介绍了三种用于序列数据异常或入侵检测的方法:基于序列、基于连续子序列和基于模式的方法。基于序列的方法从给定的序列数据集中识别异常序列,而基于连续子序列的方法确定给定长序列中的异常子序列。基于模式频率的方法,也称为模式挖掘,识别测试序列中具有特定出现频率的模式。从另一个角度来看,Xing 等人。 [24]将序列分类方法分为三类。第一种是基于特征的方法,它将序列转换为特征向量;因此,该方法在分类之前需要额外的预处理步骤。第二种是基于距离的方法,它测量序列之间的相似性。这种方法处理大规模数据库的能力是有限的,因为它需要计算测试序列和所有参考序列之间的距离。最后一种是基于模型的方法,它使用 HMM 和朴素贝叶斯 (NB) 等分类模型。基于机器学习的方法属于这一类;此外,它们需要大量的训练数据才能达到高水平的表现。

最流行的应用领域是基于系统调用的入侵检测。霍夫梅尔等人。 [25]通过分析几个 UNIX 程序的系统调用序列介绍了一种入侵检测方法。 Kosoresow 等人。 [26]提出了一种使用通过分析系统调用跟踪获得的计算机活动轨迹的方法。 Lee 和 Stolfo [27]采用数据挖掘技术进行入侵检测。他们在实验中使用了 Sendmail 系统调用数据和网络 TCP 转储数据,并证明了他们的分类器能够检测异常谎言和已知入侵。曲[28]介绍了一种新颖的基于主机的

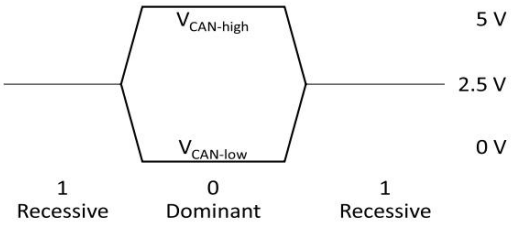

图 1.控制器局域网 (CAN) 总线的位信号逻辑。

采用基于代理的人工免疫系统的 IDS。安装在主机上的代理分析系统调用、应用程序日志、文件系统修改以及其他主机活动和状态。 Shabtai 等人。 [29]提出了一个用于检测机器人设备上恶意软件的框架,名为 Andromaly,它监视来自移动设备的各种事件,然后应用基于机器学习的异常检测器对异常进行分类。同样,网络入侵检测也得到了积极研究[30]。卡西克等人。 [31]提出了一种自适应网络 IDS,可以收集网络流量并检测异常。 Goldenberg 和 Wool [32]提出了一种基于模型的 IDS,用于使用 Modbus 流量的 SCADA 系统。余等。 [33]和 Kwon 等人。 [34]使用 Hurst 参数提出了一种基于自相似性的 IDS,该参数是根据网络流量特征计算得出的。除了入侵检测之外,网络物理系统中的故障检测也被认为是一个重要问题,因为它涉及安全问题[35]。

王等。提出了一种使用 CNN [36] 的恶意流量分类方法。由于网络流量数据被处理为图像并输入到他们的 CNN,因此使用了原始网络流量;因此,不需要手工设计的功能。

三、车载网络安全概述

3.1.控制器区域网络

现代车辆使用称为 CAN 总线网络的总线拓扑网络,这是一种基于消息的广播协议,旨在允许 ECU 相互通信。 CAN 由于其相对较低的成本和可靠性,已被广泛采用并成为车载网络的事实标准。

CAN 的所有节点都通过 CAN-high 和 CAN-low 双绞线连接到总线。图1显示了 CAN 总线的信号逻辑。当 ECU 传输零位（显性)时,CAN 高线的电压为 5 V,CAN 低线的电压为 0 V。相反,当 ECU 传输一位（隐性)时,两条线的电压均为 2.5 V。不同于基于流的协议,例如 TCP 网络,其中数据以非帧流的形式发送,CAN 是一种基于消息的广播协议,其中节点在称为消息的预定义数据帧中发送数据,如图 1 所示。 2.

在CAN总线中,包含RPM、转向角和当前速度等信息的几条消息被广播到整个网络,从而保持整个系统的一致性。

这些消息由 CAN ID 标识。 CAN 报文具有唯一的 11 位或 29 位标识符。基本 ID 字段包含 11 位 ID,扩展 ID 包含剩余的 18 位 ID。

CAN 2.0A 设备只使用基本 ID,而 CAN 2.0B 设备使用两个 ID 字段。 ECU 可以确定是否收到消息

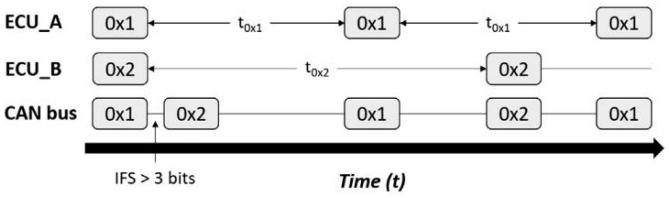

图 3.消息优先级和帧间空间 (IFS) 的概念图。

基于 CAN ID 是否有趣;因此,他们可以过滤不感兴趣的消息。

CAN帧（如图2所示)中各字段的含义如下：

· SOF（1 位）:帧开始(SOF) 位表示新消息的开始。该单个位用于同步总线上的所有节点。 ·基本标识符（11 位）:这是标识符的第一部分,用于标准帧和扩展帧。 · SRR（1 位）:替代远程请求 (SRR) 位固定为

一个并在扩展框架中使用。

· IDE（1 位）:标识符扩展位（IDE)固定为1
并在扩展框架中使用。

·扩展标识符（18 位）:这是标识符的第二部分,仅在扩展帧中使用。 · RTR（1 位）：当需要来自另一个节点的特定信息时使用远程传输请求(RTR)。

标识符指定必须响应的节点。 ·保留位（2 位）:这些是供将来使用的保留位。 · DLC（4 位）:数据长度代码(DLC) 表示数据的字节数。 · Data (64 bits):实际的payload数据,最大可达64位

位。

· CRC（16 位）:循环冗余校验 (CRC) 包含用于错误检测的先前数据的校验和。 · ACK（2 位）:发送器发送一个隐性位（1）;当接收到的消息中没有错误时,其他人会将此位更改为显性位 (0)。

· EOF（7 位）:帧结束（EOF)表示一个帧的结束
当前 CAN 消息。

CAN ID 还用于确定在仲裁阶段具有优先权的 ECU。由于 CAN 总线是一个广播系统,可能会出现多个节点试图同时发送报文而导致冲突的情况。在仲裁阶段,ECU 逐位发送它们的 CAN ID。具有更多显性位的 ECU,即具有更多前导零的 CAN ID,赢得总线并获得发送消息的机会。或者,当 ECU 在其 ID 的当前位为 1 时检测到 CAN 总线上的位为零时,ECU 将停止传输消息。此外,连续消息之间存在强制间隙,称为帧间空间 (IFS)。每条消息通过一个 IFS 与前一帧分开,该 IFS 至少包含三个隐性位。如果在连续的隐性位之后检测到显性位,则将其视为下一条消息的 SOF。此外,为了系统的一致性,即使数据值没有改变,ECU 也会定期广播消息。这导致 ECU 具有自己的消息传输周期,如图3 所示。

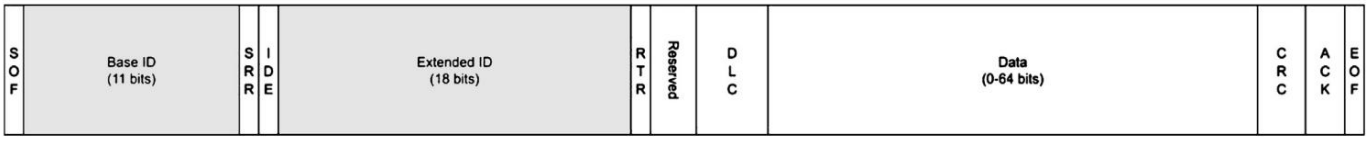

图 2. CAN 2.0B 消息帧的结构。

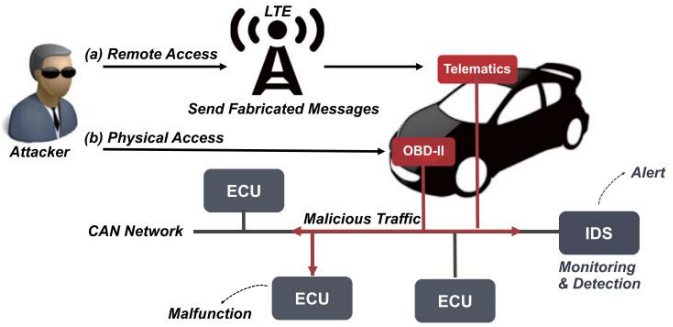

图 4.典型的消息注入攻击场景。攻击者可以通过车辆的车载诊断端口或提供无线通信通道的远程信息处理设备,将伪造的消息注入目标车辆的 CAN 总线。

ECU A在每个时间间隔t1后发送 CAN ID“0x1”的消息， ECUB在每个时间间隔 t2 后发送“0x2”消息。由于第一组消息ECUA （0x1)的优先级高于ECUB （0x2）， ECU A的消息先发送， ECUB的消息在IFS之后立即发送。实际上,具有发动机转速信息的 ECU 每 10 毫秒发送一次包含转速数据的消息。然后,管理仪表组的 ECU 接收有关 RPM 的信息并将其显示在 RPM 仪表上。由于这种周期性,某些顺序模式出现在 CAN 总线上。

尽管 CAN 是一种开放规范,但特定车辆中使用的实际标识符和数据取决于汽车制造商。但是,出于安全原因,制造商不提供此信息。因此,很难将语义特征用于入侵检测。因此,我们利用出现在 CAN 总线上的 CAN ID 的顺序模式来检测外部入侵。

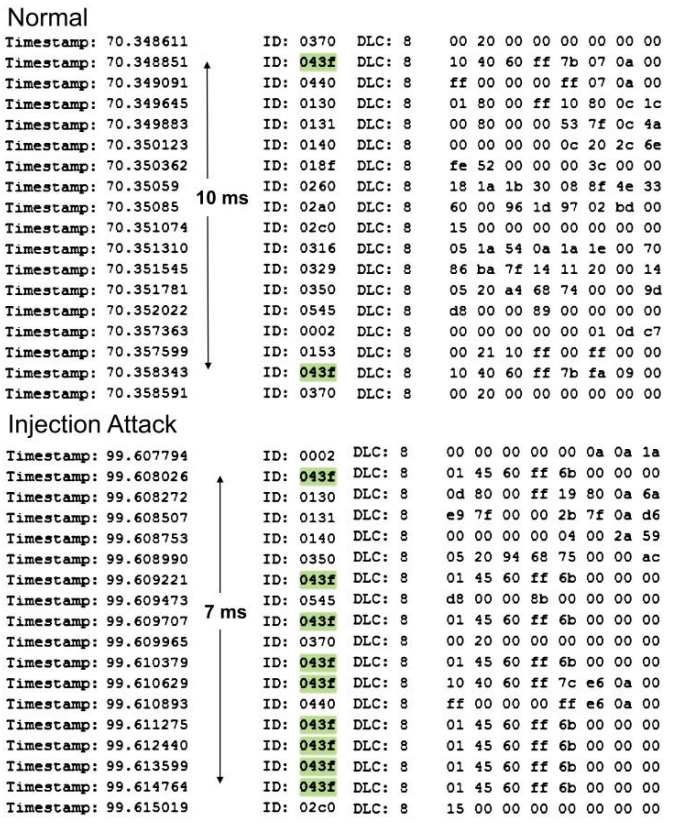

图 5.注入伪造消息的 CAN 流量示例。这些消息的 CAN ID 为“043f”。 “043f”消息的出现仅在 7 毫秒内增加到九条消息。 “043f”消息最初每 10 毫秒传输一次。

3.2.消息注入攻击

CAN协议最突出的安全问题是缺乏认证和加密等安全特性。由于 CAN 是一个基于广播的总线网络,没有身份验证,任何节点都可以连接到总线上,并且可以接收所有消息。因此,攻击者也可以轻松嗅探 CAN 总线数据。这种内部数据的暴露允许攻击者分析目标车辆的 CAN 数据,并设计后续复杂的消息注入攻击来控制目标车辆。

早期的研究发现了各种攻击面,攻击者可以利用这些攻击面将消息注入 CAN 总线[6]。
作者将各种攻击面分为物理访问和无线访问。攻击者可以通过 OBD-II 端口物理访问 CAN 总线,或通过蓝牙、WiFi 或远程信息处理服务（例如 3G 和长期演进 (LTE))远程访问。

这些攻击面可用作目标车辆 CAN 总线的入口点。

图4显示了两种不同的消息注入攻击场景 ios:(a) 远程访问和 (b) 物理访问。在远程访问的情况下,攻击者可以利用汽车制造商（例如,通用汽车的 OnStar、福特的 Sync 和现代的 BlueLink)提供的远程信息处理服务来破坏目标车辆的远程信息处理设备。使用远程信息处理服务的远程访问是远程无线攻击中最关键的部分,因为这些服务通过 3G 和 LTE 通信等蜂窝网络提供广泛的连接。这允许攻击者从远距离远程获得访问权限,而不受物理位置的限制。在物理访问的情况下,攻击者使用的最简单的方法是通过OBD-II端口直接连接访问CAN总线。攻击者可以附加他的微型计算设备,例如 Raspberry Pi 和 Arduino,作为目标车辆 OBD-II 端口的攻击节点。

一旦攻击者获得任何一个 CAN 节点,他就可以通过注入复杂的操纵消息来控制车辆。在 CAN 协议中,连接的节点认为最后收到的消息代表车辆的当前状况。因此,攻击者可以通过在目标 CAN ID 消息之后立即注入经过操纵的消息来欺骗 ECU。例如,当车辆在行驶中并且其变速杆置于“驱动”位置时,处理变速器档位状态的ECU重复发送具有指示“驱动”的值的消息。此时,如果攻击者在原ECU发送的消息后立即注入值为“reverse”的挡位状态对应的CAN ID报文,则其他ECU认为当前挡位状态为“reverse”而非“reverse” “驾驶”。为了成功控制车辆,攻击者必须比原始 ECU 更频繁地传输消息,而原始 ECU 会连续且定期地传输其消息。

图5显示了注入攻击期间的示例网络流量日志。每行包含每条消息的信息,即时间戳、CAN ID、DLC 和由最多 8 个字节组成的有效载荷数据。 “043f”消息之间的原始间隔为 10 毫秒,但在注入攻击期间仅 7 毫秒就记录了 9 个“043f”消息。当注入的消息出现在网络上时,CAN ID 的顺序模式会发生变化。因此,我们利用 CAN ID 序列模式的这种变化来检测消息注入攻击。

在这项工作中,我们假设攻击者已经可以物理或远程访问 CAN 总线,并且拥有对一个或多个 CAN 节点进行消息注入攻击的权限。由于CAN总线是一个基于广播的网络,所以不区分消息的来源； ECU 处理注入的消息

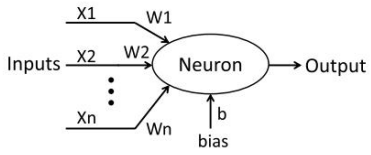

图 6.人工神经元接收特征向量作为输入并输出输入的加权和。

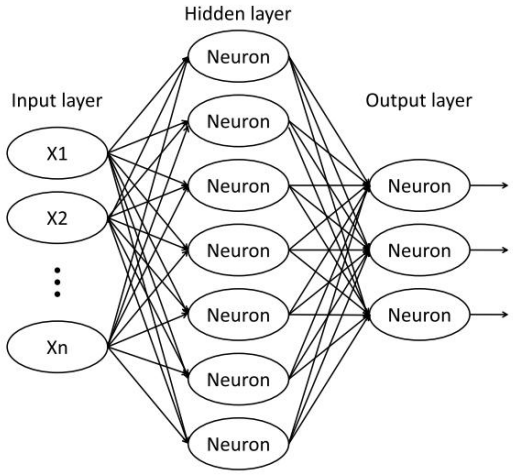

图 7.多层感知器的结构,由一个输入层、多个隐藏层和一个输出层组成。

类似地通过 OBD-II 端口或远程信息处理服务系统。
因此,IDS也被作为CAN总线的一个节点来监控网络。

尽管 Kvaser 和 Vector 等商业工具支持通过 OBD-II 端口进行 CAN 通信,但在这项工作中,我们使用了两个定制的 Raspberry Pi 设备,一个用于消息注入攻击,另一个用于记录 CAN 流量。捕获的 CAN 流量数据用作训练和评估拟议 IDS 的数据集。

4. 人工神经网络

4.1.人工神经元

人工神经网络 (ANN) 由几个相互连接的简单人工神经元组成。人工神经元是一种数学函数,被认为是生物神经元的模型;此外,它接收一个或多个输入并产生输入的加权和作为输出。这个过程如图6 所示，

$$y(x) = W \cdot X + b, \quad (1)$$

其中X是输入， y是人工神经元的输出。
W和b分别是一组权重和偏差。

传统的神经网络,例如多层感知器（MLP），是由多层人工神经元构成的,如图7 所示。这种结构化的人工神经元集合可以学习数据的特征并预测基于新样本的结果在学习上。

4.2.卷积神经网络

CNN 是一种深度神经网络,是一种基于大量神经元集合的计算方法。与传统的 MLP 模型不同,CNN 的每一层都由一个矩形的 3D 神经元网格组成。一层的神经元只连接到感受野中的神经元,感受野是紧邻前一层的一个小区域,而不是整个神经元集合。通过使用感受野,CNN 可以利用输入数据的空间局部相关性。接受域允许 CNN 对输入的小部分创建良好的表示;然后,CNN 使用它们来组装更大区域的表示。

图像识别是 CNN 的应用领域之一。 2016 年,谷歌宣布了他们最新的 CNN 模型,名为 Inception-V4 和 Inception-ResNet [11]。这些模型使用 ImageNet 数据集展示了卓越的性能。

通常,CNN 由图 8 所示的组件组成,即卷积层、激活函数、全连接层和池化层。示例中的模型有两个卷积层、一个池化层、一个全连接层和输出层。激活函数应用于卷积层和全连接层的所有神经元;但是,它们不适用于池化层。组件的更多细节在以下小节中描述。

4.2.1.全连接层和卷积层全连接层与传统神经网络（例如 MLP)中

的隐藏层相同。因为面对层中的所有神经元都是密集连接的,所以全连接层具有巨大的权重。相反,卷积层需要少量的权重,但计算量更大,因为它们有几十个或数百个特征图。因此,一般来说,大多数参数都部署在全连接层中;然而,大多数操作都是在卷积层中执行的。卷积层是 CNN 的关键组成部分,计算输入和滤波器权重之间的点积,以生成该滤波器的二维特征图。滤波器决定了上述卷积层的感受野。

图9显示了卷积层和全连接层的区别。卷积层通过计算窗口中值的加权和来计算输出,称为过滤器,它决定感受野。

卷积层需要的参数比全连接层少得多。例如,如果我们将 29 × 29 × 3 的体积处理成 27 × 27 × 6 的体积,则全连接层需要(29 × 29 × 3) × (27 × 27 × 6) = 11 M 个权重，

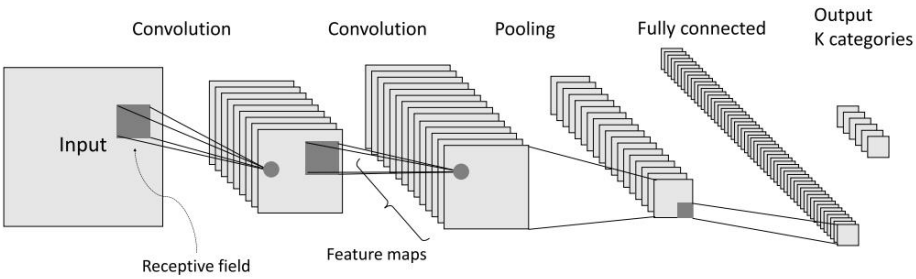

图 8. CNN 的典型示例。 CNN一般由池化层、全连接层和输出层之后的多组卷积层组成。

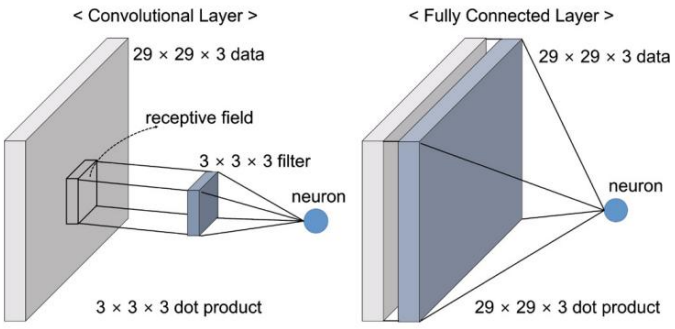

图 9.卷积层和全连接层的区别。前者连接感受野中的神经元作为输入,而后者连接下层的所有神经元作为输入。

而具有 3 × 3 大小过滤器的卷积层仅需要(3 × 3 × 3) × 6 = 162 个权重。

4.2.2.激活函数激活函数的目
的是为模型提供非线性。卷积是线性运算;因此,当不使用激活函数时,它的功能类似于单个感知器,即使采用了多个层。

激活函数应用于神经元的输出
如下所示：

$$h(x) = g(y(x)), \quad (2)$$

其中函数g是激活函数， h是激活函数的输出， y(x)是一个神经元的输出,是激活函数的输入。在 CNN 架构中,修正线性单元 (ReLU) 被广泛用作激活函数：

$$g(a) = \mathrm{ReLU}(a) = \max(0,a), \quad (3)$$

ReLU 输出零和输入 a 之间的最大值。
ReLU 的优点是在反向传播中易于微分以找到权重的最佳值。虽然它在x = 0时不可微分,但在其他范围内微分非常简单。当x > 0时,它的导数值为 1 ,当x < 0 时,它的导数值为 0。

激活函数通过投影一定范围的输出值来打破神经网络工作的线性,而使用ReLU时这些输出值为正值;因此,允许神经网络学习数据的复杂性和非线性特性。

此外,它决定了应该传递多少信息,类似于人类大脑中神经元的运作,它们是从中获得灵感的。

softmax 函数是一种特殊的激活函数,通常用于最后一个输出层。 softmax函数将任意实数值的K维向量z归一化为(0, 1)范围内实数值的K维向量σ(z) ,其和为1,计算为：

$$\sigma(z)i = \exp zi \Big/ \sum_{k=1}^{\text{钾}} \exp zk \ (i = 0 \ , \cdots , K). \quad (4)$$

softmax 函数的输出可以被视为多类分类中 K 个类的概率分布,表示每个类的分类置信度。

4.2.3.池化层池化层也
称为采样层,因为该层从内核分离的每个段生成单个值以减少数据的大小。池化层用于将每个卷积层中的计算量调整为相似。如果只使用卷积层

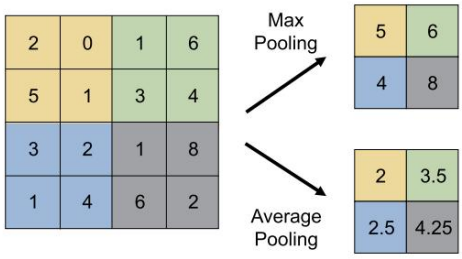

图 10. max-pooling 和 average-pooling 之间的区别。最大池化输出输入值中的最大值,而平均池化输出输入值的平均值。

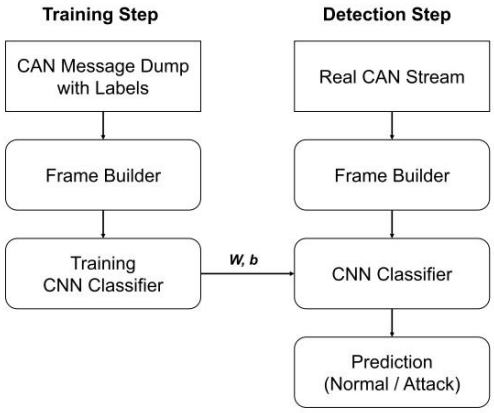

图 11.所提出的入侵检测系统的流程。该过程分为两个步骤:训练和检测。带有标签的 CAN 消息转储用于训练步骤中的监督学习,而真实的 CAN 流用于检测步骤。

如果没有池化层,后续卷积层的计算量会随着滤波器映射数量的增加而显着增加。

最大池化层输出由池核分隔的每个段的最大值。图10显示了步幅为2 的 2 × 2 内核的最大池化和平均池化之间的差异。内核决定池化区域的大小,步长表示内核在数据上滑动步长的大小。

5. 使用深度卷积神经网络的入侵检测系统（IDS）

5.1.提议的 IDS 的过程

所提出的 IDS 由两个步骤组成,训练和检测步骤,类似于一般的基于机器学习的 IDS,如图 11 所示。虽然在训练步骤中使用了标记的 CAN 流量转储,但真正的CAN流量没有标签直接用于所提出的 IDS 方案的检测步骤。

在训练步骤中,帧生成器从记录的 CAN 流量转储中检索 CAN ID,并组装由 29 个连续 CAN ID 组成的数据帧。在从 CNN 结构的第一个输入层遍历到最后一个输出层时,框架构建器构建的数据框被正确计算并分类为攻击或非攻击。

通过构建一个包含 29 个连续 CAN ID 的数据帧,所提出的基于 CNN 的 IDS 可以利用出现在 CAN 总线上的 CAN ID 的顺序模式。对于监督学习,包含一个或多个注入消息的数据帧被标记为攻击,不包含注入消息的数据帧被标记为非攻击。

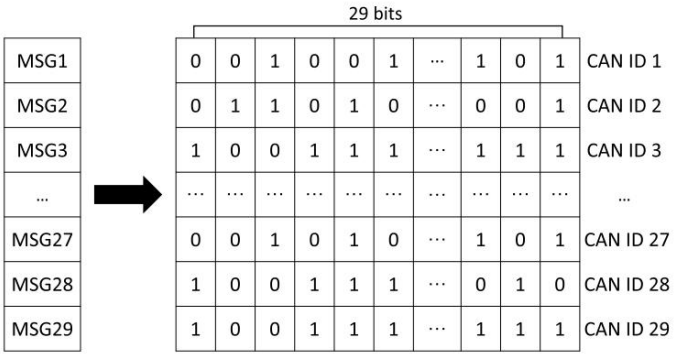

图 12.帧生成器将 CAN ID 序列转换为二维网格数据帧。
每个 CAN 消息都有一个 29 位 ID,帧生成器从 29 个 CAN 消息中收集 ID 以构建一个二维网格数据帧。

由于训练 CNN 分类器是一项耗时的任务,因此必须离线执行。一旦经过训练,CNN 分类器就可以相对快速地计算新样本的结果。在本文中,虽然所提出的 IDS 是离线测试的,但随着计算能力的提高,期望所提出的 IDS 可用于实时检测并不是不合理的。

5.2.框架生成器

通常,CNN 旨在接受网格中的数据作为输入,并利用空间局部相关性。我们将 CAN 流量的时间顺序模式处理为空间局部相关性,方法是使用框架构建器重塑顺序 CAN 消息数据以创建 2-D 数据帧,如图 12 所示。框架构建器提取29-位从最近的 29 条 CAN 消息中识别出来,并通过堆叠它们来构建一个 29 × 29 的按位帧。在不进行预处理的情况下使用纯位作为输入数据有两个原因。首先,为了提高效率,不需要额外的数据预处理,例如解码。由于 CAN 总线上每秒有数千条消息,因此设计 IDS 时应考虑计算效率以防止出现瓶颈。其次,位表示可以明确地显示标识符模式的波动。标识符的位表示如下：

$$ID = bi \quad (\text{对于 } i = 0, \cdots, 28), \tag{5}$$

其中bi是第 i 个位值。 frame builder 组装一个具有 29 个 ID 的数据帧以生成可用作卷积网络输入的正方形结构,如下所示

作为：

$$FRAME = IDi \quad (\text{对于 } i = 0, \cdots, 28) \tag{6}$$

$$= bij \quad (\text{对于 } i, \ j = \ , \cdots, 28), \tag{7}$$

0 ,其中bij是帧中第 i 个 ID 的第 j 个比特值。然后,每个帧标记为 1 表示攻击,0 表示正常,用于监督学习。

5.3.减少起始-ResNet

经过证实的

Inception-ResNet 是经过验证的 DCNN 模型之一。它旨在将多张图像分类为 1,000 个类别,并展示了创新性能。为了将 Inception-ResNet 应用到我们的环境中,由于输入和输出维度的差异以及数据结构的复杂性,我们通过减少其组件和调整参数来重新设计原始模型。最初的 Inception-ResNet 旨在将 299 × 299 × 3 输入图像数据分类为 1,000 个类别,而

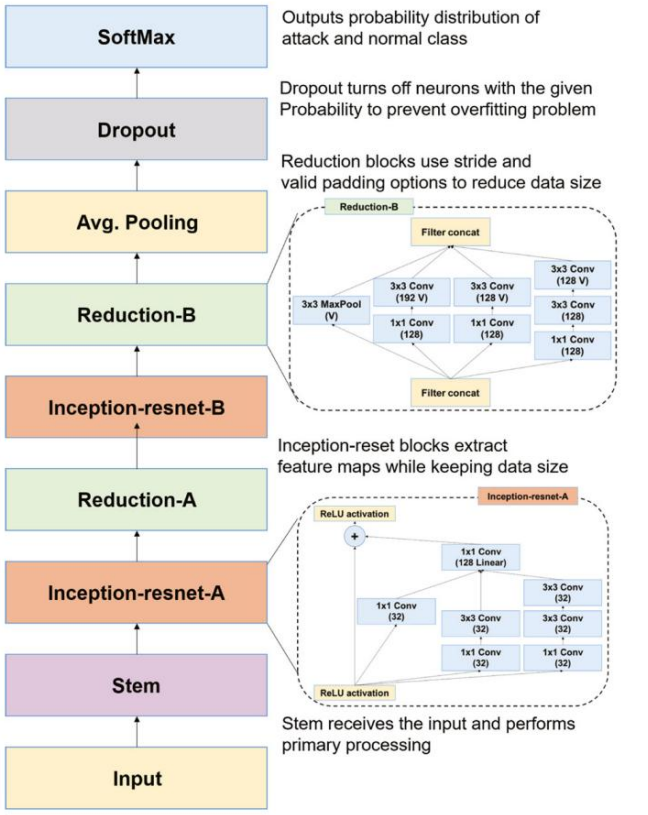

图 13.本工作中使用的简化 Inception-ResNet 的整体架构和模块。 Inception-resnet-A和-B的块数不同。原始的Inception-ResNet分别有五块、十块、五块Inception-resnet-A、-B、-C。简化后的 Inception-ResNet 每个块只有一个,没有 Inception-resnet-C 块。

我们的模型旨在将 29 × 29 × 1 输入数据仅分为两类。原始模型的输入和输出非常大,以至于无法直接应用于我们的数据集。

如果模型变得过于复杂,则需要过多的计算资源,并且有可能出现过拟合问题。在基于机器学习的技术中,过度拟合是一个常见问题,其中训练模型针对训练数据集进行了过度优化,从而降低了测试数据或实际生产数据的性能。

原始的 Inception-ResNet 由九种不同的块组成:stem、Inception-resnet-A、-B 和 -C、reduction-A 和 -B、average pooling、dropout 和 softmax。在简化的 Inception Resnet 中,几乎使用了整个原始架构,除了 Inception-resnet-C 块;此外,修改了其他块内的层结构和参数,如图13所示。在我们的模型中,在 reduction-B 块之后,正在处理的数据的大小变得足够小;因此,它直接用作平均池化的输入。此外,原始模型有 5 个 Inception-resnet-A 块、10 个 Inception resnet-B 块和其他块各一个;然而,我们只使用了 Inception-resnet-A 和 -B 块中的每一个。由于我们的数据量比原始 Inception-ResNet 的输入图像数据小得多且简单得多,因此我们不需要使用如此深的架构。表1列出了原始 Inception-ResNet 和缩减后的 Inception ResNet 块的输出维度。通过创建更轻的模型,我们有效地减少了内存需求和参数数量。原始模型每个输入需要大约 111 MB（前向+后向）;因此,当批处理大小为 20 时,大约需要 2.2 GB。但是,我们简化的 Inception-ResNet 每个输入只需要大约 2.2 MB,大约是 2%

表 1块的
输出维度。

堵塞	Inception-ResNet 简化的 Inception-ResNet
输入	299 × 299 × 3 29 × 29 × 1
Stem 35 × 35 × 256 Inception-resnet-A 35	13 × 13 × 128
× 35 × 256 Reduction-A 17 × 17 × 896	13 × 13 × 128 6
Inception-resnet-B 17 × 17 × 896 Reduction-	× 6 × 448 6 × 6
Inception-resnet-C 8 × 8 × 1,792平均。池化	× 448 2 × 2 ×
1,792 Dropout 1,792 Softmax 1,000	896未使用
	896
	896

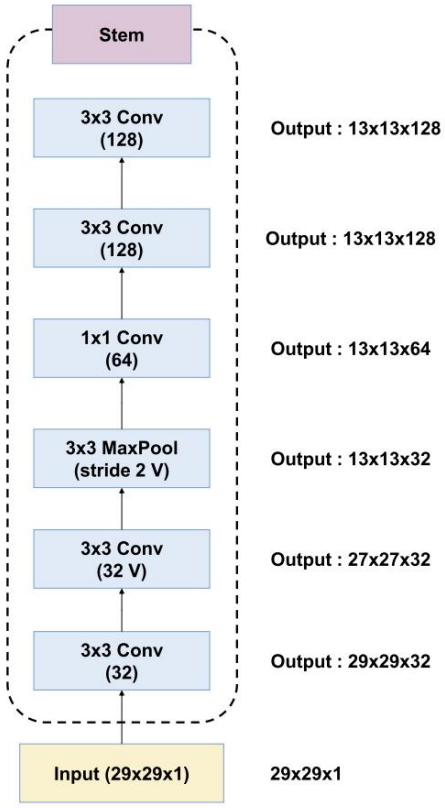

图 14.简化的 Inception-ResNet 的主干。这是网络的输入部分，并根据输入生成特征图。

原始模型所需的内存。此外，与原始模型相比，参数数量约为 18%，减少的 Inception-ResNet 中有 1.76 M 个参数，原始 Inception-ResNet 中有 9.8 M 个参数。

图14中描述的主干块从大小为 29 × 29的输入生成 128 个大小为 13 × 13 的特征图。 15和16，处理数据，同时保持其维度。相反，减少块，如图所示。 17和18，被插入到初始块之间以减少数据大小。平均池化层执行将 3-D 网格数据展平为 1-D 数组数据的作用，该层用于代替全连接层来维护每个特征图中所有元素的信息。

虽然一般的 CNN 结构在最终的 softmax 层之前有全连接层，但 Inception-ResNet 架构没有全连接层。相反，在 softmax 层之前使用单个平均池化层，因为最后一个卷积层的输出大小足够小，可以通过平均池化减少到单个值。在平均池化层，数据的维度从3×3×896减少

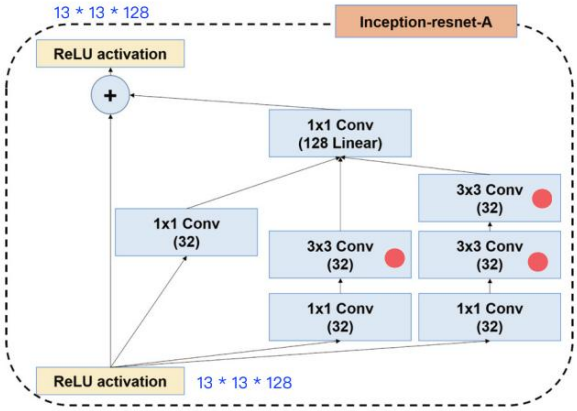

图 15.简化的 Inception-ResNet 的13 × 13 网格 (Inception-resnet-A) 模块的架构。在最初的 Inception-ResNet 中，它使用了 35 × 35 的网格数据。

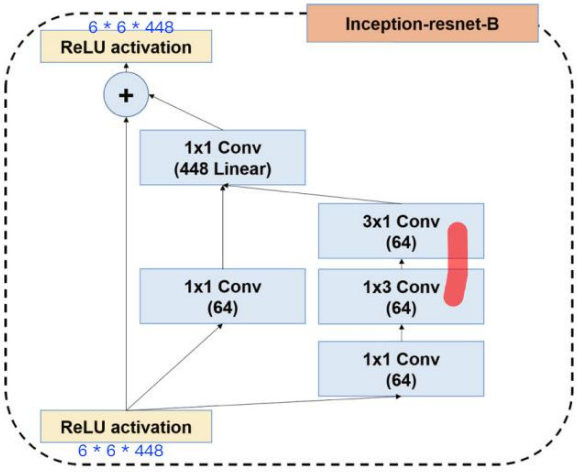

图 16.简化的 Inception-ResNet 的6 × 6 网格 (Inception-resnet-B) 模块的架构。在最初的 Inception-ResNet 中，它使用了 17 × 17 的网格数据。

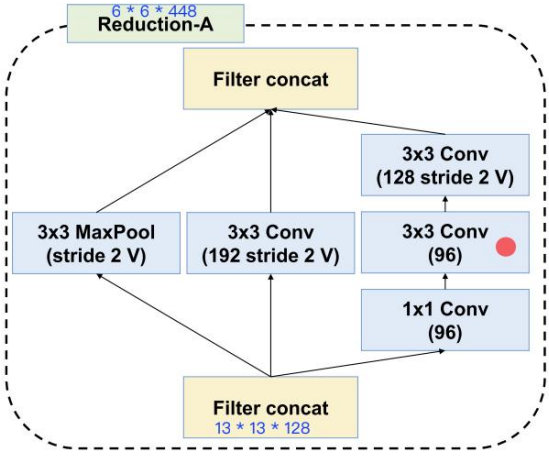

图 17. 13 × 13 至 6 × 6 网格缩减模块的架构。在原始的 Inception-ResNet 中，它将一个 35 × 35 的网格数据缩减为一个 17 × 17 的网格数据。

到 1 × 1 × 896，线性排列，然后最后传递到 softmax 层。

随着神经网络越来越深，深度神经网络需要正则化以防止过拟合问题。实践中最常用的技术是数据集扩充、权重惩罚 L1 或 L2 以及 dropout [37]。与 Inception-ResNet 模型类似，我们在平均池化层和 softmax 层之间使用了 dropout。这

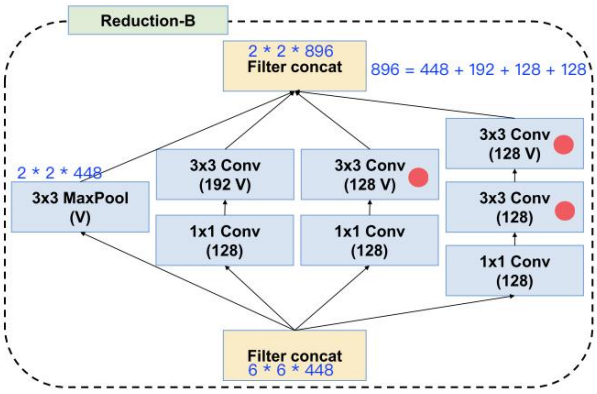

图 18. 6×6到2×2网格缩减模块的架构。在最初的 Inception ResNet 中,它将一个 17 × 17 的网格数据缩减为一个 8 × 8 的网格数据。

dropout 通过以给定概率随机停用一部分神经元来降低模型的学习程度。

训练模型,这意味着调整参数,需要测量当前模型的损失。 softmax层用作输出层。因此,我们的模型输出了目标类别 0 和 1 的概率分布,如下所示：

$$\hat{y}_i = p(c = i|x) \quad (\text{对于 } i = 0, 1), \tag{8}$$

其中x和yi是输入帧数据和当c是目标类时目标类的预测概率。因此,我们必须最大化概率p(c = y|x) ,使得目标类c等于给定x的真实类y 。在训练步骤中,我们不是最大化 y 的概率,而是最小化对数似然函数,称为交叉熵,如下所示：

$$H(y, \hat{y}) = -(1 - y)\log \hat{y}_0 - y \log \hat{y}_1, \tag{9}$$

其中y是真实标签,为零或一， y^是两个类别的预测概率分布。 y^和y的交叉熵是通过对真实概率和预测概率的乘积求和来计算的。在分类问题中,当真实概率为零或一时,交叉熵可以改写为真实标签预测概率的负对数,因为错误标签的真实概率为零,这导致另一个项是零。

然后,通过取平均值计算损失函数
给定 N 个样本的交叉熵,其推导出为：

$$\text{长（宽）} = \frac{1}{nn}\sum_{n=1}^{\text{否}} H(y, \hat{y}), \tag{10}$$

其中W是当前模型的参数集。更具体地说,我们的入侵检测模型将给定的输入数据分为两个可能的类别,标记为零和一,分别表示正常和攻击。因此,对于给定的输入数据x,模型预测目标类别的概率并输出预测标签y ∈ {0, 1} 。

六、实验结果

6.1.数据集

一辆真实的车辆被用来为我们的实验构建数据集。我们使用了两个自定义的 Raspberry Pi 设备,一个用于记录网络流量,另一个用于注入伪造的消息作为攻击节点。他们连接到车载

表 2
数据集概述。

攻击类型	普通消息 注入消息 3,078,250 (84%) 3,347,013
拒绝服务攻击	(87%) 2,766,522 (82%) 2,296,525 (16%)
模糊攻击	491,847 (13%)
齿轮欺骗	597,252 (18%)
转速欺骗	654,897 (22%)

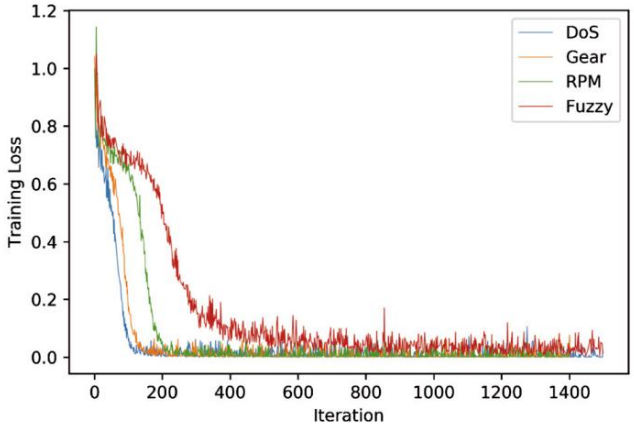

图 19.每个数据集的训练损失。从左到右分别是 DoS、gear、RPM 和模糊攻击数据集。 （对于图中颜色的解释,读者可以参考本文的网络版。）

通过位于汽车方向盘下方的 OBD-II 端口联网。通过 OBD-II 端口,我们的自定义节点能够向 CAN 总线上的真实 ECU 节点发送数据/从中接收数据。在正常状态下,CAN 总线上有 26 个不同的 CAN ID。

表2列出了实验数据集中正常消息和注入消息的数量。我们构建了四个数据集,分别是 DoS 攻击、模糊攻击、欺骗驱动装置和欺骗 RPM 仪表。每个数据集都是通过在受控环境中注入伪造消息的同时记录 CAN 流量来创建的。在收集数据时,汽车停在发动机开启的状态下。每个数据集包含 300 次注入消息的入侵。每次入侵执行 3 到 5 秒,每个数据集包含 30 到 40 分钟收集的 CAN 流量。

对于 DoS 攻击,我们每 0.3 毫秒向 CAN 消息中注入 29 个零位的 CAN ID,这是 CAN 总线中最主要的 CAN ID。 DoS 攻击的目的是损害网络可用性。由于尝试发送具有最主要 CAN ID 的消息的 ECU 总是在仲裁阶段赢得总线,因此其他 ECU 无法发送其消息。模糊攻击类似于DoS攻击;但是不同的是,模糊攻击中报文的CAN ID和数据值是完全随机的。我们每 0.5 毫秒注入一次随机消息。执行模糊攻击以使车辆发生故障。欺骗驱动齿轮和 RPM 仪表数据集是通过每 1 毫秒注入特定 CAN ID 的消息来实现的。

这些消息分别包含驱动齿轮和转速计的信息。欺骗攻击使我们能够欺骗原始 ECU 并更改仪表板上的转速表和驱动齿轮。

图19显示了早期训练步骤（~1,500)中每个数据集的训练损失。我们可以观察到,随着攻击数据复杂度的增加,训练损失的收敛速度变慢。具有相对简单模式的 DoS 攻击的训练损失收敛最早,注入随机数据的模糊攻击的训练损失收敛最后。

6.2.评价指标

我们测量了假阴性率（FNR)和错误率（ER)来评估分类性能。在我们的实验中,FNR 是未检测到的攻击帧的比例,ER 是错误分类的帧的比例,计算公式为：

$$FNR = FN/(TP + FN), \quad (11)$$

和

$$ER = (FN + FN)/(TP + TN + FP + FN), \quad (12)$$

其中TP （真阳性)和TN （真阴性)分别是被正确分类为攻击和正常的帧数， FP （假阳性）和FN （假阴性)是被错误分类的帧数分别作为攻击和正常。 FNR 应该很小,并且被认为对于车载网络中的攻击检测更为重要。

这是因为即使是少量未被发现的攻击也可能导致车辆暂时发生故障并成为安全威胁。

我们还计算了精度、召回率和 F1 分数,以便与其他算法进行比较。精度是检测为攻击的帧中实际攻击帧的分数。高精度与低 FP 率相关。频繁的误报使用户烦恼;因此,应该对它们进行管理以提高服务质量。使用以下公式计算精度：

$$精度= TP/(TP + FP)。 \quad (13)$$

召回率是正确检测到的攻击帧的一小部分,称为真阳性率 (TPR)。我们可以通过从 1 中减去 FNR 或使用等式轻松计算召回率：

$$召回= 1 − FNR = TP/(TP + FN)。 \quad (14)$$

F1 分数代表精确度和召回率之间的平衡。
F1-score 通常用于衡量数据集类别分布不均匀时的分类性能,计算

作为：

$$F 1 = 2 × （精度×召回率）/（精度+召回率）。 \quad (15)$$

6.3.超参数

根据超参数,模型的性能会发生变化。我们通过改变学习率的值来考虑实验。学习率控制模型的权重相对于训练损失梯度的调整程度。学习率的值越低,权重沿着损失面向下的斜率更新的越慢,收敛所需的时间越长。如果学习率值太小,训练模型不仅耗时,还会导致模型陷入局部极小;而不是全局最小值。相反,如果学习率太大,梯度下降可能会超过最小值并且可能无法收敛。因此,选择合适的学习率值有助于我们确定最佳权重。

最初,我们在将学习率从10－6更改为10－2 的同时测量 FNR 和 ER。在表3 中,我们可以观察到 FNR 和 ER 随着学习率的降低而增长。但是,当学习率设置为0.01时,梯度下降无法收敛,无法建立模型。因此,我们将10－3设置为最佳学习率。

表3
学习率的影响。

学习率	10－3	10－4	10－5	10－5
# 迭代次数		万		20k
FNR	0.267%	0.603%	1.04%	0.727%
急诊室	0.197%	0.292%	0.692%	0.574%

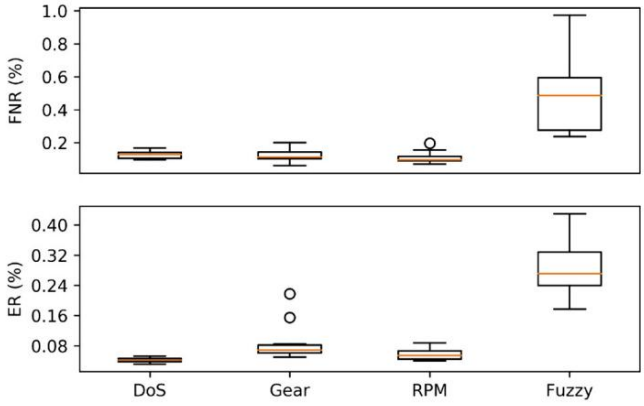

图 20.在 30 次重复实验中测得的假阴性率和错误率的箱线图。

表 4每个
数据集的总结测试结果。

数据集（%）	拒绝服务		齿轮		转速		模糊	
	错误率		错误率		FNR	急诊室	错误率	
意思是	0.13	0.04	0.12	0.09	0.11	0.06	0.50	0.28
标	0.02	0.01	0.04	0.05	0.04	0.02	0.24	0.07
准最	0.10	0.03	0.06	0.05	0.07	0.04	0.24	0.18
低	0.11	0.04	0.10	0.06	0.09	0.04	0.28	0.24
25%	0.13	0.04	0.11	0.07	0.10	0.05	0.49	0.27
50% 75%	0.14	0.05	0.14	0.08	0.12	0.07	0.60	0.33
最大限度	0.17	0.05	0.20	0.22	0.20	0.09	0.97	0.43

6.4.检测性能

根据 6.3 节的结果,我们将学习率设置为 0.001 来训练 IDS 模型。我们对每个攻击数据集进行了 30 次测试,并测量了 FNR 和 ER。对于每个测试,我们最初随机打乱数据以打破连续样本之间的局部性,因为数据的局部性会导致模型的学习性能下降。然后,我们将打乱后的数据集分成训练和测试数据,70% 用于训练,30% 用于测试。

测试结果如图20所示。如图所示,所提出的模型在 DoS、gear 和 RPM 攻击数据集中表现出稳定的检测性能,但在模糊攻击数据集中表现相对不稳定。看起来模糊攻击数据的复杂性远高于其他数据,需要更多的训练迭代才能创建稳定的模型。

表 4 总结了详细的测试结果,包括平均值、标准偏差、最小值和最大值。建议的 IDS 在 DoS、gear 和 RPM 攻击数据集中显示 FNR 和 ER 小于 0.1%,FNR 的 0.24%在最佳情况下,模糊数据集中的 ER 为 0.18%。

表 5 列出了每个攻击数据集上的多个测试之一的混淆矩阵。表中的数字代表 TN、FP、FN 和 TP 值的样本数。数据帧大小为 29 × 29 的样本由帧生成器创建,其中包含 29 个连续的位表示的 CAN ID。

表6列出了我们基于 DCNN 的技术与其他机器学习技术相比的检测性能。结果清楚地表明 DCNN 模型

表 5
拟议的 IDS 的分类结果。

拒绝服务	预测正常	预测攻击
真正常	26,555	0
真实攻击	12	11,354
模糊	预测正常	预测攻击
真正常	17,050	
真实攻击	36	3 13,405
欺骗 RPM 预测正常		预测攻击
真正常	26,412	
真实攻击	9	21,386
欺骗装备	预测正常	预测攻击
真正常	26,096	
真实攻击	27	5 19,835

在所有数据集上都优于其他机器学习模型。可以观察到,所提出的模型非常清楚网络流量的顺序性质,并且令人满意地捕获了顺序模式的变化。

有趣的是,我们可以看到 DCNN 模型优于 LSTM 模型。一般认为 LSTM 模型更适合处理时序数据。然而,所提出的模型旨在使用卷积运算捕获连续 CAN 消息之间的顺序特征。特定时间窗口内的CAN报文被视为CAN流量的二维矩阵快照,模型被归纳以学习快照的模式。与其他传统机器学习算法相比,LSTM 仍然表现出性能优势;然而,它不满足所提出的 DCNN 模型的性能要求。

与其他传统算法相比,ANN 模型还展示了针对 DoS、gear 的良好检测性能欺骗和 RPM 欺骗攻击;然而,它未能以大约 2% FNR 和 1.37% ER 令人满意地检测模糊攻击。在模糊攻击数据集上,深度神经网络模型,如 DCNN 和 LSTM,在低 FNR 和 ER 的情况下表现出显着的检测性能,而其他模型表现不佳。这些结果是由于攻击的复杂性。 DoS 攻击和欺骗攻击相对简单,因为它们会反复注入特定消息。然而,模糊攻击会注入随机消息,从而创建更复杂的流量模式。这导致模型难以学习攻击模式并降低检测性能。

支持向量机 (SVM)、k 最近邻 (kNN) 和 NB 表现出高精度,但召回率低,导致 F1 分数低。这意味着没有误报,但这些分类器倾向于偏向正常类别。

kNN 算法是一种基于聚类的算法,展示了对 DoS、齿轮欺骗和 RPM 欺骗攻击的足够性能,但没有检测到模糊攻击。 kNN 模型将大部分数据分类为正常数据。由于模糊攻击数据是随机生成和分散的,因此 kNN 算法似乎无法生成合适的聚类。

6.5.训练和测试的时间成本

在这项研究中,用于测试所提出模型性能的硬件是两个 2.30 GHz Intel Xeon CPU 和一个 Nvidia Tesla K80 GPU。使用的 CPU 是具有两个线程的单核处理器,GPU 具有 2,496 个 CUDA 内核和 12 GB GDDR5 VRAM。

表7显示“CPU only setting”的训练时间比“CPU with GPU acceleration setting”花费的时间更长。
与仅使用 CPU 进行训练相比,我们通过使用 GPU 加速实现了近 50 倍的性能提升;此外,时间成本与批量大小成正比。

表 6
与其他算法的比较。

拒绝服务	FNR	急诊室	精确	记起	F1
减少起始-ResNet	0.10%	0.03%	1.0	0.9989	0.9995
LSTM（256 个隐藏单元）	0.22%	0.07%	1.0	0.9978	0.9988
人工神经网络（2 个隐藏层）	0.18%	0.07%	0.9995	0.9982	0.9988
支持向量机 k-最近邻 (k = 5)	0.56%	0.17%	1.0	0.9944	0.9972
	0.70%	0.22%	0.9998	0.9929	0.9964
朴素贝叶斯	1.18%	0.35%	1.0	0.9882	0.9941
决策树	1.18%	1.34%	0.9762	0.8681	0.9776
齿轮欺骗	FNR	急诊室	精确	记起	F1
减少起始-ResNet	0.11%	0.05%	0.9999	0.9989	0.9994
LSTM（256 个隐藏单元）	0.32%	0.24%	0.9975	0.9968	0.9972
人工神经网络（2 个隐藏层）	0.16%	0.11%	0.9989	0.9984	0.9986
支持向量机 k-最近邻 (k = 5)	0.35%	0.15%	1.0 1.0	0.9965	0.9982
	1.58%	0.67%	1.0	0.9842	0.9920
朴素贝叶斯	0.84%	0.36%	0.9815	0.9916	0.9958
决策树	2.19%	1.72%		0.9781	0.9798
转速欺骗	FNR	急诊室	精确	记起	F1
减少起始-ResNet	0.05%	0.03%	0.9999	0.9994	0.9996
LSTM（256 个隐藏单元）	0.30%	0.13%	1.0	0.9971	0.9985
人工神经网络（2 个隐藏层）	0.11%	0.09%	0.9990	0.9989	0.9989
支持向量机 k-最近邻 (k = 5)	0.23%	0.11%	1.0	0.9977	0.9988
	0.80%	0.36%	0.9999	0.9920	0.9960
朴素贝叶斯	0.51%	0.23%	1.0	0.9949	0.9974
决策树	2.18%	1.68%	0.9842	0.9782	0.9812
模糊	FNR	急诊室	精确	记起	F1
减少起始-ResNet	0.35%	0.18%	0.9995	0.9965	0.9980
LSTM（256 个隐藏单元）	0.84%	0.65%	0.9936	0.9916	0.9926
人工神经网络（2 个隐藏层）	1.97%	1.37%	0.9886	0.9803	0.9844
支持向量机 k-最近邻 (k = 5)	4.45%	2.26%	0.9928	0.9555	0.9738
	93.42%	41.18%	1.0	0.0658	0.1236
朴素贝叶斯	9.03%	4.25%	0.9933	0.9098	0.9497
决策树	10.26%	7.23%	0.9359	0.8974	0.9163

表 7
DCNN 模型在不同硬件上的训练和测试时间。

硬件	批量大小	培训（秒/批次）	测试（秒/批次）
仅限 CPU	64	1.1805	0.2783
	128	2.2950	0.5338
	256	4.5776	1.0632
带 GPU 的 CPU	64	0.0221	0.0063
	128	0.0400	0.0109
	256	0.0749	0.0203

表 8
LSTM 模型在不同硬件上的训练和测试时间。

硬件	批量大小	培训（秒/批次）	测试（秒/批次）
仅限 CPU	64	0.0451	0.0158
	128	0.0737	0.0260
	256	0.1205	0.0423
带 GPU 的 CPU	64	0.0167	0.0103
	128	0.0169	0.0106
	256	0.0172	0.0115

我们还测量了表8中列出的 LSTM 模型的训练和测试时间以进行比较。与 DCNN 模型不同,GPU 加速的效率取决于该模型的批量大小。即使增加批量大小,LSTM 模型的时间成本也没有显着变化。对于 64 的批量大小,仅使用 CPU 和使用 GPU 加速的每次迭代训练时间分别约为 3 倍,分别为 0.0451 秒和 0.0167 秒;但是,对于 256 的批量大小,差异在 0.1205 秒和 0.0172 秒时增加到大约七倍。

在使用 GPU 加速的学习阶段,对于大批量,LSTM 模型显示出比 DCNN 快大约四倍的强度。相反,在测试阶段,对于 128 的批大小,两个模型都显示了大约 0.01 秒的相似推理时间。当批大小小于 128 时,DCNN 模型具有优势,而当批大小小于 128 时,LSTM 模型具有优势大于 128。但是,在实践中,最好设置较小的批大小,因为大批需要更长的时间来收集消息并且可能延迟入侵检测。

6.6.检测延迟

因为 CAN 是一个时间关键系统,检测延迟是实时检测性能的一个重要的安全相关指标。检测时延tl是攻击发生到检测到的时间,定义为：

$$tl = td - ta \quad (\text{其中} tcs \leqslant ta \leqslant tce), \tag{16}$$

其中， td表示检测到攻击的时间， ta表示攻击发生的时间。 tcs和tce分别是输入批次的数据收集持续时间的开始时间和结束时间。 td是tce和tp的总和,其中tp是模型的处理时间。那么,检测延迟tl可以改写为：

$$tl = tce + tp - ta, \tag{17}$$

进而，

$$tp \leqslant tl \leqslant tc + tp \quad (\text{其中} tc = tce - tcs)。 \tag{18}$$

当tcs和ta相等时,检测延迟变得最大,而当tce和ta相等时,检测延迟变得最小。因此，tc和tp都应该减少以最小化检测延误。然而,处理时间tp和数据收集持续时间tc取决于批量大小。因此,我们必须考虑的唯一因素是批量大小。

尽管就计算操作而言,更大的批量大小更有效,但我们将批量大小设置为 1 并测量推理时间以最小化检测延迟。

借助 GPU 加速,所提出的模型需要 5 毫秒来推断一个样本,该样本由 29 条 CAN 消息组成。这意味着该模型可以通过执行 200 次推理每秒处理 5,800 条消息。相反,仅使用 CPU 时,该模型需要 6.7 毫秒来推断一个样本,并且可以通过执行 150 次推断每秒处理大约 4,300 条消息。

由于测试车辆的 CAN 总线每秒传输大约 2,000 条消息,因此所提出的模型有足够的能力仅使用 CPU 处理两次,并使用 GPU 加速进行三次实时检测。

6.7.讨论与限制

本研究中使用的数据集是从真实车辆中收集的,并且可以在线公开获得[12]。该数据集是唯一包含带有标签的正常和攻击 CAN 流量的开放数据集。

我们的研究小组是唯一一个为安全研究分发车载网络入侵数据集的小组。目前,已发布的数据集仅包含消息注入攻击;但是,它将不断更新以包括复杂的攻击类型,例如消息篡改。

在本研究进行的实验中,所提出的基于 DCNN 的模型展示了针对消息注入攻击的良好检测性能。然而,该模型在检测未学习的攻击类型方面具有根本的局限性,因为它基于监督学习。为了解决这个问题,需要使用高级学习技术（例如对抗训练)对未知攻击检测进行额外的研究。除了检测从外部注入的消息外,还必须利用语义特征来检测由于车辆故障而导致的错误数据的传输,这需要车辆供应商的配合。

显然,时间性能取决于计算能力。尽管在本研究中模型的性能评估是在基于计算机的环境中进行的,但我们相信,当使用具有更高性能的计算设备时,所提出的模型将适用于实时检测。

七.结论

本研究主要侧重于构建一个入侵检测模型,该模型学习车载网络流量的顺序模式,并根据流量的变化检测消息注入攻击。我们提出了一个基于 DCNN 的 IDS。拟议的 IDS 是利用 Inception-ResNet 模型的结构设计的,该模型最初是为大规模图像分类而设计的。

Inception-ResNet 架构过于庞大和复杂,无法直接放入车载网络数据;因此,我们通过减少整个架构的大小和层数来重新设计 DCNN。为了直接使用来自车载网络流量的 CAN 消息的按位 CAN ID 序列作为 DCNN 分类器的输入而无需额外的特征工程,我们提出了一个名为 frame builder 的新模块,它生成一个二维数据帧类似于具有 CAN 消息的顺序位标识符的图像。该模块使 DCNN 分类器能够学习输入数据的时间顺序模式。

我们的实验涉及四类利用 CAN 总线的消息注入攻击,并且很可能发生在连接到外部网络的车辆环境中。我们构建了四种类型的消息注入攻击的数据集,这些数据集是公开的。实验结果表明

拟议的 IDS 能够识别车载网络流量数据之间的顺序模式。它证明了基于 DCNN 的 IDS 在识别攻击流量的入侵检测方面是可靠和有效的，包括 DoS、RPM 欺骗、齿轮欺骗和来自正常流量的模糊攻击。

本研究的主要重点之一是评估 DCNN 与其他著名机器学习算法（例如 LSTM、ANN、SVM、kNN、NB 和决策树)相比的性能。虽然 LSTM 和 ANN 表现出比其他传统算法更好的性能，但它们的 FNR 和 ER 仍然是所提出算法的两倍。特别是，对于模糊攻击数据集，所提出的 DCNN 模型展示了比其他算法显着的性能改进；因此，DCNN 模型在复杂的不规则随机攻击情况下比其他机器学习技术更有效。

未来，我们计划通过与供应商合作，进一步研究考虑语义特征，并改进 DCNN 模型，以应用对抗训练技术检测未知攻击类型。

竞争利益声明

作者声明，他们没有已知的可能影响本文报告的工作的竞争经济利益或个人关系。

致谢

这项工作得到了韩国政府 (MSIT) 资助的信息与通信技术促进研究所 (IITP) 赠款的支持（No. R7117-16-0161，自动驾驶汽车异常检测框架）。

参考

[1] G. Leen、D. Heffernan，扩展汽车电子系统，计算机35 (1) (2002) 88–93。

[2] NA Stanton、M. Young、B. McCaulder，线控驱动：驾驶员工作负载和回收控制与自适应巡航控制的案例，Saf。科学。 27 (2) (1997) 149–159。

[3] S.-H.金，S.-H。徐，J.-H。金，T.-M。月亮，C.-W。儿子，S.-H。 Hwang, JW Jeon，汽车系统的网关系统：LIN、CAN 和 FlexRay，载于：第 6 届 IEEE工业信息学国际会议，2008 年，INDIN 2008，IEEE， 2008 年，第 967–972 页。

[4] M. Farsi、K. Ratcliff、M. Barbosa，控制器局域网概述，Com
放。控制工程J. 10 (3) (1999) 113–120。
[5] C. Miller、C. Valasek，远程利用未改动的乘用车，载于： Black Hat USA，2015 年。

[6] S. Checkoway、D. McCoy、B. Kantor、D. Anderson、H. Shacham、S. Savage、K.
Koscher、A. Czeskis、F. Roesner、T. Kohno 等人，汽车攻击面的综合实验分析，载于：USENIX 安全研讨会，旧金山，2011 年，第 77-92 页。

[7] T. Hoppe、S. Kiltz、J. Dittmann，汽车 CAN 网络的安全威胁 实例和选定的短期对策，Reliab。工程。系统。
安全。 96 (1) (2011) 11–25。
[8] RM Ishtiaq Roufa、H. Mustafaa、SO Travis Tayora、W. Xua、M. Gruteserb、W.
Trappeb, I. Seskarb，车载无线网络的安全和隐私漏洞：轮胎压力监测系统案例研究，载于：第 19 届 USENIX 安全研讨会，华盛顿特区，2010 年，第 11-13 页。

[9] C. Szilagyi、P. Koopman，通过时间触发的嵌入式控制网络中的有效性投票进行低成本多播身份验证，载于：第五届嵌入式系统安全研讨会论文集，ACM，2010，p。 10.

[10] C.-W。 Lin, A. Sangiovanni-Vincentelli，控制器区域网络 (CAN) 通信协议的网络安全，载于：2012 年网络安全国际会议 (CyberSecurity)，IEEE，2012 年，第 1-7 页。

[11] K. He、X. Zhang、S. Ren、J. Sun，用于图像识别的深度残差学习，载于： IEEE 计算机视觉和模式识别会议论文集，2016 年，第 770-778 页。

[12] HM Song、HK Kim，Can network intrusion datasets， http://ocslab.hksecurity。净/数据集/汽车黑客数据集。 （2018 年 12 月 30 日访问）。
[13] C. Miller、C. Valasek，汽车网络和控制单元的历险记，载于： DEF CON，卷。 2013 年 21 日，第 260–264 页。

[14] M. Müter、N. Asaj，基于熵的车载网络异常检测，载于：智能车辆研讨会 (IV)，2011 IEEE，IEEE，2011，第 1110–1115 页。

[15] UE Larson、DK Nilsson、E. Jonsson，一种基于规范的车载网络攻击检测方法，载于：智能车辆研讨会，2008 年IEEE，IEEE，2008 年，第 220-225 页。

[16] HM Song, HR Kim, HK Kim, Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network, in: 2016 International Conference on Information Networking (ICOIN), IEEE, 2016, pp. 63 信息网络会议–68。
[17] K.-T。 Cho, KG Shin，用于车辆入侵检测的指纹识别电子控制单元，载于：第 25 届 USENIX 安全研讨会（USENIX 安全 16），USENIX协会，2016 年，第 911-927 页。

[18] M.-J。康，J.-W。 Kang，使用深度神经网络实现车载网络安全的入侵检测系统，PLoS ONE 11 (6) (2016) e0155781。
[19] A. Taylor、S. Leblanc、 N. Japkowicz，使用长短期记忆网络在汽车控制网络数据中进行异常检测，载于：2016 年 IEEE 数据科学和高级分析国际会议 (DSAA)，IEEE，2016 年，第 130-139 页。

[20] E. Seo、HM Song、HK Kim、GIDS：基于 GAN的车载网络入侵检测系统，载于：2018 年第 16 届隐私、安全和信任 (PST) 年会，IEEE，2018 年，第 1– 页6.

[21] C. Wang、Z. Zhao、L. Gong、L. Zhu、Z. Liu、X. Cheng，使用 HTM 的车载网络分布式异常检测系统，IEEE Access 6 (2018) 9091–9098。

[22] M. Levi、Y. Allouche、A. Kontorovich，联网汽车网络安全的高级分析，载于：2018 年 IEEE 第 87 届车辆技术会议（VTC 春季）， IEEE，2018 年，第 1-7 页。

[23] V. Chandola、A. Banerjee、V. Kumar，离散序列的异常检测：一项调查，IEEE Trans。知识数据工程师。 24 (5) (2012) 823–839。
[24] Z. Xing、J. Pei、E. Keogh，序列分类简要调查，ACM SIGKDD
探索。新闻12 (1) (2010) 40–48。
[25] SA Hofmeyr、S. Forrest、A. Somayaji，使用序列的入侵检测
系统调用，J. Comput。安全。 6 (3) (1998) 151–180。
[26] AP Kosoresow、S. Hofmeyer，通过系统调用跟踪进行入侵检测，IEEE
软件14 (5) (1997) 35–42。
[27] W. Lee、SJ Stolfo 等人，入侵检测的数据挖掘方法，载于： USENIX 安全研讨会，德克萨斯州圣安东尼奥市，1998 年，第 79-93 页。
[28] C.-M。 Ou，改编自基于代理的人工免疫系统的基于主机的入侵检测系统，Neurocomputing 88 (2012) 78–86。
[29] A. Shabtai、U. Kanonov、Y. Elovici、C. Glezer、Y. Weiss，“Andromaly”：安卓设备的行为恶意软件检测框架，J. Intell。信息。系统。 38 (1) (2012) 161–190。

[30] MH Bhuyan、DK Bhattacharyya、JK Kalita，网络异常检测：方法、系统和工具，IEEE Commun。生存。导师。 16 (1) (2014) 303–336。
[31] RR Karthick，副总裁 Hattiwale，B. Ravindran，使用混合方法的自适应网络入侵检测系统，载于：2012 年第四届通信系统和网络国际会议 (COMSNETS)，IEEE，2012 年，第 1-7 页。

[32] N. Goldenberg, A. Wool，用于SCADA 系统入侵检测的 Modbus/TCP 精确建模，Int。 J.暴击。基础设施。保护。 6 (2) (2013) 63–75。
[33] SJ Yu, P. Koh, H. Kwon, DS Kim, HK Kim, 基于 Hurst 参数的入侵检测系统异常检测，载于：2016 年 IEEE 计算机与信息技术国际会议(CIT)，IEEE，2016 年，第 234-240 页。

[34] H. Kwon、T. Kim、SJ Yu、HK Kim，基于自相似性的云计算轻量级入侵检测方法，载于：亚洲智能信息和数据库系统会议，Springer，2011 年，第 353–362 页.

[35] R. Mitchell, I.-R. Chen，网络入侵检测技术综述
物理系统，ACM Comput。生存。 46（4）（2014)55。
[36] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: 2017 国际信息网络会议 (ICOIN), IEEE, 2017 , 第 712–717 页。
[37] N. Srivastava、GE Hinton、A. Krizhevsky、I. Sutskever、R. Salakhutdinov， Dropout：一种防止神经网络过度拟合的简单方法，J. Mach。
学习。水库。 15 (1) (2014) 1929–1958。