基于特征关系图的流量异常检测系统设计与实现

(申请清华大学工学硕士学位论文)

培 养 单 位 ： 网络科学与网络空间研究院

学　　科 ： 网络空间安全

研 究 生 ： 高　涛

指 导 教 师 ： 李 风 华　副研究员

二〇二一年八月

Design and Implementation of Traffic Anomaly Detection System Based on Feature Graph

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Cyberspace Security

by

Gao Tao

Thesis Supervisor: Associate Professor Li Fenghua

August, 2021

学位论文公开评阅人和答辩委员会名单

公开评阅人名单

张千里	副研究员	清华大学网络研究院
施新刚	副研究员	清华大学网络研究院

答辩委员会名单

主席	段海新	教授	清华大学网络研究院
委员	李风华	副研究员	清华大学网络研究院
	安长青	副研究员	清华大学网络研究院
	施新刚	副研究员	清华大学网络研究院
	王会	副教授	清华大学网络研究院
秘书	刘保君	博士后	清华大学网络研究院

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）按照上级教育主管部门督导、抽查等要求，报送相应的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

现如今互联网已经成为最重要的信息化基础设施，智能运维作为新兴的网络管理技术倍受关注。网络流量异常检测是智能运维的关键环节，具有重要意义。然而，现有的流量异常检测研究多是面向仿真的开源数据集环境，少有针对真实网络环境的有效算法。本文面向清华大学校园网真实环境，研究网络流量异常检测算法和系统。校园网流量具有流量规模大、应用类型多、异常种类多且实时变化等特点。现有的异常检测算法在校园网环境下具有以下几个问题：（1）应用类型多导致流量特征复杂，因此很难达到较高的检测率和较低的误报率；（2）不管是传统机器学习算法还是拟合能力更强的深度学习模型，都很难有效地从复杂的流量环境中学习到正常流量的基线；（3）面对海量数据规模时，现有的算法模型无法或很难做到实时性。为应对这些问题，本文主要的研究内容和创新点如下：

1. 首先在经典开源数据集和清华大学校园网数据集上对多个算法进行了实验评估，分析了现有算法存在的问题。然后设计了一种基于特征关系图的循环神经网络算法 (FG-RNN)。通过在 RNN 算法中引入特征图信息，FG-RNN 模型可以增强对特征的利用。实验结果表明，FG-RNN 算法检测结果优于现有的基于机器学习、深度学习的算法，且这些提升效果来源于本文引入的特征关系图。
2. 在 FG-RNN 的基础上，设计和实现了基于 Spark Streaming 的实时异常检测系统，该系统可以在校园网海量的数据下实时异常检测。
3. 该系统在校园网中检测出的端口扫描进行了定量分析，发现其不仅是安全问题，同时会对无线网性能产生影响。因此进一步提出了两种优化策略，可以有效地缓解该问题。

关键词：大规模校园网；异常检测；系统设计；神经网络

Abstract

Nowadays, the Internet has become one of the most important information infrastructure. As an emerging network management technology, AIOps has attracted much attention. Network traffic anomaly detection is a key component of AIOps. However, most of the existing traffic anomaly detection studies are developed based on simulated environments with open source dataset, and few effective algorithms are available for real network environments. In this paper, we study network traffic anomaly detection algorithm and system for campus networks, in particular, the campus network of Tsinghua University. Campus network traffic is characterized by large traffic volume, many types of application, anomalies and real-time changes. Existing anomaly detection algorithms have several problems in such an environment: (1) multiple application types lead to complex traffic characteristics, so it is difficult to achieve a high detection rate and a low false alarm rate; (2) neither traditional machine learning algorithms nor deep learning models with better fitting ability can effectively learn the baseline of normal traffic from the complex traffic environment; (3) when facing the massive data scale, existing algorithmic models are unable or difficult to work in a real-time fasion. In order to solve the above problems and challenges, the main work and contributions of this paper are as follows.

1. We first evaluate multiple algorithms with both classic open source datasets and traffic collected from Tsinghua University campus network dataset, and analyzed the problems of these algorithms. Then we design a feature graph based recurrent neural network algorithm (FG-RNN). By introducing feature graph into the RNN algorithm, the FG-RNN model can make a better use of features. The experimental results show that FG-RNN algorithm works better than those algorithms that are based on machine learning and deep learning, and the improvement effects comes from the feature graph we have introduced in FG-RNN.
2. Based on FG-RNN, we design and implement a real-time anomaly detection system on top of Spark Streaming. This system can detect real-time anomaly from the massive data of our campus network.
3. We conduct a quantitative analysis on the port scanning activities detected by our system. We find that these activities are not only security problem, but also affect the performance of the wireless network. We further propose two optimization

strategies to effectively alleviate this problem.

Keywords: Large-scale Campus Network; Anomaly Detection; System Design; Neural Network

目 录

插图清单

附表清单

符号和缩略语说明

RNN	循环神经网络
FG-RNN	基于特征关系图的循环神经网络
BGP	边界网关协议
PCA	主成分分析
DoS	拒绝服务攻击
DDoS	分布式拒绝服务攻击
SVM	支持向量机
SOM	自组织映射
CAMPUS	校园网数据集
ARP	地址解析协议
PortScan	端口扫描

第 1 章　引言

1.1　研究背景

互联网自诞生以来，随着数十年的发展，已经成为了最重要的信息化基础设施。我们从有关我国互联网络发展状况的统计报告[?] 中可以认识到：“截至 2020 年 12 月, 我国的移动网民用户规模已经累计达到 9.88 亿, 互联网的网络普及率已经高达 70.4%”，如图 1.1所示。互联网的主要应用领域已经包括即时移动通信、搜索结果引擎、网络即时新闻、线下线上教育、购物以及出行等各个方面, 可以说如今互联网已经和每一个人的工作日常生活息息相关密不可分。

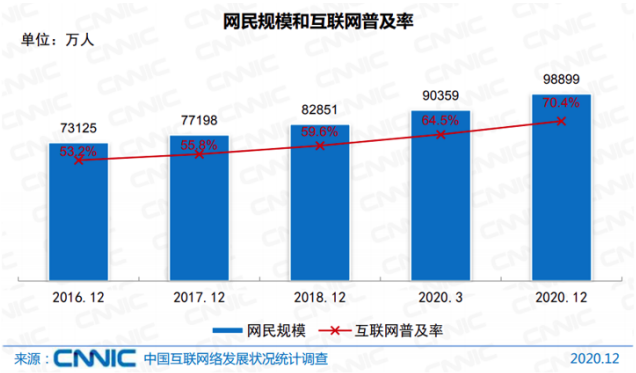


图 1.1　网民规模和互联网普及率

随着网络重要性的提升、用户规模的膨胀，管理网络的难度也越来越大。网络系统繁琐而又复杂，它的设计、运营以及维护都必须依靠专业的网络运维技术人员。早期的网络运维管理工作绝大多数都是由运维工作人员手工操作来完成，此后人们逐渐发现一些重复性的工作可以用自动化脚本来实现，于是诞生了自动化运维。自动化的运维系统可以认为是一种建立在专家经验和人为制订规则之上的系统。但是随着互联网规模迅速增长，以及其服务种类和方式的日益多样化，人为设定基础运维规则的方法与技术，渐渐难以应付大规模网络的运维需求。在此情况下，智能运维应运而生，与自动化运维仰仗专门人员的知识、人为制定规则有别，智能运维的着重点在于使用机器学习算法，先从庞大的运维管理数据中反复吸收经验进行学习，进而有条不紊的逐步提取规则。

异常检测是智能运维的关键环节，具有至关重要的意义。从网络故障管理的角度来说，做好异常检测可以提前预测故障的发生；从性能管理的角度来说，可以发现性能不佳的区域，避免因误配置、架构不合理导致性能下降；从安全管理的角度来说，在网络攻击的前期阶段，及时发现并预警后续攻击，进而做出防御措施。

因此，在复杂的网络环境中甄别出有效和异常流量尤为重要：在重大事故发生前，根据各项流量特征的变化，提前预测出即将发生的事故，提高应急响应速度，防患于未然。

本文以校园网为研究对象，进行网络流量异常检测算法和系统的研究。清华大学校园网是全球规模最大、架构最复杂、流量场景最多变的校园网之一，具有以下几个特点：

1. 用户规模大，流量峰值高。每天有 10 万台设备活跃在线，同时在线设备最多为 7 万台，峰值流量约为 30Gbps，图 1.2展示了日常中的用户流量变化。
2. 用户应用类型多。如图 1.3所示，清华大学校园网中的用户类型远比一般的企业网复杂，在网络环境中几百种应用同时使用，这给数据分析带来了很多困难。
3. 异常流量是常态。扫描流量、攻击流量占比多。

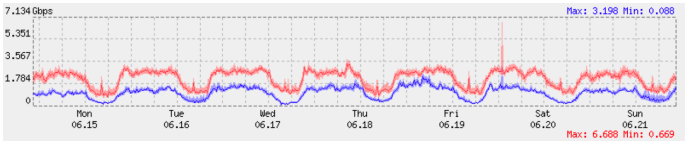


图 1.2　用户流量变化图

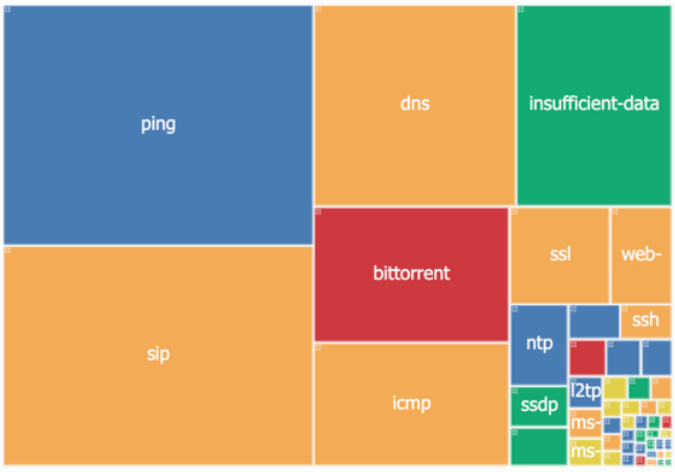


图 1.3　用户应用类型图

1.2　论文的研究内容

围绕 1.1 节中提到的清华大学校园网流量异常检测的特点和挑战，本文分别展开了以下几方面的研究：

1. 本文分析对比了流量异常检测领域的两个经典开源数据集 UNSW-NB15 和 CICIDS2017，引入了清华大学校园网的真实数据集，并且分别在这几个数据

集上对现有的异常检测算法进行了实验对比。

2. 在现有算法的基础上，本文将特征关系引入到神经网络的训练中，提出了一种基于特征之间关系图的循环神经网络算法 (FG-RNN，Feature Graph-Recurrent Neural Network)。在复杂的网络流量环境下，流量特征种类繁多，且特征之间的相关性会随着流量变化而变化。原有的异常检测算法通常直接利用提取的特征进行训练，本文提出的 FG-RNN 算法有效利用了特征之间的相关性信息，将其加入到神经网络的训练过程中。
3. 本文对基于特征之间关系图的循环神经网络算法进行了流式改造，使之能够满足当前实时异常检测的需求。为了应对海量的校园网流量数据和高速数据流，本文设计了一个基于 Spark Streaming 的实时异常检测系统。该系统分为输入模块、模型模块、检测模块三部分，输入模块负责将流量数据进行窗口划分、特征选择、特征抽取、合并计算，得到的特征矩阵与预训练得到的关系矩阵一同送到模型中进行训练，模型中的参数周期性更新以避免模型效果衰减过快，最后由检测模块对当前流量进行判别，并给出异常流量的类别。
4. 本文对系统检测出的端口扫描进行了定量分析，发现其不仅是安全问题，同时会对无线网性能产生影响，严重时会造成 100ms 以上的时延增加。本文提出了两种优化策略，可以有效地缓解该问题。

1.3　论文内容和组织结构

第一章为引言，主要介绍本文的研究背景以及研究内容。

第二章是关于网络流量异常检测的相关工作。首先对网络流量异常的定义和分类进行了介绍；接下来介绍了基于不同原理的常用异常检测算法；然后重点介绍了本文提出的算法所依赖的模型：循环神经网络模型；然后分析对比了开源数据集与校园网真实数据集，并且在多个数据集上对现有的算法进行了实验对比，指出了现有的异常检测算法在校园网流量环境下存在的主要问题。

第三章至第五章是本文的主要研究内容。第三章首先对现有算法的不足进行了详细分析，然后提出了基于特征关系图的循环神经网络算法，并进行了算法性能评估。第四章设计并实现了一个基于 Spark Streaming 的流式处理系统，介绍了各个模块实现的架构和流程，对系统运行结果进行了分析。最后第五章研究了端口扫描问题对无线网性能的具体影响，并且提出了两种优化策略，均可有效地缓解端口扫描造成的性能问题。

最后，第六章对全文进行了总结，并且提出了未来展望。

第 2 章　异常检测算法分析与对比

2.1　引言

首先本章介绍网络流量异常的定义和分类，异常检测系统的通用框架如图 2.1所示[?]。然后介绍流量异常领域常用的各类算法，由于本文主要工作是以循环神经网络作为基础，接下来介绍了 RNN 及其变体的原理。然后介绍了常用的异常检测数据集和本文中使用的校园网数据集并在这些数据集上对异常检测算法进行了分析与对比。

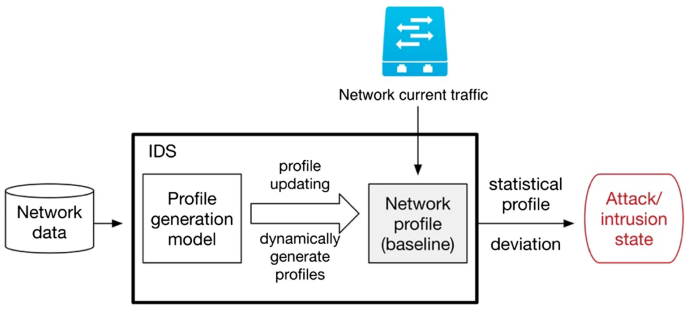


图 2.1　异常检测的通用框架

2.2　网络流量异常的定义和分类

Hawkins(1980)[?] 为“异常”下了一个定义: 某些数据由于在数据集中表现出差异，因此值得怀疑这些数据不是一般的随机偏差，而是来自完全不同的机制，我们称这些数据为“异常”。例如在道路交通领域，某条道路的车流量突然增多，甚至多至堵塞，又或者突然减少，此时车流量数据就是一个异常。因此网络行为的异常就是指那些与正常的、标准的，我们所预期的行为相异的表现。为了检测网络异常，网络所有者必须有一个预期或正常行为的概念，我们称其为“基线”。要检测网络行为的异常，就需要确保基线持续稳定，如果我们在监测的过程中发现有不寻常的走向，或者有特殊事件出现，就会合理怀疑有外部的蓄意攻击导致网络流量呈现异常，以及监控指标被改变的状况。

本文只关注引起网络流量特征变化的恶意行为，而对于系统权限提升，缓冲区溢出等黑客攻击手段暂时不做研究。

网络流量异常具体有哪些类别，学术界没有统一的意见。本文中关注的网络流量异常按照产生意图分为恶意和非恶意两类，其中恶意行为主要有拒绝服务攻击、网络扫描、BGP 劫持、网络蠕虫、僵尸网络等；非恶意行为主要有物理故障、

突发事件等。接下来我们对这些异常分别进行介绍：

- 拒绝服务攻击（Denial of Service，DoS）：攻击者往往通过构造大量请求访问目标主机，使其无法处理正常的流量请求，导致正常用户被拒绝服务。
- 网络扫描：攻击者在发动网络攻击之前，普遍会先进行网络扫描，寻找可攻击的目标。网络扫描通常是网络攻击的前奏。
- BGP 劫持：BGP 劫持是指通过一个自治系统错误宣称 IP 为其所有，使得使用边界网关协议维护的互联网路由表的路由器错误地将用户发送的数据传送给非数据传送目的地。
- 网络蠕虫：网络蠕虫通过网络和电子邮件进行复制和传播，它们首先搜索并且识别出本身具有缺陷的通信端点，在对这些端点施加攻击后，又经由局部地区的私有网路或者庞大的互联网扩散到其他通信端点。例如“熊猫烧香”及其变种就是蠕虫病毒。
- 僵尸网络：僵尸网络指由感染了恶意软件的计算机组成的网络，这些计算机集群由攻击者控制。
- 物理故障：物理故障也就是指现实世界中的设备有了损坏，包括但不限于电源与设备未连通、路由器无法传输讯息、链路间无法连接、线缆破损等意外事故。
- 突发事件：突发事件是指引起网络瞬时拥塞的事件，有可能是管理员配置错误，也有可能是正常的网络操作，如某网站访问量激增。

2.3　网络流量异常检测算法

本节将介绍在异常检测领域主流的一些算法，根据所依赖的技术原理的不同，将这些算法分为了基于分类、基于统计、基于聚类、基于信息论的异常检测算法。

2.3.1　基于分类的异常检测算法

本文在现有的大量基于分类的网络异常检测技术中，主要讨论以下三种技术，“支持向量机”（Support Vector Machine，SVM），“贝叶斯”（Bayes）以及“神经网络”（Neural Network，NN）。

2.3.1.1　支持向量机

Eskin 等人[?] 引入无监督 SVM 的概念来检测异常事件。常规的 SVM 的原理是推导出一个超平面，使得正类样本和负类样本之间的分离余量最大化，将特征空间中的两类数据进行分离。标准的 SVM 算法是一种监督学习方法，需要标记数

据来创建分类规则。而该算法经过改进 SVM，试图将整个训练数据集从原点分离出来，找到一个以最大余量将数据实例与原点分离的超平面。

Hu 等人[?] 提出了一种忽略噪声数据的异常检测方法，该方法使用 Robust SVM(RSVM) 来开发。标准的 SVM 有一个主要假设，即：“所有的训练样本数据都是独立且相同分布的 (i.i.d)”。但是在实际场景中，训练数据往往包含噪声，这就会导致标准 SVM 会学习出一个高度非线性的决策边界，从而导致通用性较差。有鉴于此，RSVM 以类中心的形式加入了平均化技术，使得决策面更加平滑。此外，RSVM 另一个优点是能大大降低支持向量的数量，从而减少运行时间，提高效率。

2.3.1.2　朴素贝叶斯

朴素贝叶斯是一个简单的概率分类器，通常用于网络入侵检测问题。它将先验信息与样本信息结合起来，并以统计推论的方式进行实施，从而利用概率显示出各种形式的不确定性。朴素贝叶斯假设了所有输入属性在条件上都是彼此独立的。

Kruegel 等人[?] 假设异常检测系统包含许多模型，用于分析一个事件的不同特征。他们指出了在这种系统下异常检测技术造成高误报率的两个主要原因：一是异常检测系统通过将多个概率模型的输出进行汇总，而每个模型往往只给出一个事件的常态/异常的得分或概率，从而导致高误报率；二是异常检测系统无法处理那些不正常但合法的行为，如 CPU 利用率、内存使用率突然增高等。基于贝叶斯网络的概念，Kruegel[?] 指出可以使用一种方法降低误报率，该方法能够正确处理那些在合法范围内稍显不正常的行为。对于一个输入事件的有序流 ($\boldsymbol{S} = e_1, e_2, e_3...$)，异常检测系统决策每个事件是正常还是异常。该决策基于 k 个模型 ($\boldsymbol{M} = m_1, m_2, ..., m_k$) 的输出 ($o_i | i = 1, 2, ..., k$) 和可能的附加信息 ($\boldsymbol{I}$)。应用贝叶斯网络来识别异常事件，引入根节点，根节点代表一个具有两种状态的变量。一个子节点用于捕捉模型的输出，子节点与根节点相连，预计当输入异常或正常时，输出事件会有所不同。

2.3.1.3　神经网络

神经网络是深度学习的热门技术，早已在多个方面实践成功，但其对计算量的要求很高。神经网络对数据进行分类的优势也可被用于网络异常检测。在网络异常检测领域，神经网络通常会和其他技术进行结合，如统计方法。

Hawkins 等人[?] 提出了一个多层的前馈神经网络，该神经网络可以用来进行异常值的检测。这个多层前馈的神经网络 (multi-layer feed-forward neural networks) 也就是 Replicator Neural Networks。具体而言，它的运行方式是设置了三个隐藏层，

夹在输入层和输出层的中间，目标是通过训练，使输出层能以最小的误差重现出输入的数据模式。其原理在于输入层与输出层节点的数量比隐藏层节点的数量要多，因此隐藏层能够具有压缩数据和恢复数据的作用。

?] 提出了一种分层入侵检测模式，这是他在神经网络的技术基础上搭配了统计方法建立的模型，此系统用一个连续变量 (*t*) 表示神经网络分类器的输出，若变量为 1 那么当前没有攻击，而-1 出现时则是确切的入侵。

除了统计模型，自组织映射 (Self-Organizing Map，SOM) 亦被用于网络异常检测。Ramadas 等人[**?**] 提出，由于 SOM 基于一个假设：即网络攻击可以由不同的神经元组来描述，这些神经元组与其他神经元组相比，在输出神经元图上覆盖更大的区域。因此利用 SOM，可以对网络流量进行实时分类。

Poojitha 等人[**?**] 设计出了一种前馈神经网络，使用的训练算法是“反向传播算法”，该模型的要点在于学习出系统无异常期间所表现出的活动模式，并且将违反此期间的行为归类于异常。

2.3.2　基于统计的异常检测算法

从前检测异常是使用假设-检验的方法，也就是要利用统计与概率模型。第一步是预测数据的分布，并从中找到预先认定的“异常”，在此操作下总是使用极值分析或假设检验。也就是说，当前存在基础的一维数据时，首先拟定数据是遵守正态分布的，若有超出均值达到某个范围外的点，就认定是异常点。此操作推广到高维后，假设各个维度相互独立。这类方法的好处是速度比较快，但是因为总要进行“假设”，所以效果不一定很好。因此统计技术的主要挑战是减少固定阈值引起的误报[**?**]。例如，可以使用统计信号处理程序来提高检测率，同时减少误报，如 Lakhina 等人在主成分分析工作中所做的工作[**???**]。

Nong 等人[**?**] 基于卡方检验的距离测量方法，将卡方检验的理论应用于异常检测。根据此技术，首先需要建立一个正常事件的基线，然后将偏离统计值的点视为异常。基于 chi-square 检验统计量的距离测量方法为：

$$\chi^2 = \sum_{i=1}^{n} \frac{(X_i - E_i)^2}{E_i} \tag{2-1}$$

其中 X_i 表示的是第 i 个变量的观测值，E_i 表示的是第 i 个变量的期望值，变量的数量以 n 表示。当一个变量的观测值接近预期时，χ^2 的值就会很低。根据 3σ 定律，当观测值的 χ^2 大于 $\bar{X^2} + 3S_X^2$ 时，该值被视为异常。

2.3.2.1　小波分析

小波变换的基本原理涵盖在傅里叶变换的过程当中，起初三角函数基是无限长的，通过小波变换将其变成长度有限且会衰减的小波基。强大的基函数小波，在时间和频率上是局部的，允许被表示的序列和它们的系数之间有密切的联系。如此一来频率得以获取，同时还可以定位到时间。

Callegari 等人[?] 提出了一种利用小波与基线相结合的实时异常检测方法。它是通过提取 NetFlow 轨迹，并将其转化为 ASCII 数据文件，进行路由器级别的分析。经过格式化后，通过哈希函数将不同的流量汇总到基线中。然后，将时间序列数据进行小波变换，若发现不连续点则视为异常点。

另一项使用小波的研究是由 Hamdi 等人[?] 产生的。它依赖于通过区分危险和非威胁性的异常来识别与攻击相关的异常。这项任务是在周期观测概念的基础上完成的，小波理论被用来分解一维信号，以分析其特殊频率和时间定位。

2.3.2.2　主成分分析

通常情况下，“主成分分析”(Principal component analysis，PCA) 方法的用法是将原本在高维空间里面的数据点, 以投影法降至低维空间内。但是我们也可以将之用于检测网络流量异常这个领域。该方法的主要原理是将特征空间为 n 的原数据集映射到新的更小的特征空间 k 中，新特征即为主成分 (PC)，其中 $k << n$，这些 PC 是一组正交向量，它们构成了一个 k 维子空间。同理，异常点的判别方法，是将存在于初始空间内的那些数据，映射到主成分空间，接着再把投影拉回到原始的空间。假如进行投影和重构的数据只包含第一主成分，那么大多数的数据在重构前后误差值不大；但是对于异常点而言，重构之后的误差，相较此前的误差值依然很大。

Lakhina 等人[?] 利用主成分分析法对流量测量结果进行二分，分为正常和异常两类。主成分分析是基于将一组网络流量测量所占的高维空间分离成不相干的子空间，分别对应正常和异常的网络条件。PCA 的结果是 k 个子空间，对应正常的网络流量行为，至于剩下的 m 个子空间 ($m = n - k$) 的组成成分则是异常或噪声。接下来需要通过制定有差异的阈值, 使这些结果中的异常流量被检测出来。这时的做法是将每一个新的流量测量值一一投射到 k 和 m 这两个子空间上。

Pascoal 等人[?] 提出的异常检测方法采用了鲁棒 PCA 检测器与鲁棒特征选择算法合并，以获得对不同网络背景和环境的适应性。该鲁棒 PCA 与经典的 PCA 方法相反，它对异常值不敏感，并且无需使用可靠标记的数据集进行训练。

Fernandes 等人[?] 发表了根据统计程序主成分分析生成出的一种异常检测算

法。这种算法首先生成一个 “使用流量分析的网络段数字签名 (DSNSF)” 的网络基线，然后将该基线与真实网络流量进行比较以识别异常事件。该系统分析了几天内的历史网络流量数据，从中找出最重要的流量时间间隔，同时对数据集进行了缩减，使新的缩减集能够有效地描述正常的网络行为。然后，以 DSNSF 作为阈值，通过得到的 PCA 参数，限制一个区间，偏离阈值的被认为是异常的。该系统共使用七个流量特征，三个 IP 流量特征 (bits/s，packets/s，flows/s) 被用来生成 DSNSF，四个流量属性（源 IP 地址、目的 IP 地址、源 TCP/UDP 端口和目的 TCP/UDP 端口）被用来生成一个包含有关异常流量间隔的信息报告。这种方法的缺点是只针对流量属性进行判定异常，只考虑检测基于流量的攻击。

2.3.2.3　协方差矩阵

协方差矩阵是二阶统计，已经被证明是一种强大的异常检测方法。该领域的一个有趣的方向是寻找哪些变量最能标记网络异常，用以提高检测性能。

Yeung 等人[?] 采用协方差矩阵分析来检测洪泛攻击。该方法将网络流量建模为协方差矩阵样本，以利用时序样本中包含的统计量达到检测泛洪攻击的目的，直接利用协方差矩阵的变化和相关特征的差异来揭示正常流量与各种类型的洪泛攻击之间的变化。

Xie 等人[?] 研究了一种基于段的方式处理数据的技术。因为在无线传感器网络 (WSNs) 中，人们观察到大多数异常事件都会持续相当长的时间。由于现有的异常检测技术通常是以基于点的方式单独处理每个观测值，它们无法可靠和有效地报告在单个传感器节点中出现的这种长期异常。因此该方法采用斯皮尔曼等级相关系数（Spearman’s rank correlation coefficient 或 Spearman’s ρ）和差分压缩概念近似的样本协方差矩阵，以大幅降低计算和通信成本。

2.3.3　基于信息论的异常检测算法

信息论是一门以信息量化和冗余分析为核心的数学学科，其前身是 1948 年 Claude E. Shannon 在寻求信号处理和通信操作的数据压缩、传输和存储时提出的设想[?]。然而，它的应用扩展到许多其他领域，如电子通信、决策支持系统、模式识别等。常用的信息理论测量方法有香农熵、广义熵、条件熵、相对熵、信息增益和信息成本等。信息论应用于异常检测的途径主要是依靠计算流量特征的相互信息或熵值来识别异常分布。

2.3.3.1　熵

熵 (entropy) 在实验中可以当作是不确定性的度量，因为它是根据收到的每条消息中所包含的信息，平均后得出的量值，所以信息来源越随机，熵的值就越大。异常检测领域中，熵可以有效地将流量特征描述为分布，例如源/目的端口或 IP 地址，因为有某些类型的异常会对严重影响这些分布。通过这种方式，可以检测到例如由目的端口熵的变化表示的端口扫描攻击。

Behal 等人[?] 指出，由于 DDoS 攻击和突发事件会引起网络流量模式的大幅改变，而基于信息理论的熵或散度可以快速捕捉网络流量行为中的这种差异。因此，他们提出了一种利用流量之间的熵差进行异常检测的算法。通过采用了一组泛化的 ϕ-熵和 ϕ-散度，检测合法流量和攻击流量之间的信息距离。经过实验，该算法对于突发事件和 DDoS 的检测精度较高，而在其他数据集上表现一般。

David 等人[?] 提出了一种通过快速熵和基于流量的分析来增强对 DDoS 攻击的检测的方法。作者将观察到的流量汇总成一个单一的流量，并考虑到每个连接在一定时间间隔内的流量数，而不是取每个连接的数据包数。第二步基本上是计算每个连接的流量计数的快速熵。最后，根据快速熵和流量计数的均值和标准差生成一个自适应阈值。阈值随流量模式状况不断更新，提高了检测精度，同时快速熵的使用减少了计算处理时间。

2.3.3.2　KL 散度

KL 散度，又称相对熵，通常用于测量一个随机变量 X 的真实概率分布 P 与任意概率分布 Q(P 的近似) 之间的差异。设 $p(x)$、$q(x)$ 是离散随机变量 X 中取值的两个概率分布，则 p 对 q 的相对熵是：

$$D_{KL}(p||q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)} = E_{p(x)} \log \frac{p(x)}{q(x)} \tag{2-2}$$

在机器学习中，P 往往用来表示样本的真实分布，Q 用来表示模型所预测的分布。

Xie 等人[?] 利用 KL 散度着重检测了无线传感器网络 (WSN) 中的一种特殊类型的异常，这种异常会同时出现在邻近节点的集合中，并持续相当长的时间。基于节点的技术在此场景下效果和效率都不尽如人意。作者提出了基于分布式段的递归核密度估计，通过计算两个时间段的概率密度函数的差异，来判断是否发生了异常。同时也为了以较低的通信成本实现分布式估计，作者采用 KL 散度作为度量方法。利用真实世界的数据集对算法进行评估，结果表明，该算法可以以更低的通信成本实现很好的性能。

Li 等人[?] 以检测无线传感器网络中的异常数据值为目标，提出了一种基于差

分 KL 散度的异常检测方案。该方案首先将整个传感器网络划分为若干个簇，每个簇中的传感器在物理上相互接近，并且具有相似的感知值。然后，在每个簇内使用 KL 散度，以通过统计测量两个数据集之间的差异来检测异常值。他们的工作取得了良好的检测率和较低的误报率，同时比其他文献中的类似研究消耗更少的 CPU、内存等资源。

2.3.4　基于聚类的异常检测算法

Rajasegar 等人[?] 发明了一种聚类算法，是根据分布式超球面集群得出的，主要作用在于检查测试无线传感器网络中有无异常。检测过程是以聚类法进行建模，对象是每个端点的流量数据。建模后还要使用 k 个最近邻（KNN）集群的平均集群之间的距离去辨认出异常集群，辨认后就可以将数据向量分类为正常或异常。该算法的特点是操作过程在分布式系统中，通过传感器端点指出集群聚类的信息。在与其他端点传递信息之前，先凭借中间节点的先行合并，能使得通信开销降至最低。

K-means 虽然有局部收敛性和在簇中心节点选择上具有敏感性这些缺点。但是它仍然是一种经典的聚类技术，能够将数据划分为不同的类别。由于这种技术十分热门，有相当多的技术人员尝试着想借助其他技术来优化 k-means，去改善 k-means 的不足之处。Karami 等人[?] 就设计了一种基于“粒子群优化”(particle swarm optimization，PSO) 和 k-means 与局部优化混合的模糊异常检测系统，以确定最优的簇数。

Carvalho 等人[?] 开发了一种主动式网络监控系统，可以检测异常事件，减少决策中的人工干预和错误概率。他们提出一种创建网络基线轮廓 DSNSF（使用流量分析的网段数字签名）的方法。该方法通过修改蚁群优化算法，使用聚类方法描述正常的网络使用情况，该方法称为 ACODS。ACODS 在大量高维输入数据中，通过无监督学习机制优化提取行为模式，对网络流量发现进行表征。然后为了检测异常行为，他们首先计算每个时间区间内真实流量与正常曲线的相似度；然后计算序列之间的距离，并提供基于距离的测量方法。作者所提出的告警系统采用七种流量属性 (Bits，Bytes，Flows，Origin IP，Destination IP，Origin Port，Destination Port) 工作，利用熵来计算 IP 地址和端口特征的相关信息。当检测到异常时，ACODS 会提供一份包含 IP 流量信息的完整报告，说明每个属性对检测到的异常时间间隔的影响。ACODS 具有平方复杂度，导致解的收敛要经过多次迭代，作者试图通过使用局部搜索和信息素更新来缓解。

Dromard 等人[?] 所设计的无监督异常检测器，是建立在网格增量聚类算法和离散时间滑动窗口之上的。网格增量聚类在众多聚类算法当中效率拔群，因为后

者只更新之前的特征空间分区，而不是每当增加或删除很少的点时，就对整个空间进行重新分区。也因此，网格增量聚类的使用是在帮系统减少繁复性，并且使其在实时检测方面效能更强。最后系统合并这些更新的分区，用以识别最不相似的异常值。

Syarif 等人[?] 在 NSL-KDD 数据集上系统对比了各种聚类算法的性能，他们选出了 5 种应用最广泛的聚类算法：k-means、改进的 k-means、k-medoids、EM 算法和基于距离的异常检测算法。实验表明，在同等水平的误报率情况（约 21%）下，基于距离的异常检测算法效果最好，准确率为 80.15%，k-means 算法效果最差，准确率为 57.81%。

2.4　循环神经网络模型

经过数十年的发展，神经网络有非常多的种类，按照网络中是否包含循环可以将神经网络分为前馈神经网络和循环神经网络。

1. 前馈神经网络：前馈神经网络是一种单元之间连接不形成循环的神经网络。在这种网络中，信息从输入到输出正向流动。前馈神经网络如果只有一层输入节点、一层输出节点，不包含隐藏层，那么被称为单层感知器（Single Layer Perceptron）。若网络由多层计算单元组成，以前馈方式相互连接，则被称为多层感知器（Multi Layer Perceptron）。
2. 循环神经网络在循环神经网络 (RNN) 中，单元之间的连接形成了一个定向循环 (它们向前传播数据，同时也向后传播数据，从较后的处理阶段到较早的阶段)。这使得它能够表现出动态的时间行为。与前馈神经网络不同，RNNs 可以利用其内部存储器处理任意输入序列。这使得它们适用于未分割、连接的手写识别、语音识别和其他一般序列处理器等任务。

假设一个神经元接收 D 个输入 $x_1, x_2, ..., x_D$ 令向量 $x = [x_1; x_2; ...; x_D]$ 来表示这组输入，净输入也叫净活性值（Net Activation）。并用净输入（Net Input）$z \in \mathbb{R}$ 表示一个神经元所获得的输入信号 x 的加权和，

$$\begin{aligned} z &= \sum_{d=1}^{D} w_d x_d + b \\ &= w^T x + b \end{aligned} \tag{2-3}$$

其中 $w = [w_1; w_2; ...; w_D] \in \mathbb{R}^D$ 是 D 维的权重向量，$b \in \mathbb{R}$ 是偏置。

净输入 z 在经过一个非线性函数 $f(\cdot)$ 后，得到神经元的活性值（Activation）。

$$a = f(z) \tag{2-4}$$

其中非线性函数 $f(\cdot)$ 称为激活函数。

一个人工神经元的结构如图 2.2所示：

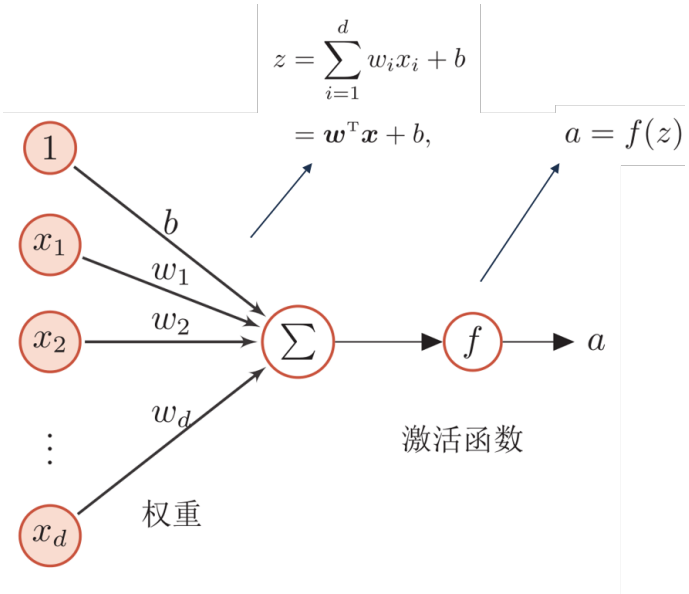


图 2.2　人工神经元模型

在图 2.2中，$\vec{x}$ 为输入向量，w 和 b 分别是权重和偏移。

神经网络主要由以下几部分组成：

- 输入节点（输入层）。在这一层中不进行任何计算，它们只是将信息传递给下一层（大部分时间是隐藏层）。
- 隐藏节点（隐藏层）。中间处理或计算在隐藏层中完成的，然后将输入层的权重（信号或信息）传递给下一层（另一个隐藏层或输出层）。一个神经网络也可以不包含隐藏层。
- 输出节点（输出层）。此层位于神经网络的最末层，负责接管来自前面隐藏层所输入的信息或信号。再通过激活函数，最终将得到合理范围内的理想数值，例如用于分类的 softmax 函数。
- 连接和权重。神经元之间会有边进行连接，每条边会有一定的权重。即每个连接将神经元 i 的输出传递给神经元 j 的输入，每个连接被赋予一个权重 W_{ij}。
- 激活函数。这个函数的作用在于将非线性特征引入到神经网络当中。同时它会将值的范围紧缩至更小，所以一个 Sigmoid 激活函数的值区间为 [0,1]。深度学习中有很多激活函数，如 Sigmoid、Tanh、ReLU 、Softplus、Softmax 等。表2.1为常见的激活函数。

表 2.1　激活函数

名称	表达式	导数
Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$	$f'(x)=f(x)(1-f(x))$
Tanh	$f(x)=\frac{2}{1+e^{-2x}}-1$	$f'(x)=1-f(x)^2$
ReLU	$f(x)=max(0,x)$	$f'(x)=\begin{cases}0 & x<0\\ 1 & x>=0\end{cases}$
Softplus	$f(x)=log(1+e^x)$	$f'(x)=\frac{e^x}{1+e^x}$
Softmax	$S_i=\frac{e_i}{\sum_j e_j}$	

- 学习规则。利用学习规则修改神经网络中的权重和阈值。

2.4.1　RNN

下图 2.3是循环神经网络的示意图[?]。该图显示了一个循环神经网络被展开成

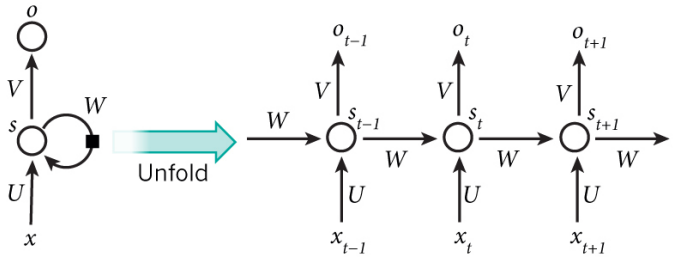


图 2.3　循环神经网络

一个完整的神经网络。例如，如果输入序列是时间窗口为 T 的一组向量，那么网络会被展开成 T 层的神经网络。用公式表示如下：

$$\begin{aligned} O_t &= g(V \cdot S_t) \\ S_t &= f(U \cdot X_t + W \cdot S_{t-1}) \end{aligned} \tag{2-5}$$

x_t 表示第 t 步的输入，例如 x_1 表示时刻 1 的特征向量。s_t 表示第 t 步隐藏层状态，也就是网络中的 “记忆”。获得 s_t 的过程是计算位于之前的隐藏层状态及当前层的输入向量。函数 $f(\cdot)$ 通常是一个非线性函数，如 tanh 或者 ReLU。

RNN 伪代码如算法2.1 所示，架构图如图2.4所示[?]。

2.4.2　LSTM

普通 RNN 有不能处理长依赖的问题，因此 Hochreiter 提出了一种长短期记忆网络-LSTM，LSTM 是一种特殊的 RNN，适用于学习长期依赖。现在 LSTM 已经被广泛应用于各个领域。如图2.5所示[?]，LSTM 和 RNN 类似，网络中都具有链式结构，但是 LSTM 中的循环单元与 RNN 中简单的 *tanh* 层不同，其构建了一些

算法 2.1 *RNN 伪代码*

输入： t 时刻的特征向量
输出： t+T 时刻的特征向量
1: 初始化 t 时刻单元状态
2: **for** t ← 1 to T **do**
3: $output_t = activation(input_t, state_t)$
4: $state_t = output_t$
5: **end for**

“门”（Gate）。利用构建出的“门”单元，能够将历史信息保留在当前节点状态，具体来说，是保留那些权重高的单元删除权重低的单元。

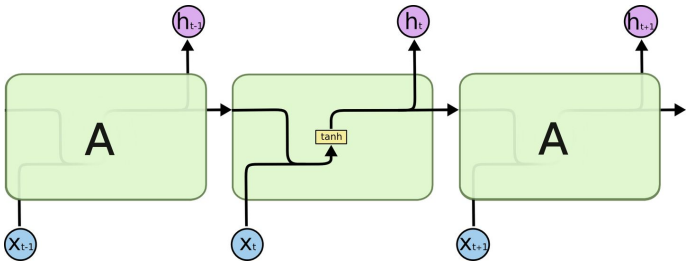


图 2.4　普通 RNN 结构

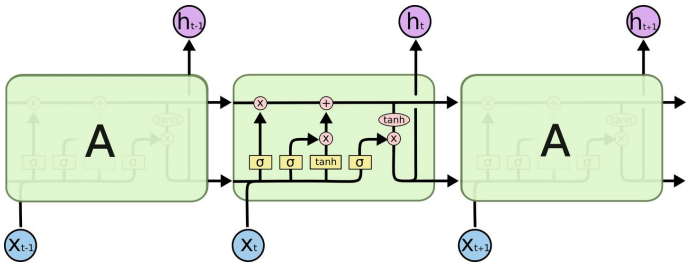


图 2.5　LSTM 结构

LSTM 首先通过 σ 层的“遗忘门”从单元状态中丢弃不重要的信息。遗忘门会读取上一时刻的输出 h_{t-1} 和当前时刻的输入 x_t，计算出一个维度为 n 的向量 f_t，该向量的值均在 $[0, 1]$ 之间。1 表示“完全保留”上个神经元的状态信息，0 表示“完全舍弃”。

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \tag{2-6}$$

下一步是确定该神经元的哪些新状态信息被存放在单元状态中。这里包含两个部分。第一，sigmoid 层，即“输入门层”，决定 LSTM 单元将更新哪些值。然后，tanh 层创建一个新的候选值 z_t 的向量，该向量可以加入到下一层单元状态中。

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2-7}$$

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2-8}$$

最后一步是将旧单元状态 c_{t-1} 更新为新状态 c_t。把旧状态与遗忘门 f_t 相乘，丢弃掉之前无需保留的信息，接着与新状态进行相加，综合得出该神经元的输出的状态，也即更新单元的状态。

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2-9}$$

算法 2.2 *LSTM 伪代码*

输入： 一组按时间排列的向量组
输出： 按时间排列的向量组

$\vec{C}_0 = \vec{0}$
$\vec{h}_0 = \vec{0}$
for $t \leftarrow 1$ to T **do**
    $output_t = activation(dot(W, input_t) + dot(U, state_t) + b)$
    $state_t = output_t$
    $i_t = activation(dot(state_t,))$
end for

2.4.3　GRU

门控循环单元（Gated Recurrent Unit，GRU）简化了 LSTM 模型，不仅合并了遗忘门和输入门，也合并了单元状态和隐藏层状态，在保证训练效果的同时大大减少了参数数量。

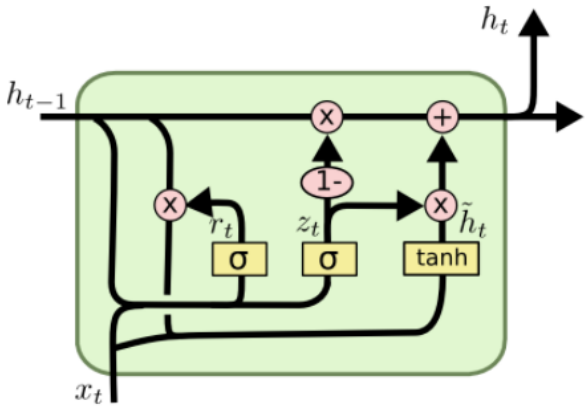


图 2.6　GRU 结构

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{2-10}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2-11}$$

$$\widetilde{h_t} = tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{2-12}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t} \tag{2-13}$$

由图2.6中结构可以看出[?]，GRU 是通过一个循环神经网络和 “门” 机制来不断更新内部参数。

2.5　开源数据集介绍

流量异常检测领域的开源数据集要么过于久远，无法反映当前的网络环境，要么模拟访问环境过于简单，可信度很低。因此流量异常检测领域的高质量的开源数据集很少。本文采用目前应用最为广泛的两个数据集，UNSW-NB15 和 CICIDS2017，尽管这两个数据集也有很多不足。

澳大利亚网络安全中心（ACCS）的网络范围实验室在 KDD99 数据集的基础上，生成了 UNSW-NB15 数据集[?]的原始网络流量数据。UNSW-NB15 利用的主要工具是 IXIA PerfectStorm，期望是可以体现网络在实际状况当中所呈现的流量。相比于陈旧的 KDD99 数据集[?]，其更能代表真实的网络流量。

第二个数据集 CICIDS2017 则是由加拿大网络安全研究所（Canadian Institute for Cyber-security）提供。与 UNSW-NB15 类似，CICIDS2017 也是在仿真环境中模拟正常流量和攻击流量生成的。

UNSW-NB15 的实验环境划分了 3 个子网，采用了 45 个独立的 ip 地址，持续约 30 个小时，采集了共 100GB 的原始网络流量数据。CICIDS2017 的实验环境划分了 2 个子网，分为受害者子网和攻击者子网，受害者子网共有 12 台主机，攻击者子网共 4 台主机，累计测量时间持续约 5 天，共采集了 51.1GB 的 pcap 流量数据。由这两者的实验环境可以看出，开源数据集在规模上远远无法和真实流量环境对比。

以 UNSW-NB15 为例，图2.7展示了该数据集的生成过程。首先，通过使用 Tcupdump 工具监测仿真环境中的流量情况，生成 pcap 文件；然后使用 Argus①和 Bro-IDS②工具从报文信息中直接提取基于数据包和数据流的特征，并根据五元组

① https://qosient.com/argus/index.shtml
② https://www.bro.org/index.html

（源/目的 ip 地址，源/目的端口号，协议类型）信息进行匹配，此时基本特征、内容特征和时间特征已生成；接下来为了能够更有效地识别攻击，还需要额外生成一些统计特征。最终将这些特征保存成 csv 文件。

这些数据集生成流程和特征提取方法给后续我们分析清华大学校园网的流量数据提供了参考。

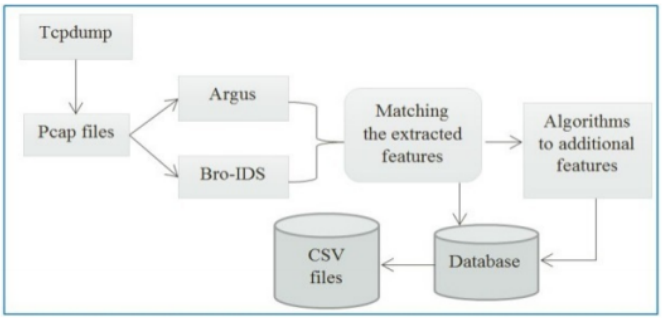


图 2.7　UNSW-NB15 数据集生成过程

2.6　校园网数据集介绍

清华大学无线网规模巨大，具有超过 6 万名日活跃用户，13 万个可用 ip 地址，是全球最大的校园无线网之一。通过出口网关的服务器我们可以得到海量的真实流量数据，但是这与可进行分析和训练的数据集之间还有巨大的鸿沟。因此本节参考图2.7中的数据集生成流程，结合 UNSW-NB15 和 CICIDS2017 两个数据集中特征的特点，生成了校园网的特征数据集。校园网数据集制作流程如图2.8所示。首先将 pcap 流量数据切分成多个会话（session，双向流中所有的数据包），依次从每个会话中提取报文信息以及统计信息构成特征；然后以五元组作为流的标识符，把从 SOC 平台中得到的告警信息与特征信息合并起来，就生成了最终的数据集。将该数据集命名为 CAMPUS 数据集。

在本文的实验中，我们采集了三段共约 25 小时的数据，第一段从 2021 年 3 月 19 日 8 点 43 分采集到 2021 年 3 月 19 日 12 点 08 分，持续 3.5 小时，共 420GB 大小，用于验证特征提取算法以及经典模型的实验对比，第二段从 2021 年 3 月 19 日 13 点 06 分采集到 2021 年 3 月 20 日 6 点 52 分，持续 18 小时，共 1100GB 大小，用于分析特征关系图的有效性以及训练 FG-RNN 算法，第三段从 2021 年 3 月 20 日 14 点 26 分采集到 2021 年 3 月 20 日 18 点 21 分，持续 4 小时，共 350GB 大小，用于验证 FG-RNN 算法的鲁棒性。

以上数据为出口网关处的全流量数据，此外我们还利用了安全管理平台（Security Operations Center，SOC）的威胁告警日志，接下来分别介绍这两类数据。

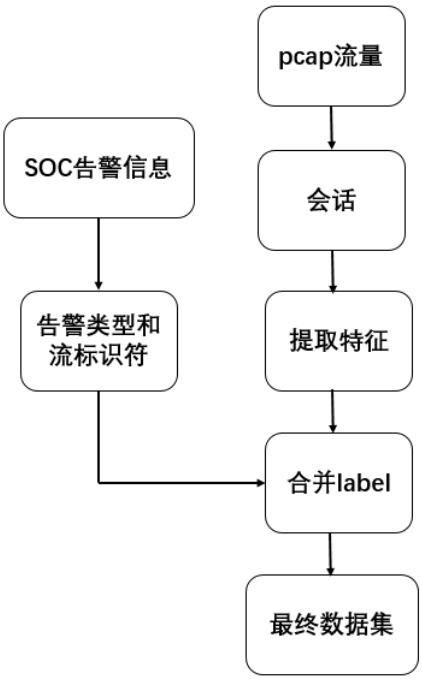


图 2.8　校园网流量数据集制作流程

2.6.1　全流量日志数据

校园网的原始数据为抓包（Packet Capture，pcap）流量，如下图 2.9所示，其中包含每个数据包的详细信息，如五元组、报文头部信息、报文内容等。清华大学校园网共有 6 个 B 的地址资源，由于本实验的计算和存储资源的限制，我们只采集了一个 B 类地址的流量数据，最多可容纳 6 万台主机，相比于开源数据集采集环境的数十台主机，本数据集规模可谓宏大。

另外，本文对所有用户隐私信息（如 IP 地址和 MAC 地址，报文中的明文信息等）都进行了匿名化处理，保证不会泄露用户的任何隐私信息。

全流量数据宏观上看具有周期性，例如图1.2表示每秒钟报文数量的变化。但是由于我们是基于每条流的信息进行异常检测，平均每秒钟数百万的报文信息之间大多毫无关联，因此仅需从流的视角对全流量数据进行特征提取。本文采用了 CICFlowMeter[①]工具，其大致原理为从 pcap 文件中依次读取报文（packet），判断当前报文是否属于当前统计的所有流中，若存在则更新该流的统计特征信息，否则创建一条新的流，直至遇到包含 FIN 标志的报文或者超时。最后将得到的每条流的特征信息依次打印到 csv 文件中。

① https://github.com/ahlashkari/CICFlowMeter

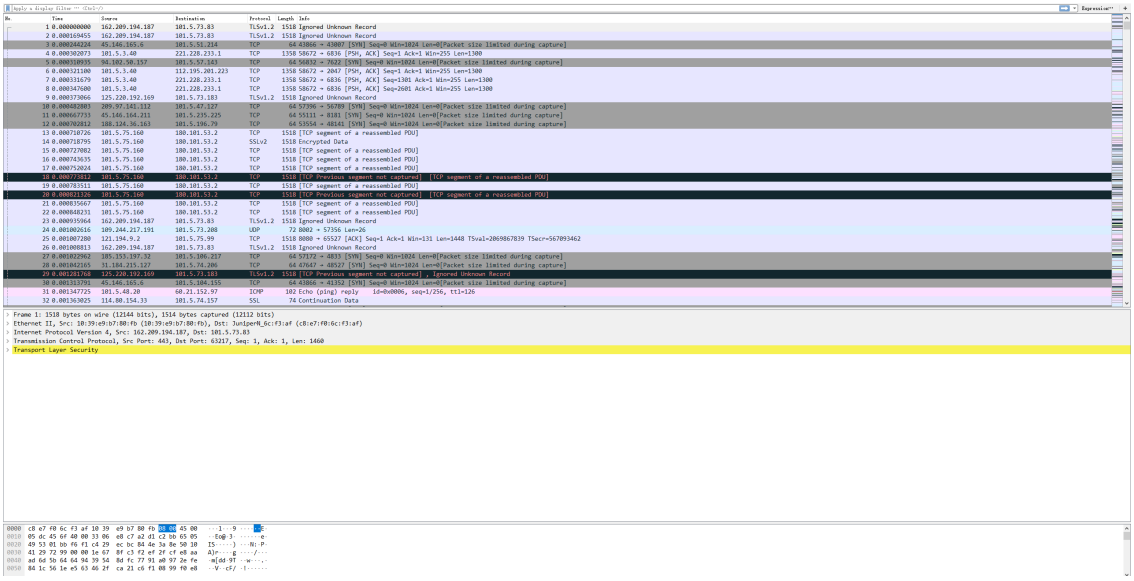


图 2.9　流量数据示意图

2.6.2　威胁告警日志

我们通过全流量日志得到了特征数据集，但是由于缺乏标注，该数据集无法进行训练以及验证。因此为了得到可进行训练的有效数据集，我们还需要对特征数据集添加标注。我们根据现有的 SOC 平台得到标注数据，然后进行数据清洗、数据预处理等步骤。图2.10为 SOC 平台的数据样例。

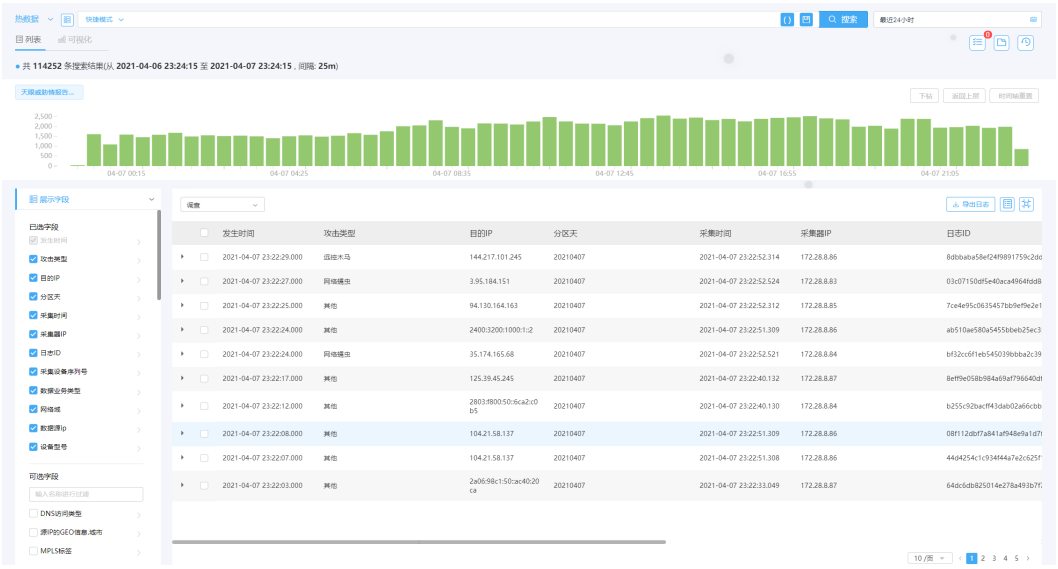


图 2.10　SOC 平台数据样例

图2.11为 SOC 平台给出的一个月内告警信息的趋势图。从图中可以看出，告警信息大致稳定在每天 9000 个，存在极端情况。

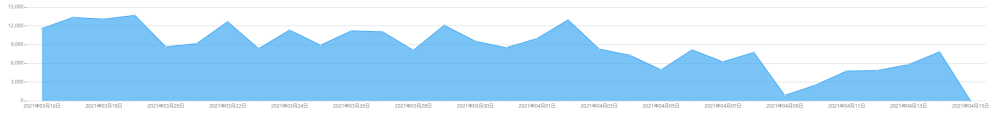


图 2.11　告警趋势图

2.7　两类数据集对比

在对两大类数据集进行异常检测的算法验证之前，我们首先从数据规模和异常种类两方面对其进行了对比，以便对 CAMPUS 数据集有一个直观的印象。

表2.2是三种数据集在规模上的对比，从中可以看出，CAMPUS 数据集规模宏大，子网内 ip 数量和使用的流量文件大小是开源数据集的百倍甚至千倍，并且 CAMPUS 中异常种类也要比开源数据集更多。

表 2.2　数据集规模对比

数据集名称	独立 ip 数量	采集时长	流量文件 (pcap) 大小	异常种类
UNSW-NB15	45	30h	100G	9
CICIDS2017	16	5 天	51G	4
CAMPUS	数万	25h	2T	10 多种

图2.12展示了三种数据集在异常类别上的对比，可以看出 CICIDS2017 数据集中的异常大部分是拒绝服务和端口扫描，分别占比约 70% 和 28%，其中包括多种子类型，例如慢速连接攻击、SYN 洪泛攻击、黑市工具等；而 UNSW-NB15 数据集约 67% 的异常是暴力猜解；CAMPUS1 和 CAMPUS2 分别是 CAMPUS 数据集的两个时间切片，异常的种类比开源数据集更加丰富。并且尽管它们同属于一个数据集，但是由于处于不同的时间窗口内，异常种类的占比差异很大。而反观开源数据集，由于全部流量都是人工生成，其异常种类和占比都是固定不变的。这种异常种类的动态性给我们的异常检测算法带来了挑战，因此算法的适应性需要足够强，面对复杂的流量环境依然可以有效地检测出异常。

2.8　现有算法在不同数据集上的对比

本文利用 sklearn、pytorch 等 Python 库分别实现了“逻辑回归（LR）”、“决策树”（DT）、“随机森林（RF）”、标准的“循环神经网络”（RNN）、“长短期记忆网络”（LSTM）、“门控循环单元”（GRU）几个模型。

本节分别在以上三个数据集上进行实验，对比了以上 6 个经典模型，表2.3是

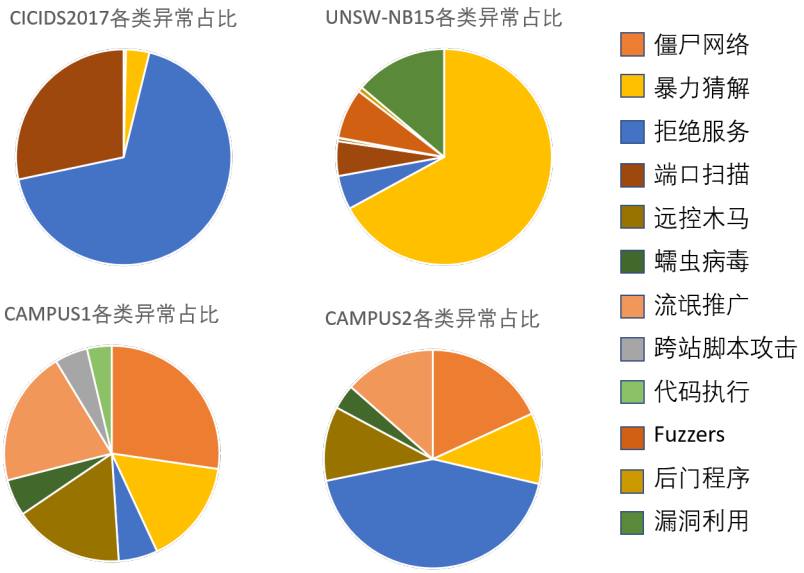


图 2.12　数据集异常类别对比

评估结果，评估指标为正样本的准确率，其中多分类任务的准确率指标为对于每个标签，分别计算 precision，然后取不加权平均值。由表中可以看出，在两个开源数据集上，各个算法在二分类任务（即仅需检测是否发生异常）的效果相差不大，在多分类任务（即需要判断异常类型）里，神经网络类的模型效果均要略好于基于分类的机器学习模型。这说明针对模拟仿真出的有规律的流量数据，各类算法可以有效地学习出正常流量的基线，而不同类型的异常具有各自的特征，各个算法的能力在学习这些特征的过程中显现出差异。但是这些算法在 CAMPUS 数据集上表现很差，甚至和随机猜测的效果差不多。说明现有的算法无法应对复杂的清华大学校园网流量环境，需要我们进一步对这些算法进行优化，以适应校园网流量环境。因此，接下来我们以 DT 和 RF 模型作为 baseline，对 RNN 类算法进行特征分析，找到其表现不佳的原因，并且做出改进。

2.9　现有算法存在的问题

截至目前，异常检测算法的种类已经不可胜数，其中所使用的原理都有差别，针对的领域也各异，流量特征自然也不一样。但通过本文的分析我们可以看出，在应对校园网大规模、应用类型多、异常流量是常态且多种异常相互叠加的场景下，现有异常检测算法大多仍存在以下缺陷：

1. 应用类型多导致流量特征复杂，因此很难达到较高的检测率和较低的误报率；
2. 不管是传统机器学习算法还是拟合能力更强的深度学习模型，都很难有效地

表 2.3　不同数据集下实验评估结果 (%)

数据集	任务	LR	DT	RF	RNN	LSTM	GRU
UNSW-NB15	二分类	96.8	97.33	98.52	95.51	96.74	94.91
	多分类	90.73	91.36	91.82	93.98	92.11	92.30
CICIDS2017	二分类	98.26	97.11	95.54	90.18	88.04	91.49
	多分类	91.01	92.37	94.41	92.31	93.16	94.18
CAMPUS	二分类	76.98	77.95	77.51	55.62	59.80	55.25
	多分类	73.33	74.01	74.54	53.01	56.59	59.28

从复杂的流量环境中学习到正常流量的基线；

3. 面对海量数据规模时，现有的算法模型无法或很难做到实时性。

根据以上几点缺点，本文后续章节提出了一种基于特征关系图的神经网络算法，并且在此基础上实现了一个流量异常检测系统。

2.10　本章小结

本章首先对网络流量异常检测领域的相关工作进行了综述，然后介绍了常用的异常检测数据集和本文中使用的校园网数据集，最后在多个数据集上对异常检测算法进行了分析与对比。

第 3 章　基于特征关系图的循环神经网络算法

3.1　引言

从第三章的实验可以知道，现有的异常检测算法能够在公开数据集取得较为不错的表现，但是在真实数据集——清华大学校园网流量数据上效果均无法达到要求。这是因为相比于人工构造的公开数据集，清华大学校园网数据具有规模大、应用类型种类繁多、异常流量占比多、难以学到基线等特点。而 RNN、LSTM、GRU 等拟合能力强大的深度学习模型在校园网数据集中有着精确率低、误报率高的问题。因此本章将先从特征分析着手，分析 RNN 模型效果不佳的原因，然后引入特征之间的关系矩阵，将其加入到神经网络的训练中，从而获得更好的检测效果。

3.2　特征分析

模型可解释的分析方法通常有两类：

1. 通过数学推导来解释每个特征如何发挥作用，例如线性回归中的系数，SVM 中的支撑平面等，这类方法旨在分析因果关系。
2. 针对神经网络等“黑盒”模型，通过分析特征如何影响最终预测结果，来部分解释模型的原因，这类方法旨在分析关联关系。

本文中采用第二类方法，使用 Shapley 值这一工具来尝试解释 RNN 类模型在 CAMPUS 数据集中效果不佳的原因。

Shapley 值起源于 1953 年的博弈论[?]，是为了解决这样一个问题：一群拥有不同技能的参与者为了集体奖励而相互合作。那么，如何在小组中公平分配奖励？在机器学习领域，参与者就是输入的特征，而集体奖励则是模型输出的结果。Shapley 值可用于计算每个特征对于模型输出的贡献。其公式如下：

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v(S \cup \{i\}) - v(S)) \tag{3-1}$$

其中，N 是所有特征构成的集合，S 是 N 的子集，$v(\cdot)$ 可以认为是模型，$\phi_i(v)$ 即为第 i 个特征的“重要性”。

本文利用 python 中的 shap 库[①]分别计算了 RNN 模型在三个数据集中的 Shapley 值，并从 SHAP 特征重要度、SHAP 摘要图两方面进行了可视化分析。

SHAP 特征重要度的原理是 Shapley 绝对值越大，其对应的特征越重要。由于

① https://github.com/slundberg/shap

公式 4-1 中计算得到的是单个实例的 Shapley 值，因此我们需要对每个特征的全部实例取平均值：

$$I_j = \sum_{i=1}^{n} |\phi_j^{(i)}| \tag{3-2}$$

图 4.1 和图 4.2 分别是 DT 和 RF 模型在 UNSW-NB15 数据集中的特征重要度排序，图 4.3 是 RNN 模型在 CAMPUS 数据集中的特征重要度排序。从中可以看出，两种树模型的前 20 个重要特征的重合度很高，都利用了 sttl（源报文的 ttl 字段）、sbytes（会话中从源发出的总字节数）、synack（第一次发送报文到收到确认报文的时间）等特征，这些“强特征”对于模型判断某条流是否为异常提供了重要信息。而 RNN 模型在 CAMPUS 数据集上仅利用了极少数的特征。

图 4.4-4.6 是 SHAP 摘要图。SHAP 摘要图将特征影响和特征重要度结合起来。摘要图上的每个点都是一个特征和一个实例的 Shapley 值，y 轴上的位置由特征决定，x 轴上的位置由 Shapley 值决定，颜色代表特征值从小到大，重叠点在 y 轴方向上抖动，因此我们可以了解每个特征的 Shapley 值的分布。例如 sttl 这一特征，红色说明特征值很高，在图中 sttl 很高时，其 SHAP 值为 0，说明对预测结果影响很小，这点也符合我们的认知。从图中对比可以看出，RNN 模型的大部分特征的 SHAP 值都为 0。

通过本节的分析，我们得出以下结论：

1. SHAP 特征重要度和摘要图可以一定程度的解释模型。
2. 对比不同的数据集，树模型中 top20 重要性的特征相似，说明这些“强特征”更能影响检测结果。
3. 在 CAMPUS 数据集中，相比于其他模型，RNN 仅利用到了少部分特征信息。

3.3　特征关系

流量特征可以视为反映当前网络流量的一组观察值，从多个角度描述当前的流量场景。好的特征能够真实地反映出流本身的状态信息。例如对于一条单个 TCP 流，这条流可能是一个正常的端到端的连接，也可能是分布式拒绝服务攻击的一部分，在被攻击者的视角内，它的入流量的带宽急剧增加，且远大于出流量的带宽。图3.7表示某一时刻下部分特征之间的关系，从中我们看出，发送方/接收方数据包长度大小与流的传输速率具有很强的关联性。

我们分析了暴力猜解攻击的部分特征信息，从表3.1中可以看出，相比于正常流量，当发生暴力猜解攻击时，其从源报文发出的总字节数（sbytes）和源报文的间隔时间的平均值（sinpkt）都非常小。在这种场景下，sbytes 和 sinpkt 的关联性更

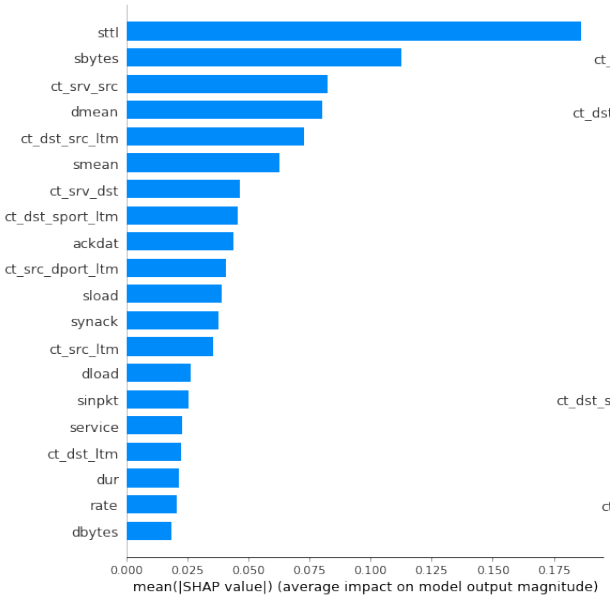


图 3.1　DT 模型的特征重要度 top20

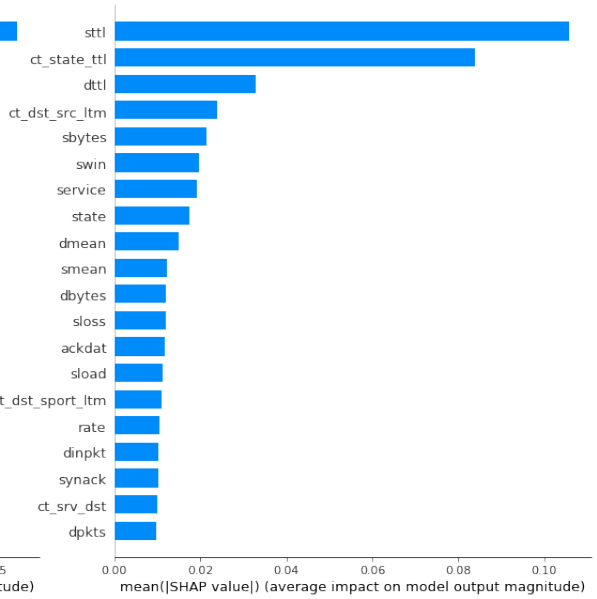


图 3.2　RF 模型的特征重要度 top20

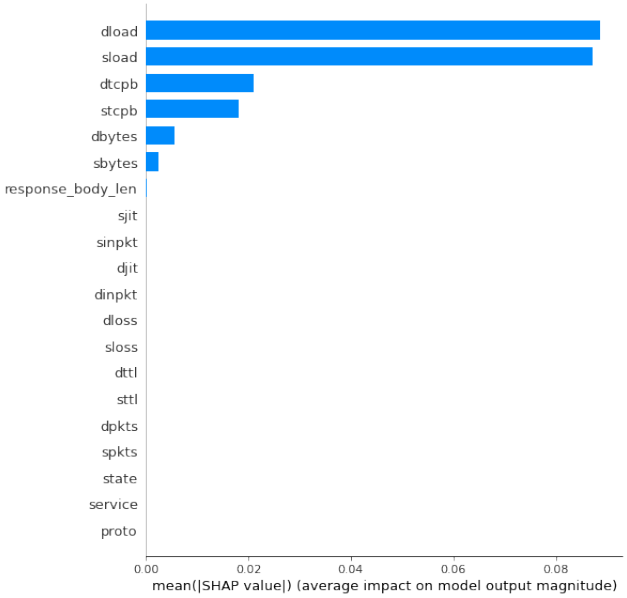


图 3.3　RNN 模型的特征重要度 top20

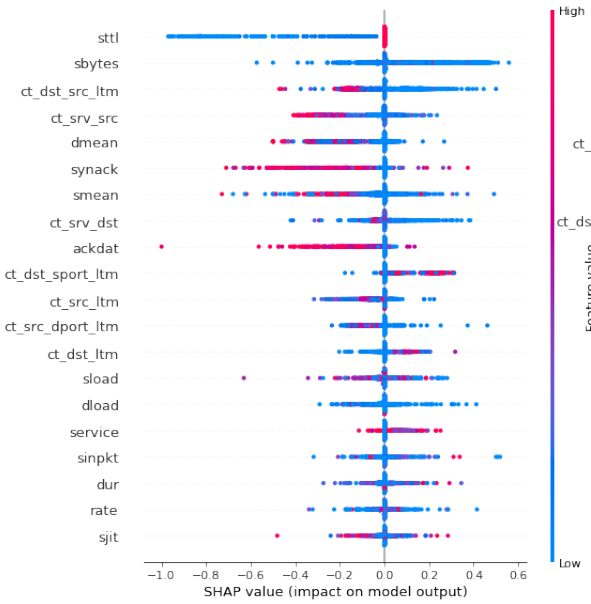


图 3.4　DT 模型在 UNSW 中的摘要图

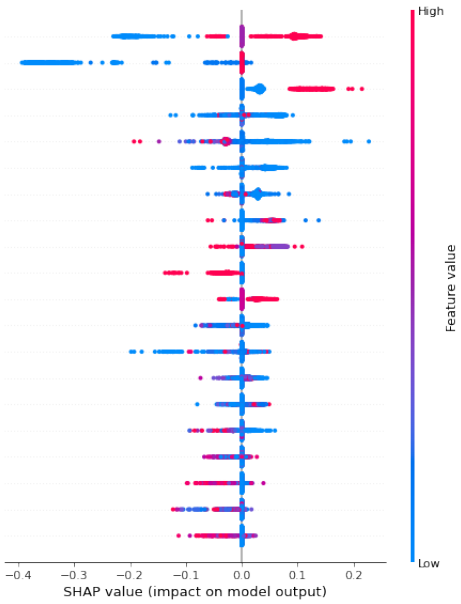


图 3.5　RF 模型在 UNSW 中的摘要图

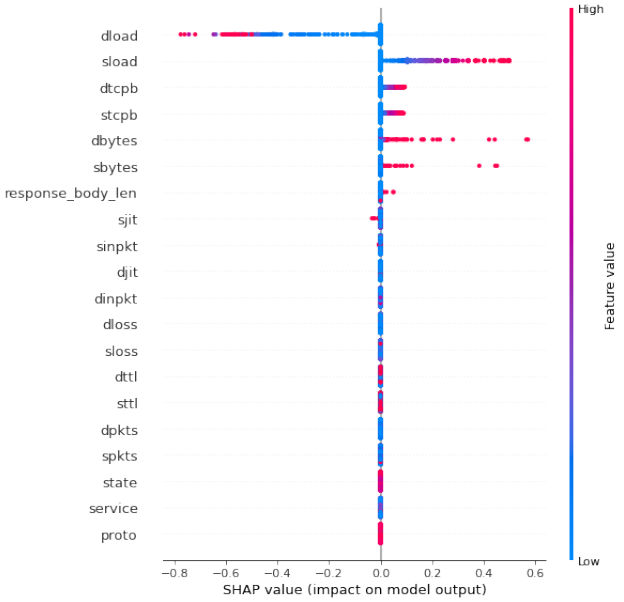


图 3.6　RNN 模型在 CAMPUS 中的摘要图

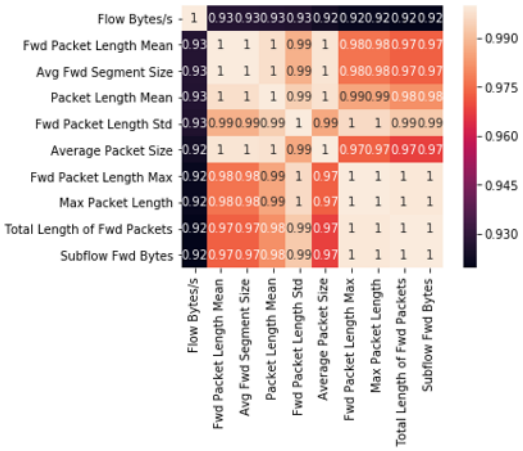


图 3.7　特征关系

强，而在正常流量场景下两者的关联性更弱。

表 3.1　暴力猜解 case 分析

dur	proto	sbytes	dbytes	sinpkt	dinpkt	...	attack type
0.216048	113	990	2038	24.005333	28.037572	...	Normal
0.824394	113	1054	1344	43.335262	42.801578	...	Normal
0.000007	119	114	0	0.007	0	...	Generic
0.000007	119	114	0	0.007	0	...	Generic
0.000005	119	120	0	0.005	0	...	Generic

由上节的分析可知，在 CAMPUS 数据集中，RNN 仅利用了极少数的特征信息，为了捕获更多特征信息，因此需要挖掘特征之间的潜在关联。从本质上来说，各种机器学习、深度学习模型就是挖掘特征信息的过程。然而针对 CAMPUS 数据集，RNN 模型无法有效提取信息，通常的做法有两种：

1. 从特征工程着手，进行特征组合或者特征交叉（Feature Crosses）。如果直接进行特征组合，则复杂度过高，需要 $O(N^2)$ 的计算复杂度，并且导致维度爆炸。因此常用的方法有因子分解机（Factorization Machines），对于每个原始特征，FM 都会学习一个隐向量。FM 通过穷举所有的特征对，并进行逐一检测特征对的权值来自动识别出有效的特征组合。
2. 从模型结构着手，在神经网络中增加额外辅助信息。例如机器翻译领域往往会加入文章的关键词、摘要等信息。这部分信息会影响到模型对于神经元参数的计算，从而影响特征权重。

这两种做法各有优劣，方法 1 可以在线性时间内完成，但是只能对特征进行两两组合，并且组合信息难以进行解释；方法 2 能够捕捉到更多的关联信息，因此会增加神经网络的复杂性，带来额外的计算开销。由于本文使用的模型为预训练模型，无需在线实时训练，相比于几小时的模型训练时间，这部分计算延迟可以忽略，我们采用方法 2 挖掘特征潜在关系。

为了在神经网络中引入特征关系矩阵，本文利用皮尔逊相关系数来计算流量特征之间的关系。皮尔逊相关的别名是积差相关（或者称之为积矩相关），命名来源是 20 世纪英国的描述统计学派先驱皮尔逊。对于两个变量 X、Y，它们之间的皮尔逊相关系数为：

$$\rho_{X,Y} = \frac{cov(X,Y)}{\rho_X \rho_Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}} \tag{3-3}$$

其取值范围为 [-1,1]，$\rho_{X,Y} = 1$ 时，说明 X 和 Y 完全正相关，$\rho_{X,Y} = -1$ 时，说明 X 和 Y 完全负相关，$\rho_{X,Y}$ 接近 0 时，说明 X 和 Y 无线性相关性。

3.4　基于关系图结构的 RNN

由前两节分析可知，LSTM 和 GRU 作为两种循环神经网络，都可以很好地提取时序相关性。但是在复杂的清华大学校园网流量环境下，仍然有很多改进空间。根据第 3 章的实验结果，LSTM 和 GRU 在 CAMPUS 数据集下准确率仅为 65.4% 和 67.8%。经过特征分析，我们发现不同时刻下特征之间的相关性会发生变化。因此本文中我们利用循环神经网络（RNN）对时间依赖性进行建模的同时对特征关系也进行建模。值得指出的是，本文中使用的 RNN 子类是 “门控循环单元”（GRU），相比于普通 RNN，它可以很好地捕捉到时序数据中相隔较远的依赖关系。在 GRU 的矩阵乘法中，我们加入了前文提到的特征关系图（Feature Graph）。

定义无向权重图 $G = (V, E, W)$, 其中 V 表示特征节点的集合，$|V| = N$；E 表示特征间的关联关系，即图中的边；$W \in R[N * N]$ 为特征节点的相似度的加权邻接矩阵，我们称 W 为特征关系矩阵。将时刻 t 观察到的流量特征向量表示为 X^t。那么流量预测目的就是：给定 G 下，学得一个函数将 T 个历史图信号映射到未来 T 时刻的图信号：

$$h[X^{t-T+1}, X^{t-T+2}, ..., X^t; G] \Rightarrow [X^{t+1}, X^{t+2}, ..., X^{t+T}] \tag{3-4}$$

$$r^{(t)} = \sigma(\Theta_r \star G[X^{(t)}, H^{t-1}] + b_r) \tag{3-5}$$

$$u^{(t)} = \sigma(\Theta_u \star G[X^{(t)}, H^{t-1}] + b_u) \tag{3-6}$$

$$C^{(t)} = tanh(\Theta_C \star_G [X^{(t)}, (r^{(t)} \odot H^{(t-1)})] + b_c) \tag{3-7}$$

$$H^{(t)} = u^{(t)} \odot H^{(t-1)} + (1 - u^{(t)}) \odot C^{(t)} \tag{3-8}$$

其中 $X^{(t)}$, $H^{(t)}$ 表示在时间 t 的输入和输出，$r^{(t)}$ $u^{(t)}$ 分别是在时间 t 的复位门和更新门。$\star G$ 表示扩散卷积，并且是对应滤波器的参数。与 GRU 相似，该模型可用于构建递归神经网络层，并使用反向传播进行训练。FG-RNN 算法的原理如图3.8所示。

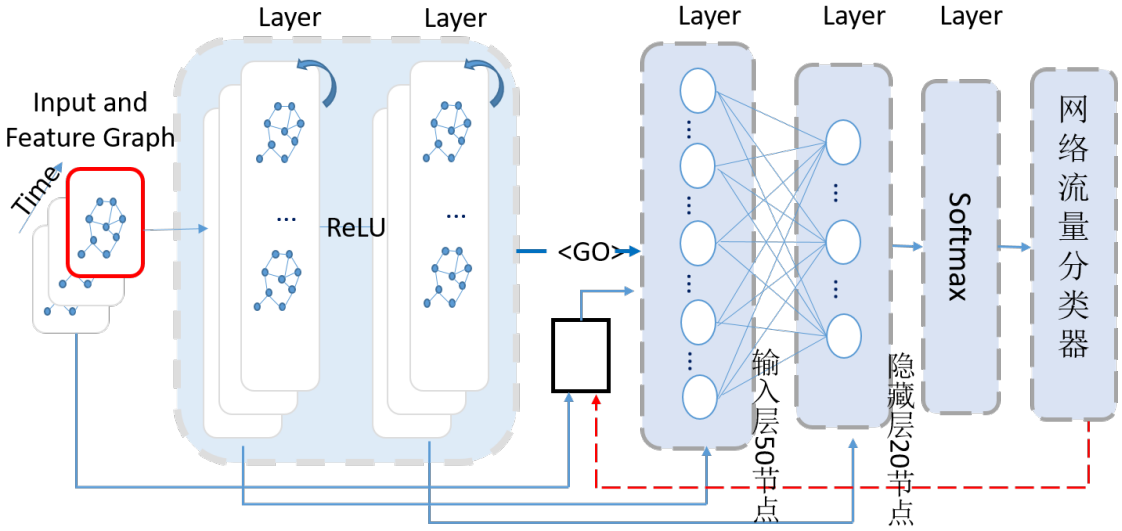


图 3.8　FG-RNN 原理图

FG-RNN 算法的伪代码如算法3.1所示。本算法的核心代码已上传到 github 平台[①]。

通常，计算卷积会很费时。但是，如果 G 很稀疏，则可以使用总时间复杂度 $O(K \mid \varepsilon \mid) \ll O(N^2)$ 递归稀疏矩阵乘法来有效地进行计算，本文中计算特征关系图和 GRU 都可以使用该矩阵乘法。

—

在多步预测中，我们采用了 seq2seq 结果 (Sutskever et al，2014)。编码器和解码器都是具有 DCGURU 的递归神经网络。在训练过程中，我们将历史时间序列输

① https://github.com/GTnull/FG-RNN

算法 3.1 *FG-RNN 伪代码*

输入： 一组按时间排序的向量组 $\vec{x_1},\vec{x_2},...,\vec{x_T}$, 特征间关系图 G 以及它的邻接矩阵 W
输出： 按时间排序的向量组 $\vec{h_1},\vec{h_2},...,\vec{h_T}$

通过 Xavier 方法初始化 q_{net1}, q_{net2} and p_{net} 的所有参数
while $\mathcal{L}$ 没有收敛 **do**
    利用公式计算每一个节点 v 的 q_{net1}, q_{net2} and p_{net}
    for i ← 1 to T **do**
        更新特征关系矩阵 $W(e)$ 的系数
        更新 GRU 中的参数 $W(X)$
        计算 $tanh(W(e) + W(X) + b)$
    end for
    用公式计算目标函数 $\mathcal{L}(\theta, \phi)$
    用梯度下降法更新 q_{net1}， q_{net2} 和 p_{net} 的所有参数
end while

入编码器，并使用其最终状态来初始化解码器。解码器根据先前的观测值生成预测。在测试时，真实值将由模型本身生成的预测值代替。训练集和测试集分布之间的差异会导致性能下降。为了解决这个问题，我们将预采样 (Bengio et al,2015) 集合到模型中，其中，我们向模型提供概率为 epsiloni 的真实观测值或者第 i 次迭代时模型以概率为 1−ϵi 进行的预测。在训练过程中，ϵi 逐渐减少到 0，以允许模型学习测试分布。

加入特征图后，新的神经网络是否能够收敛？我们从理论上证明了这点。FGRNN 的收敛性证明，参考了部分图神经网络的证明过程。在图神经网络中，目标节点的特征是根据它邻近节点的特征计算而来。

我们认为收敛的网络就是计算可达的网络。计算可达就是可以通过计算的方式，把输入数据输出成人们指定的结果。以图像识别为例，输入数据是一张图，输出数据是判断这幅图中是否有人/猫/狗等物。计算的载体则是深度学习模型。如果该网络模型经过海量数据学习，可以完全正确地完成图片分类任务，那么这个网络就是计算可达的。假设存在一个理想化的映射 $f(\cdot)$，对任何输入的图像，它都能给出一个正确的分类结果。将我们的模型学习出来的复杂神经网络抽象为映射 $g(\cdot)$。计算可达就是指 $g(\cdot)$ 无限接近于 $f(\cdot)$，即 $||f(\cdot) - g(\cdot)|| \to 0$。

GNN 中的 n nn 表示图神经网络计算的结果是每个节点都会获得一个特征向量。论文中也提及了 GNN 从 GNN map 计算模型推导：

$$\begin{aligned} x_i^{(l)} &= Up_l\Big(\sum_{v_j \in N_i^* m_{i\leftarrow j}^{(l)}} \Big) \\ &= Up_l\Big(\sum_{v_j \in N_i^*} MSG_l\big(x_i^{l-1}, x_j^{l-1}, v_i, v_j, a_{i\leftarrow j}\big)\Big) \\ &= AGG_l\Big(\big\{\big(x_i^{l-1}, x_j^{l-1}, v_i, v_j, a_{i\leftarrow j}\big) : v_j \in N_i^*\big\}\Big) \end{aligned} \tag{3-9}$$

其中 AGG 是聚合函数，可以理解为是 MSG 和 UP 的融合形式。

从 LOCAL 计算模型推导：

$$\begin{aligned} s_l^{(l)} &= ALG_l^1\left(\left\{\left(s_{i\leftarrow j}^{(l-1)}, a_{i\leftarrow j}\right) : v_j \in N_i^*\right\}, v_i\right) \\ &= ALG_l^1\left(\left\{\left(\left(ALG_l^2\left(s_j^{(l-1)}, v_j\right)\right), a_{i\leftarrow j}\right) : v_j \in N_i^*\right\}, v_i\right) \\ &= ALG_l\left(\left\{\left(s_j^{(l-1)}, v_i, v_j, a_{i\leftarrow j}\right) : v_j \in N_i^*\right\}\right) \end{aligned} \tag{3-10}$$

再讲具体证明之前，咱们先理清一下逻辑：

证明的内容：FGRNN 在条件 X 下是计算可达的，X 是未知的；假设我们掌握一个这样的已知条件：已知条件：计算模型 LOCAL 在条件 Y 下是计算可达的；Y 是已知的；其中 LOCAL 是一个计算复杂度更高的理想化网络模型，是细胞间传递信息的生物模型。我们只需要试图构造一个条件 Z，在条件 Z 下，FGRNN 和 LOCAL 是等价的。这样，FGRNN 计算可达的条件就是 Y+Z。证明思路如下：第一步：证明在条件 Z 下，FGRNN 与 LOCAL 模型等价。第二步：根据 LOCAL 的性质，证明在满足条件 Y 的情况下，LOCAL 计算可达；第三步：FGRNN 计算可达的条件就是 Y+Z。

这是 LOCAL 的性质，这里不去做证明，直接拿来用。 指的是图中最长的最短距离（the length of the longest shortest path）。最大的最短距离，看上去有的矛盾哈。先理清楚一个图中任意两节点的最短距离定义 shortest path。遍历一个图中所有的点对，计算该点对的距离，把所有点对的距离都由大到小地排列在一起，那么最前头的最短距离就是最大的最短距离。宽度没有界就是 w。

那什么是宽度 w 没有界呢？就是。宽度 w 指的是图神经网络中神经元中最大的特征维度。举个例子，w = 64 w=64w=64，说明有一层的神经元个数是 64 个。w 则说明有一层的神经元个数是极其大的。神经元个数越多，网络的表示能力就会越强。当 w，该网络在理论上是图灵完备的。一个图灵完备的网络对应 LOCAL 中图灵完备的运算符 ALG 深度 $d \geqslant G$ 这个条件也可以去解释。当 d ≥ δ 图中任一一个节点都可以接收到图中其他任意节点的信息。当然 $d \geqslant G$ 是一个很强的条件。

第三步（结论）：GNN 计算可达的条件是：MSG 和 UP 是图灵完备函数，且深度 $d \geqslant G$ 并且宽度没有界。

把前两步的结果拼起来即可。注意这个结论是一个必要条件（必要条件!）：根据 MSG 是图灵完备函数，且深度 G 并且宽度 w ww 没有界，可以去得到 mpn 计算可达。（正确的结论）但论文可没有下这样的充分条件哦：mpn 计算可达一定要 MSG 是图灵完备函数，且深度 G 并且宽度 w ww 没有界。（错误的结论）我们着重解释一下这个结论。首先，这个结论只是个理论上的结论。G 是一个非常强的必要条件。它是必要条件哈。满足 dG 和其他条件，那么 GNN m pmpn 计算可达。

如果 d < δ GGGNN m p mpn 也未必是不收敛的。宽度 w ww 没有界同样是一个非常强的必要条件。而且从实际角度出发，是难以去训练的，需要足够大内存和海量的数据以及充分的训练时间等等。因此，这仅仅是一个理论结论。

3.5　在 RNN 中引入特征图信息

在 RNN 的训练过程中，加入有效的额外辅助信息往往能够提升训练效果，例如在机器翻译领域，加入文章的关键词、摘要、作者信息等。通常加入信息的方式有以下几种。

1. 直接将额外信息向量与当前特征向量叠加。原向量为 $p = (p_1, p_2, ..., p_n)$，额外信息向量为 $e = (e_1, e_2, ..., e_n)$，最终输入向量为 $w = (p_1 + e_1, p_2 + e_2, ..., p_n + e_n)$。这种方法需要保证额外信息向量与原向量维度相同。
2. 将额外信息向量与当前特征向量拼接。原向量为 $p = (p_1, p_2, ..., p_n)$，额外信息向量为 $e = (e_1, e_2, ..., e_m)$，最终输入向量为 $w = (p_1, p_2, ..., p_n, e_1, e_2, ..., e_m)$。也就是增加输入的维度，缺点是通常要求额外信息的特征与原特征类型保持一致，例如均为词向量。
3. 增加一个额外的隐藏层，分别使用不同的矩阵进行变化，将结果用 *tanh* 函数映射到所需的维度。相当于增加一个普通循环神经网络模型和额外信息模型的感知器，然后加载到输出层上。即：

$$h'_t = tanh(W(p_t) + W(e) + b^{h'_t}) \tag{3-11}$$

因此，本文采用第三种方式将额外的特征关系图信息与原有特征结合起来。

3.6　超参数设置

超参数的优化在训练机器学习模型中极为重要，会直接影响模型的最终效果。在神经网络中，超参数是用于控制学习过程的参数，而普通参数是指节点权重等网络内部的参数。因此训练模型的首要步骤就是选择合适的超参数。具体地，本文中的超参数设置如下。

1. 参数初始化。参数初始化又称权重初始化。深度学习模型训练的本质是对各个节点的权重进行更新，所以这些权重需要有相应的初始值。权重初始化方法对对模型的收敛速度和性能有着至关重要的影响。除去全零初始化这种方法外，最常用的有 Xavier 初始化[?] 和正交初始化[?] 等。本文使用 Xavier 初始化方法，保持输入和输出的方差一致（服从相同的分布），它可以帮助减少梯度消失的问题，使得信号在神经网络中可以传递得更深，在经过多层神

经元后保持在合理的范围（不至于太小或太大）。

2. 优化器。优化器也就是寻找模型最优解所用的方法。比如最基本的梯度下降法，沿着梯度的方向不断减小模型参数，从而最小化损失函数。但是其具有局部最优解的缺点，本文采用 Adam 优化器，它是一种自适应矩阵估计的优化器，可以根据梯度动态地调整学习速率。
3. 学习率。权重更新时，在梯度项之前会乘以一个系数，这个系数就是学习率。如果学习率太小，则收敛很慢会增长模型训练时间，如果学习率太大，可能会导致损失函数振荡，甚至最终发散。由于本文采用 Adam 优化器，可以自适应地调整学习率，因此设置初始学习率为 0.01。
4. 批次大小。批次大小将决定我们一次训练的样本数目，也会影响到模型的优化程度和速度。相对于正常数据集，如果 batch-size 过小，训练数据就会非常难收敛，导致欠拟合，增大 batch-size，模型训练速度会加快，但是所需的内存容量也会增加。因此选定一个合适的 batch-size，就是在内存效率和内存容量之间作出最佳平衡。本文中的 batch-size 设置为 128。
5. 正则化。正则化可以防止过拟合和提高模型泛化能力，但是会对模型产生一定约束。本文使用 L2 正则化，正则化系数设置为 0.001。

3.7　实验评估

为了验证本文提出的 FG-RNN 模型的性能，本文分别在 UNSW-NB15、CICIDS2017、CAMPUS 三个数据集上，对 FG-RNN 进行了对比评估。首先，将数据集按照 70%、15%、15% 的比例分为训练集、验证集和测试集，分别用于训练模型、调整模型超参数和测试模型。模型的损失函数为交叉熵损失函数，其公式为：

$$L = \frac{1}{N}\sum_{i} L_i = \frac{1}{N}\sum_{i} -\sum_{c=1}^{M} y_{ic}\log p_{ic} \tag{3-12}$$

3.7.1　训练过程对比

图3.9中横坐标为 FG-RNN 和 LSTM 两个算法随训练轮次的增加，准确率的变化，可以看出 LSTM 训练时间更短，大约 25 轮即可稳定，而 FG-RNN 需要训练 50 轮以上。

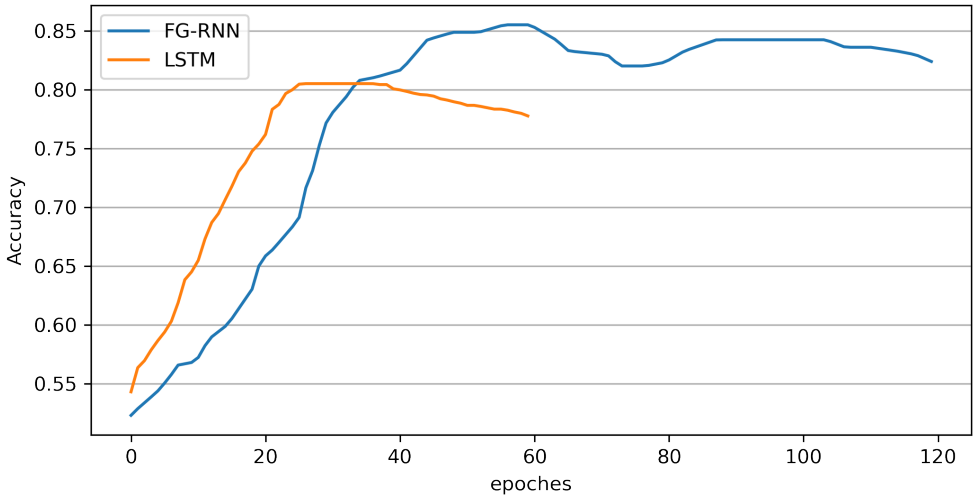


图 3.9　随着训练轮次的增加准确率的变化

图3.10表示 FG-RNN 和 LSTM 两个算法 loss 收敛的对比，在达到稳定的准确率后，loss 也会维持一个相对稳定的水平。

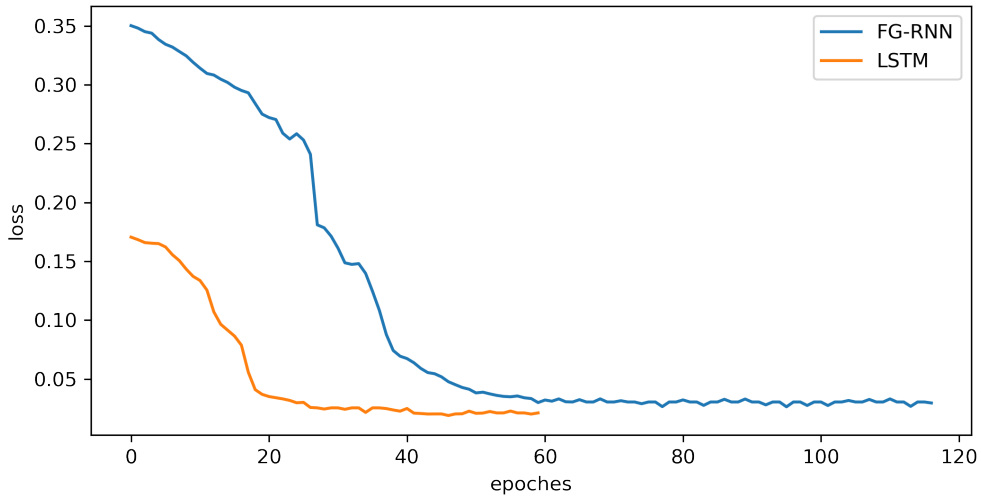


图 3.10　loss 收敛对比

3.7.2　实验结果对比

对于本文提出的模型和 baseline 模型，我们都进行了 5 次重复试验。从表3.2中可以看出，在 UNSW-NB15 和 CICIDS2017 数据集上，FG-RNN 相比于基准模型没有明显提升，这可能是因为这两个数据集场景简单，各类算法的效果已经到达瓶颈。而在 CAMPUS 数据集上，FG-RNN 明显好于 LR、DT、RF 这些基准模型，并且在基本的 RNN 类模型上提升巨大。该表格中使用的指标为准确率，为了全方位

对比实验结果，我们从精确率、召回率、F1-score、运行时长等多方面进行了对比。结果如图 4.11-4.14 所示。本实验中除 FG-RNN 进行了超参数优化外，其余模型均使用默认参数，因此其他模型的效果可能仍有提升空间。

1. 各类算法在开源数据集上的效果均要好于在 CAMPUS 数据集上的效果，这是因为 CAMPUS 数据集远比其余数据集更复杂。
2. FG-RNN 在精确率指标上比机器学习模型提升了约 10.8%，比深度学习模型提升了超过 50%，说明 FG 能够为 RNN 有效增加特征信息。
3. 从召回率来看，相比于 baseline 模型，FG-RNN 没有很好地提高召回率，说明该模型会发生很多“漏报”情况，后续我们更详细地分析了不同类别异常的召回率。
4. F1-score 兼顾了精确率和召回率的评估。由于召回率的影响，FG-RNN 仅比机器学习模型提升了 2% 的 F1-score。后续工作可以考虑从提高召回率的角度优化模型。
5. 由于数据集大小不同，对比同一算法在不同数据集上的运行时间没有意义。在相同数据集上，可以看出 RNN 类模型的训练时间要普遍比机器学习模型耗时更长，而 FG-RNN 又要消耗约两倍的时间，但是相比于准召和 F1-score 带来的提升，这些训练开销是可以接受的。

表 3.2　不同数据集下实验评估结果 (%)

数据集	任务	LR	DT	RF	RNN	LSTM	GRU	FG-RNN
UNSW-NB15	二分类	96.8	97.33	98.52	95.51	96.74	94.91	95.8
	多分类	94.73	97.96	96.82	93.98	92.11	94.30	94.24
CICIDS2017	二分类	98.26	97.11	95.54	90.18	88.04	91.49	92.45
	多分类	97.01	97.37	94.41	88.31	91.16	90.18	90.67
CAMPUS	二分类	76.98	77.95	77.51	55.62	59.80	55.25	84.34
	多分类	73.33	74.01	74.54	53.01	56.59	59.28	82.74

接下来，我们针对 FG-RNN 模型在 CAMPUS 数据集中的实验结果，具体分析了每种异常类型的准确率和召回率。将准确率定义为：

$$P = \frac{\text{模型检测出的真正的异常量}}{\text{模型检测出的异常量}} \tag{3-13}$$

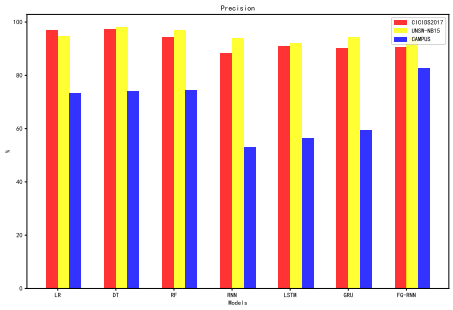


图 3.11　实验结果对比-精确率

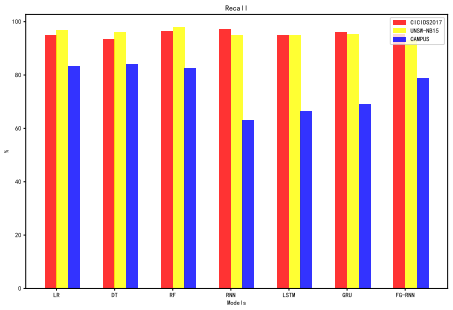


图 3.12　实验结果对比-召回率

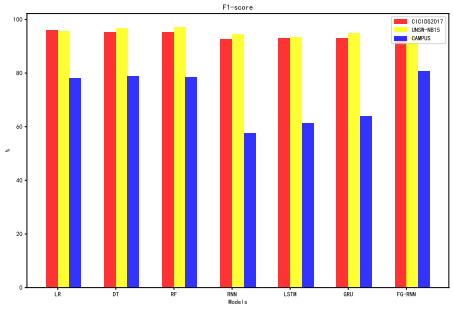


图 3.13　实验结果对比-F1 score

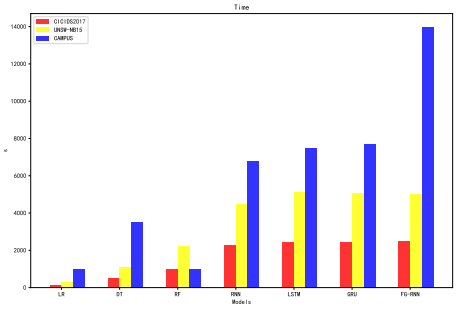


图 3.14　实验结果对比-时间

召回率定义为:

$$R = \frac{\text{模型检测出的真正异常量}}{\text{数据集中的全部异常量}} \tag{3-14}$$

如图3.15所示，模型的准确率整体而言尚可，最低为远控木马类的 76.3%，最高为拒绝服务类的 90.5%，平均为 82.7%，这体现了模型检测结果误报率低的特点，检测结果具有较强的可信度。但是模型对于不同类型异常的召回率相差很大，远控木马仅为 12%，而暴力猜解为 94.3%，平均为 78.8%，以平均值作为分界线，高于平均值的类别有僵尸网络、暴力猜解、拒绝服务、流氓推广、端口扫描，低于平均值的类别有远控木马、网络蠕虫、代码执行，这些类别具有较强的伪装性，往往和正常流量的模式行为极为相似。从召回率可以看出，模型在检测某些特定异常时效果不佳，存在较严重的漏报，由于其余类别的召回率优于 baseline 模型，所以我们可以将这些特定异常类别的检测结果仅作为参考。这部分缺陷也是我们未来工作中进行优化的方向。

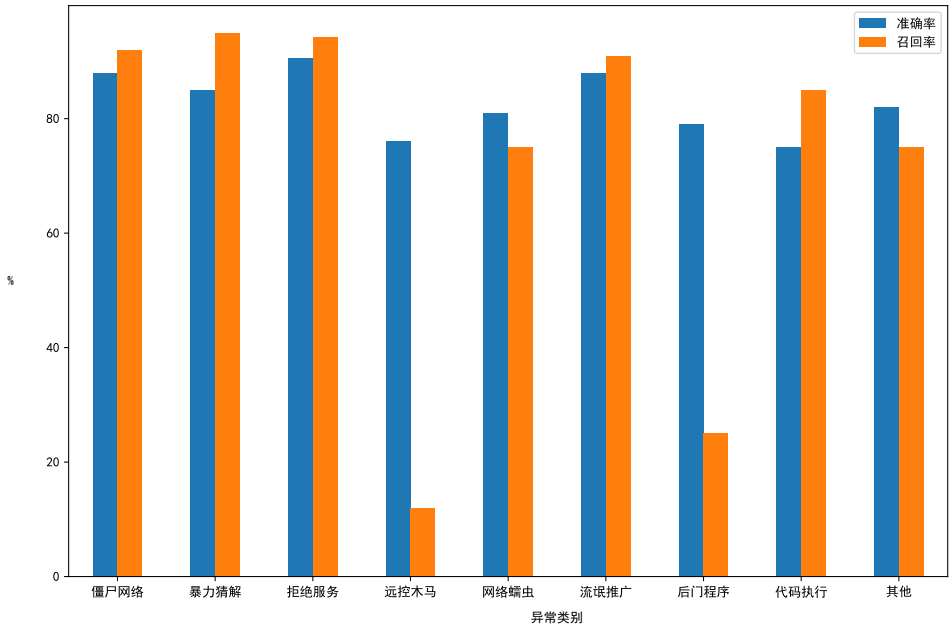


图 3.15　不同异常类型的准召对比

3.8　本章小结

为应对校园网流量环境的复杂情况，本章首先针对不同算法进行了特征分析，找出 RNN 在 CAMPUS 数据集中表现不佳的原因，然后设计了基于特征关系图的 RNN 算法，最后在多个数据集上进行了实验评估。

第 4 章　流量异常检测系统的设计与实现

4.1　引言

本章结合前四章的分析和结论，将基于特征关系图的 RNN 模型应用到实际流量环境中，验证其真实有效性。因此本文针对清华大学校园网的流量环境，设计了一个流量异常检测系统。该系统应该满足以下几点要求：

1. 实时性。如果一个系统检测延迟很高，那么即使其准确率同样很高，这个检测结果也将毫无意义。因为异常检测的目的是在攻击的早期阶段就将其发现，并且作出应对。
2. 高效性。校园网用户规模大，流量峰值高，该系统需要能够高效处理海量的流量数据。
3. 鲁棒性。该系统应该保证能够在复杂的网络环境面前依然有较好的检测效果，例如发生突出事件或者新的异常类型时依然能够有一定的检测效果。
4. 模块化。该系统的各个模块之间的耦合度应尽可能低，一方面便于调试，另一方面可以便捷的验证其他算法模型。

本文的流量异常检测系统按照上述需求设计如图4.1所示，可以分为三个模块：输入模块、检测模块、输出模块。其中输入模块将流量数据和 SOC 标签数据整合成特征数据；模型模块分为离线训练和在线检测两部分，离线训练部分定期产出更新好参数的模型，在线检测部分会接受当前的特征数据进行判别；最后输出模块将判别结果并根据历史信息进行异常等级划分，给运维人员下一步提供参考。

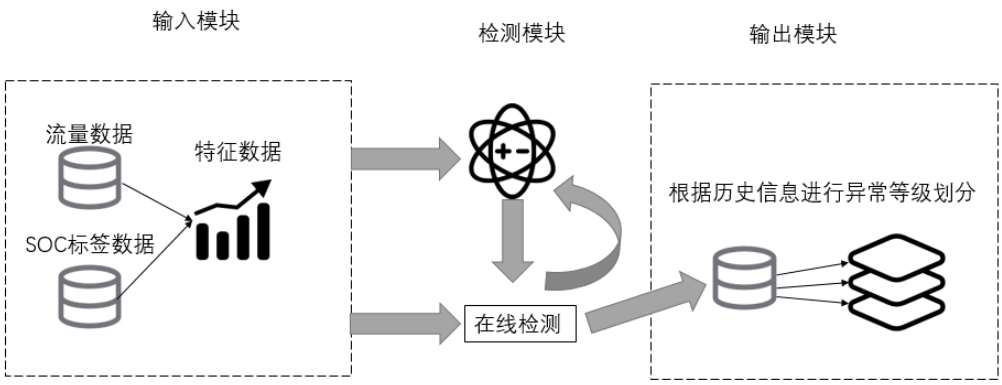


图 4.1　系统架构图

4.2　Spark 平台介绍

Spark Streaming 贯穿本节系统的始终，因此首先对 Spark 和 Spark Streaming 做一个简单介绍。

Apache Spark 是由 UC Berkeley AMP Lab 开源的用于大规模数据处理的统一分析引擎[?]，现如今已经成为 Apache 软件基金会的最为出名的开源项目之一。Spark 原理与 Hadoop 类似，都是基于 MapReduce 进行分布式计算，但是功能更加丰富，且支持 SQL 查询、流式处理、图计算等功能。

Spark 具有以下几个特点：

- 速度快。这是因为 Spark 基于内存计算，而 Hadoop MapReduce 必须进行读取和写入磁盘，IO 操作比内存操作更加耗时。图4.2是 Spark 官网给出的一项逻辑回归任务在 Hadoop 和 Spark 两个系统的运行时长对比。

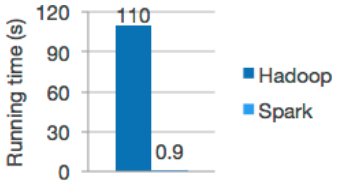


图 4.2　Logistic regression in Hadoop and Spark

- 易用性。Spark 提供了 80 余种高级 API，可以轻松编写高度并行的应用程序。Spark 支持 Java, Scala, Python, R, and SQL 等多种编程语言。
- 兼容性。Spark 支持多种数据源作为输入，例如本地文件系统、HDFS、HBase 等，并且 Spark 支持多种运行模式，能够做到类似 java 的“一次编写，多处运行”。

如图4.3所示[?]，Spark 项目包含多个独立组件。其最核心的模块为深蓝色的 Spark Core，其定义了核心 API，如“弹性分布式数据集”，该模块也负责调度、监控计算集群中的计算任务，具有内存管理、故障恢复、与管理系统和存储系统交互等功能。为了满足分布式计算系统的可扩展性，Spark 支持在各种集群管理器之上运行，例如图中灰色的模块 Hadoop YARN，Apache Mesos 以及浅蓝色的 Spark 系统中包含的“独立调度程序”（Standalone Scheduler）。在 Spark Core 之上，具有浅蓝色的四个组件：Spark SQL（处理结构化数据），Spark Streaming（实时处理海量数据），MLlib（运行机器学习模型），GraphX（提供图计算）。本文中的系统主要使用 Spark Streaming 组件。

图4.4是 Spark Streaming 的基本工作原理[?]。Spark Streaming 本质上是微批处理系统，其将输入数据流划分成批式数据，然后依次处理每批数据。

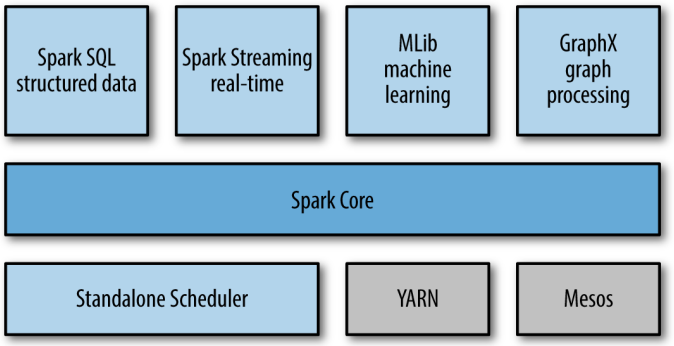


图 4.3　Spark 组件

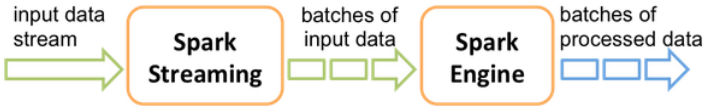


图 4.4　Spark Streaming 工作原理

4.3　系统各模块设计

本文实现流量异常检测系统的架构如图4.1所示，整个架构是基于 Spark Streaming 实现。本节将分模块具体介绍。

4.3.1　输入模块

输入模块的主要功能是对流式数据进行特征抽取，与第三章中的特征提取单一文件不同的是，本模块处理的数据是“无界”的。我们需要将无界数据进行切片，每次处理一个片段内的数据。这些片段又被称为“window”，window 是流式计算领域的核心概念。

本模块的输入为由 kafka 传入的流量数据，然后将流量报文划分到一个个滑动窗口（sliding window）中，每次处理一个窗口内的全部报文。流量特征提取的整体流程图4.5所示。读取到 packet 后，首先为每条报文建立一个由五元组信息组成的标志，然后根据图中的规则依次判断该报文是否是一条新的流的起始报文、是否是当前流的结束报文、是否需要调用 updateflow 更新各项统计信息。

本模块提取的结果是以一条 TCP 流或者 UDP 流为单位，该条流中全部报文的统计信息。在一条流中有很多个数据包，TCP 流以三次握手为开始，以 FIN 标志为结束；UDP 流以首次出现为开始，以超出超时时间为结束。并且每条流都是双向数据流，由源 ip 地址到目的 ip 地址为正向，目的 ip 地址到源 ip 地址为反向。

使用该方法得到的 csv 文件如图4.6所示：

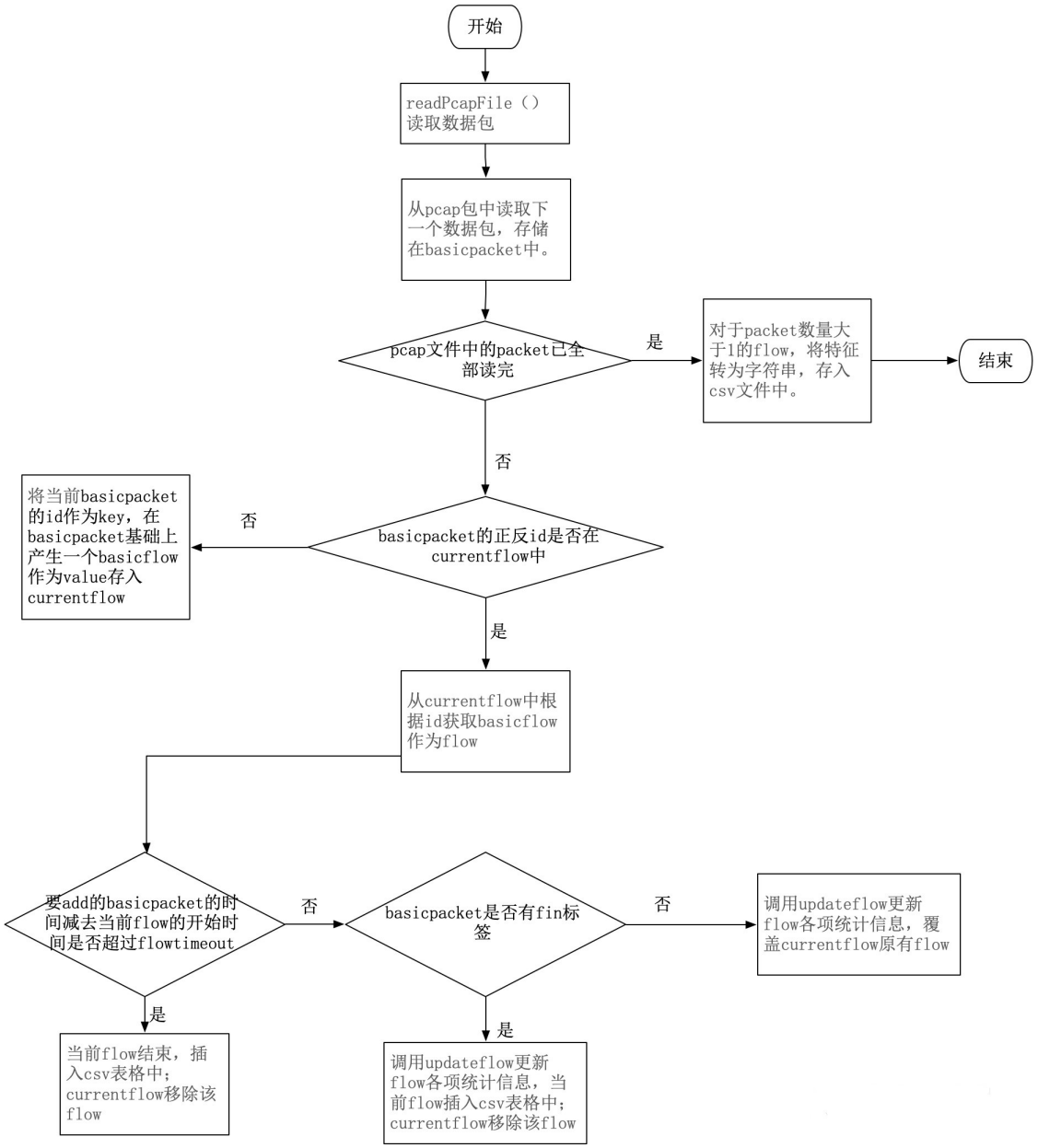


图 4.5　特征提取流程图

4.3.2　检测模块

检测模块分为离线和在线两部分，并且由于第四章中 FG-RNN 算法的特点，特征关系图（Feature Graph）的更新频率与循环神经网络（RNN）不同，并且这两者在实现中已经解耦，因此 RNN 的参数更新仅需在离线部分完成，在线部分可以兼顾实时流量的检测和特征关系图参数的更新。该模块的架构图如图4.7所示。

该模块核心伪代码如下：

```
% 从kafka中读取DataFrame数据，转换成Spark Streaming的RDD
kafkaStream = KafkaUtils.createDirectStream
```

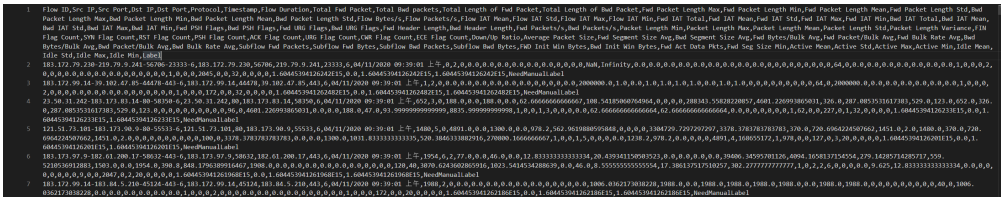


图 4.6　特征文件

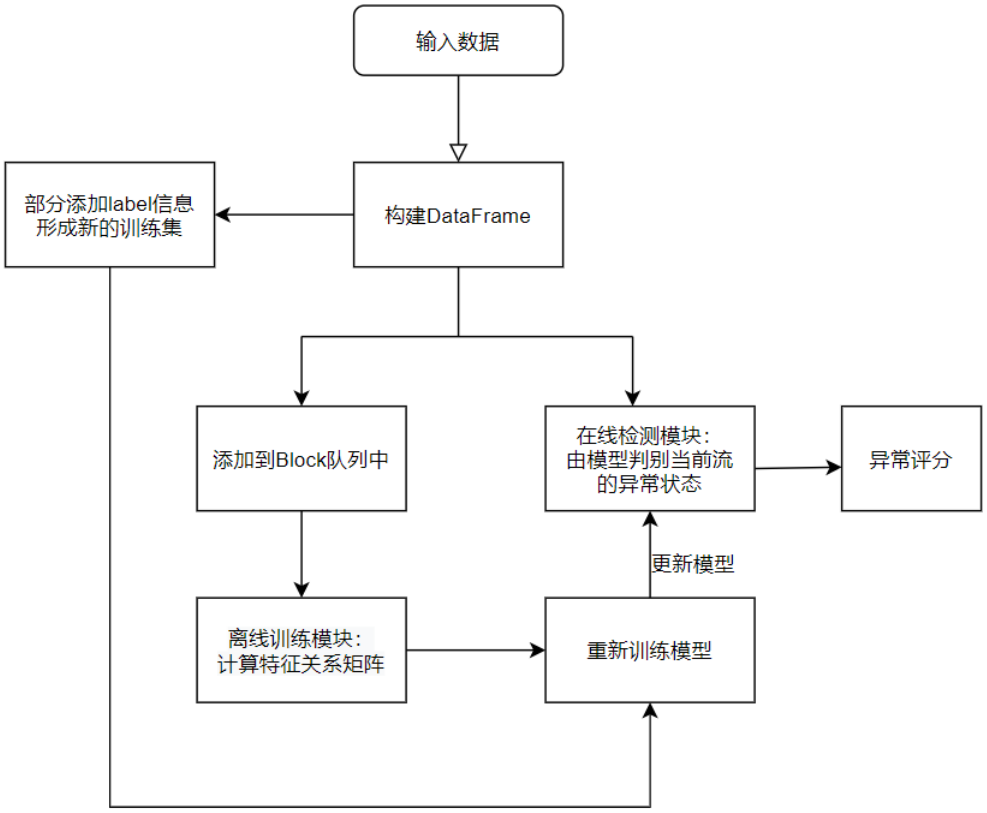


图 4.7　检测模块架构图

```
        (ssc, dataframe, {"metadata.broker.list": brokers})

% 离线训练，将RDD添加到block队列，计算特征关系矩阵
kafkaStream.foreach(caculate_feature_graph)
model = FGRNN(fg)
model.save(sc, "path/models")

% 在线检测，判别当前流的异常状态
kafkaStream.foreach(model.predict)
```

该模块部分运行过程如图4.8所示，

```
2021-03-23 09:32:32,631 - INFO - Log directory: data/model/pretrained/CAMPUS/
2021-03-23 09:32:32,632 - INFO - {'base_dir': 'data/model', 'data': {'batch_size': 128, 'dataset_dir': 'data/CAMPUS', 'graph_pkl_filename': 'data/adj_mx.pkl',
'test_batch_size': 64, 'val_batch_size': 64}, 'log_level': 'INFO', 'model': {'cl_decay_steps': 2000, 'filter_type': 'dual_random_walk', 'horizon': 12, 'input_d
im': 2, 'll_decay': 0, 'max_diffusion_step': 2, 'num_nodes': 325, 'num_rnn_layers': 2, 'output_dim': 1, 'rnn_units': 64, 'seq_len': 12, 'use_curriculum_learnin
g': True}, 'train': {'base_lr': 0.01, 'dropout': 0, 'epoch': 53, 'epochs': 100, 'epsilon': 0.001, 'global_step': 30780, 'log_dir': 'data/model/pretrained/CAMPU
S/', 'lr_decay_ratio': 0.1, 'max_grad_norm': 5, 'max_to_keep': 100, 'min_learning_rate': 2e-06, 'model_filename': 'data/model/pretrained/CAMPUS/models-1.6139-3
0780', 'optimizer': 'adam', 'patience': 50, 'steps': [20, 30, 40, 50], 'test_every_n_epochs': 10}}
2021-03-23 09:33:03,027 - INFO - ('x_train', (36465, 12, 325, 2))
2021-03-23 09:33:03,028 - INFO - ('y_train', (36465, 12, 325, 2))
2021-03-23 09:33:03,028 - INFO - ('x_val', (5209, 12, 325, 2))
2021-03-23 09:33:03,028 - INFO - ('y_val', (5209, 12, 325, 2))
2021-03-23 09:33:03,028 - INFO - ('x_test', (10419, 12, 325, 2))
2021-03-23 09:33:03,028 - INFO - ('y_test', (10419, 12, 325, 2))
WARNING:tensorflow:At least two cells provided to MultiRNNCell are the same object and will share weights.
2021-03-23 09:33:22,176 - INFO - Total number of trainable parameters: 372352
2021-03-23 09:34:38,780 - INFO - Horizon 01, MAE: 0.85, MAPE: 0.0163, RMSE: 1.53
2021-03-23 09:34:39,003 - INFO - Horizon 02, MAE: 1.12, MAPE: 0.0225, RMSE: 2.22
2021-03-23 09:34:39,226 - INFO - Horizon 03, MAE: 1.31, MAPE: 0.0273, RMSE: 2.75
2021-03-23 09:34:39,451 - INFO - Horizon 04, MAE: 1.45, MAPE: 0.0312, RMSE: 3.17
2021-03-23 09:34:39,678 - INFO - Horizon 05, MAE: 1.56, MAPE: 0.0343, RMSE: 3.49
2021-03-23 09:34:39,936 - INFO - Horizon 06, MAE: 1.64, MAPE: 0.0369, RMSE: 3.74
2021-03-23 09:34:40,163 - INFO - Horizon 07, MAE: 1.72, MAPE: 0.0392, RMSE: 3.94
2021-03-23 09:34:40,389 - INFO - Horizon 08, MAE: 1.78, MAPE: 0.0411, RMSE: 4.11
2021-03-23 09:34:40,620 - INFO - Horizon 09, MAE: 1.83, MAPE: 0.0426, RMSE: 4.25
2021-03-23 09:34:40,856 - INFO - Horizon 10, MAE: 1.88, MAPE: 0.0440, RMSE: 4.37
2021-03-23 09:34:41,112 - INFO - Horizon 11, MAE: 1.92, MAPE: 0.0451, RMSE: 4.47
2021-03-23 09:34:41,378 - INFO - Horizon 12, MAE: 1.96, MAPE: 0.0462, RMSE: 4.55
```

图 4.8　检测模块运行过程

该模块实测运行过程中，我们发现模型检测效果会随时间剧烈衰减，如图4.9所示，经过 5 小时后 FG-RNN 模型效果衰减到普通 RNN 模型的程度。本文中的模型本质上是一个时间序列模型，根据过去一段时间的数据的行为模式总结出历史规律然后对未来进行预测，将预测结果与真实数据对比达到检测异常的效果。因此模型准确率较高的一个前提是要预测的未来是训练数据集中历史规律的延续，测试集和训练集越相似，准确率就越高。但是在实际现网环境中，数据中的异常种类和数量在不断变化，时间间隔越大，数据集的行为模式差异越大。

而本文用来训练模型的数据集是一个静态数据集，只能描述过去一段时间的规律模式，随着时间变化，训练数据中的规律模式会逐渐不再准确，这必然导致模型在使用一段时间后会出现检测能力下降的现象，模型结果不可靠。

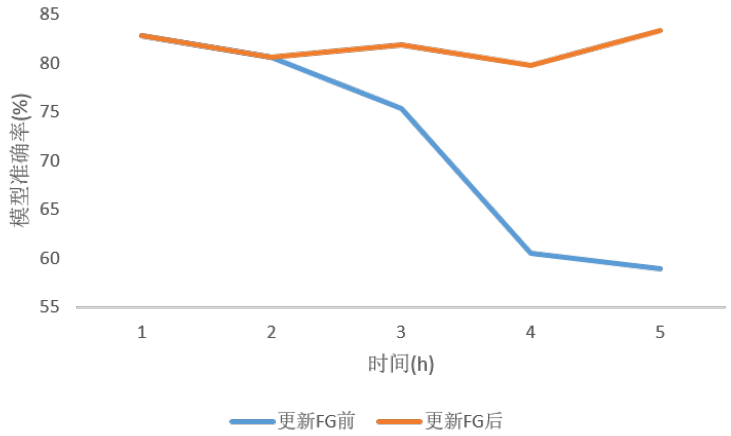


图 4.9　模型效果随时间衰减

为了减缓模型检测效果衰减的速度，本文在系统中引入了动态更新特征图的机制，由于模型中的特征图代表了训练集的行为模式（pattern），本机制无需重新训练模型参数，仅通过更新特征图就可以满足需求。为了确定更新频率，我们计

算了一段时间内的模型衰减程度，因为如果更新过快，则需要更多的算力，更新过慢，则检测效果不佳。如表4.1所示，其中每段数据集时间间隔为半小时，最终我们设置 2 小时更新一次特征图。

表 4.1　模型衰减程度

数据集序号	1	2	3	4	5	6	7	8
模型准确率	0.83	0.81	0.79	0.78	0.74	0.67	0.63	0.61
模型衰减程度	-	0.02	0.04	0.05	0.09	0.16	0.20	0.22

4.3.3　输出模块

在得到实时检测模块给出的异常流信息以及类型后，我们可以基于此做出进一步的处理。由于校园网流量“异常是常态，甚至半数流量是异常”的特点，若将全部异常信息都汇报给运维人员，大量危害程度低的异常告警势必会掩盖真正值得关注的异常，例如时刻都在发生的网络扫描，由于其重要性低微我们可以将其忽略。

因此，我们需要对异常的重要等级进行评分，具体的评分规则如下：首先初始化每种异常类型的基础分数，将异常分为低危、中危、高危三类，然后每次运维人员处理一条异常流后，将发现该流异常到处理异常的时间差作为权重，重新计算该异常的评分。

4.4　系统实现

本节对上述设计的异常检测系统进行了测试和验证。本文使用了三台主机搭建的集群运行该系统，分别命名为 master、slave1、slave2，每台主机安装 jdk-1.8.2，hadoop-2.7，spark-3.0，zookeeper-3.4，kafka-2.13 等软件，用于搭建该系统所需要的环境，具体的配置参数如表4.2所示。

在系统环境搭建好后，启动并测试系统中各个模块的流程如下：

1. 在三台主机上，执行 start-dfs.sh 启动 hdfs 系统，执行 start-yarn 启动 yarn 资源管理器，执行 zkServer.sh 启动 zookeeper，执行 kafka-server-start.sh ./config/server.properties & 后台启动 kafka。
2. 使用脚本 spark-submit –class streaming.FGRNN \
   –master yarn –deploy-mode cluster \
   –driver-memory 8g –executor-memory 8g \
   –executor-core 1 –num-executors 2 \

表 4.2　集群各节点的配置情况

主机名	内存	CPU	进程
master	16GB	Intel(R) Core(TM) i7-8700	NameNode, ResourceManager, QuorumPeerMain, Kafka, NodeManager
slave1	8GB	Intel(R) Core(TM) i5-7300U	NameNode, ResourceManager, QuorumPeerMain, Kafka
slave2	8GB	Intel(R) Core(TM) i5-8300	NameNode, ResourceManager, QuorumPeerMain, Kafka

–name feature_extract_job feature_extract.py 以及–name off_fgrnn.py，–name online_fgrnn.py 分别向 spark 集群提交特征提取、离线训练模型和在线检测模型三个任务。

3. 运行 hadoop fs -put *.pcap /input 将 pcap 文件拷贝到 hdfs 系统。
4. 访问 error.html，total.html 等页面查看任务执行情况，访问 hdfs 系统中的输出查看检测结果。

我们可以在 Spark Web UI 中查看系统的运行过程，Web UI 展示了全部任务的执行状态，其中 Jobs 展示的是整个 spark 应用任务的 job 整体信息，当程序遇到一个 RDD 算子时，就会提交一个 job。一个 job 通常包含一个或多个 stage，各个 stage 之间按照顺序执行。Task 是比 stage 更加细分的执行单元，task 的数量就是 stage 的并行度。从图4.10可以看出，对于每个窗口内的数据，系统在执行特征提取模块时尽管可以并行执行，仍然需要耗时约 6-11min，而在完成模型训练后，在线检测仅需要 100ms 即可完成，本系统的运行瓶颈主要在特征提取模块。

4.5　系统运行结果

我们在本系统中持续导入了约 200 小时的流量数据以验证该系统的有效性。系统共对 15,998,551,646 个 packet 进行分析，根据时间戳依次读取每个报文信息，提取特征，最终以 5 分钟为粒度汇报异常信息，累计汇报异常 packet 数量为 3521687 个，与此同时，SOC 平台汇报异常 packet 总数为 5496239 个，总占比为 64.07%，具体的每类异常数量如图4.11所示，其中远控木马、网络蠕虫、流氓推广、代码执

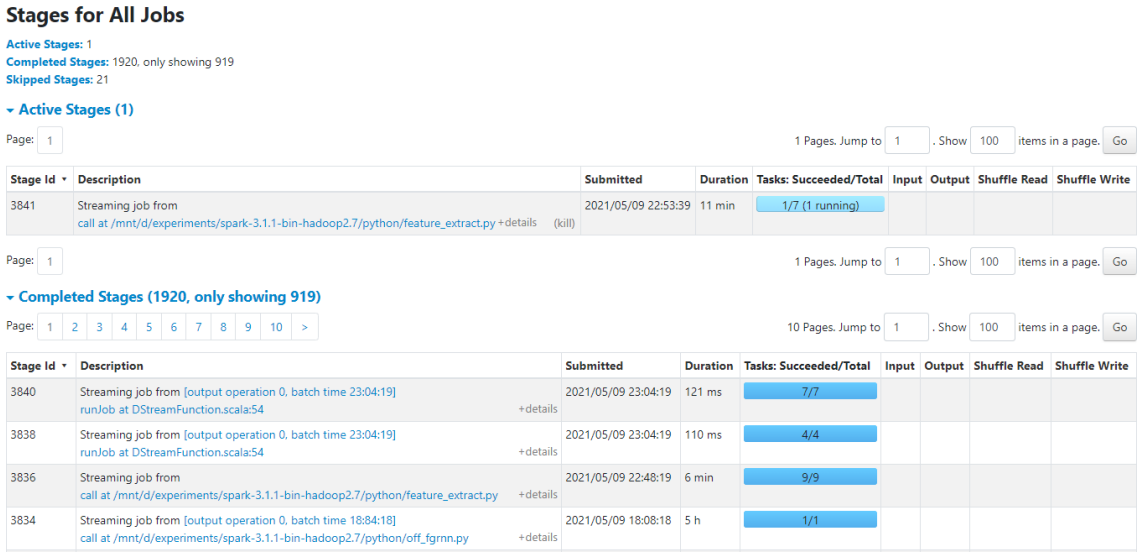


图 4.10　Spark Web UI

行等类别的检出率偏低。我们将汇报结果与 SOC 平台中的告警信息进行对比，绘制了检测到的异常数量随时间变化图4.12和异常准召随时间变化图4.13，图中横坐标为时间轴，从 2021-03-19 13:30 开始，到 2021-3-20 19:30 为止，以 5 分钟为一个刻度，从图中可以看出：

1. 异常数量分布不均匀，异常数量会出现瞬间增多的现象，结合图 4.14 的 CDF 图可以看出，91% 的时间内异常数量在 3000 以内，1% 的时间内异常数量在 28000 以上。
2. 以 SOC 平台作为 groudtruth，本系统大部分时刻能够有效检测出 85% 以上的异常，但是部分时刻召回率极低，仅有 10% 左右，例如图中 25、127、225 三个时刻点，分别对应 2021-03-19 15:20、2021-03-20 00:05 和 2021-03-20 08:15:00。以 2021-03-19 15:20 为例进一步分析，我们发现此时 88% 的异常类型为远控木马，如图 4.15 所示，而本系统几乎未能检测出远控木马这类异常。远控木马具有持续性长，欺骗性强等特点，从流量特征来看，其与正常流量十分相似，可能会有长期的定时心跳包，仅凭这一点特征本系统难以检测此类型的异常。
3. 对于长尺度时间而言，该系统在检测部分类别的异常时漏报率偏高，但是由于这部分异常本身占比不高，并且该系统的误报率较低，因此检测结果依然有一定参考价值。

更具体地，我们选取了三个时刻点的数据进行进一步分析，如图 4.15-4.17 所示，当召回率极低时，往往是由于远控木马等难以检测的异常类别占比较高；而异常数量突增时，如果其组成成分多为拒绝服务、端口扫描等，系统的检测结果甚至会高于平均值。并且大部分时刻系统表现良好，能够有效检测出主要异常，且

准确率较高。

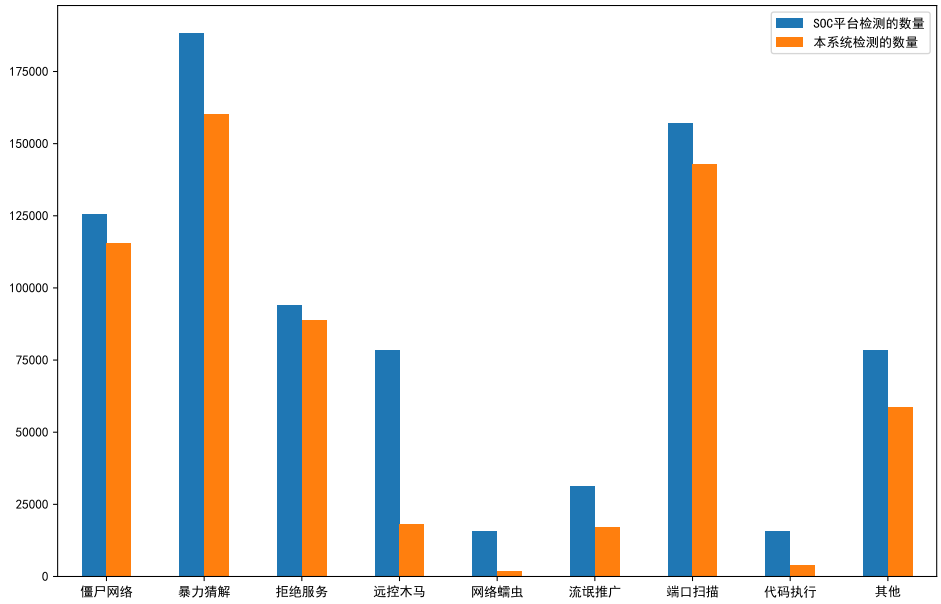


图 4.11　每类异常数量对比

4.6　本章小结

这一章设计并实现了一个流量异常检测系统，首先分别分模块对其进行了介绍，然后介绍了搭建系统的参数和流程，并且对系统运行结果进行了分析。

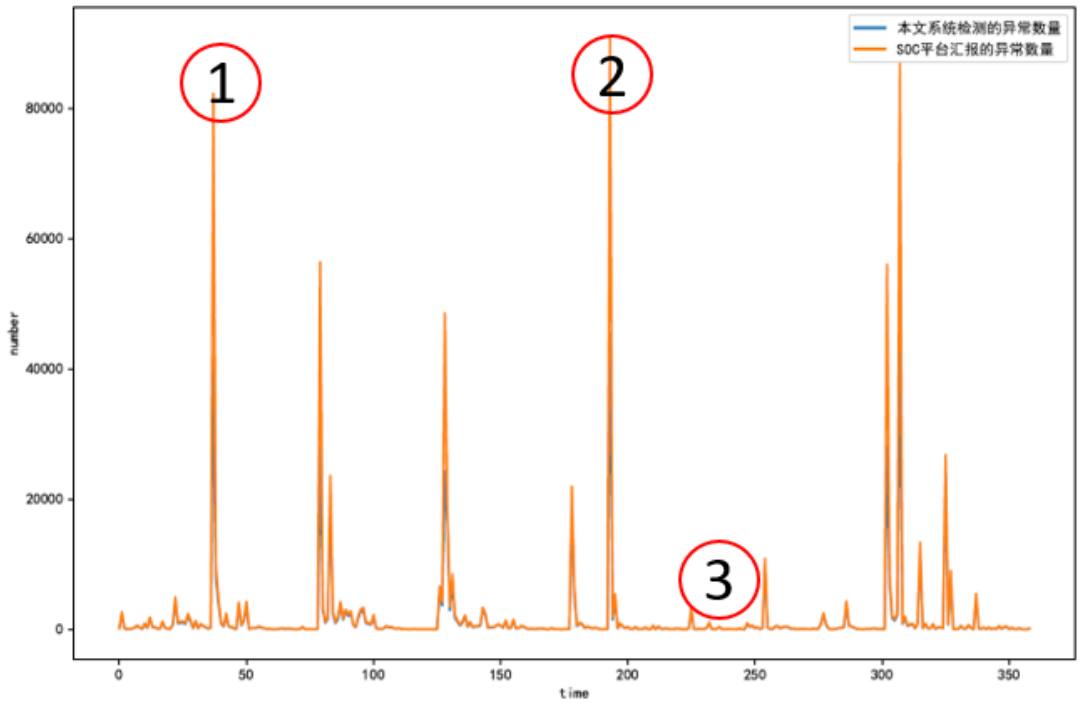


图 4.12　异常数量随时间变化图

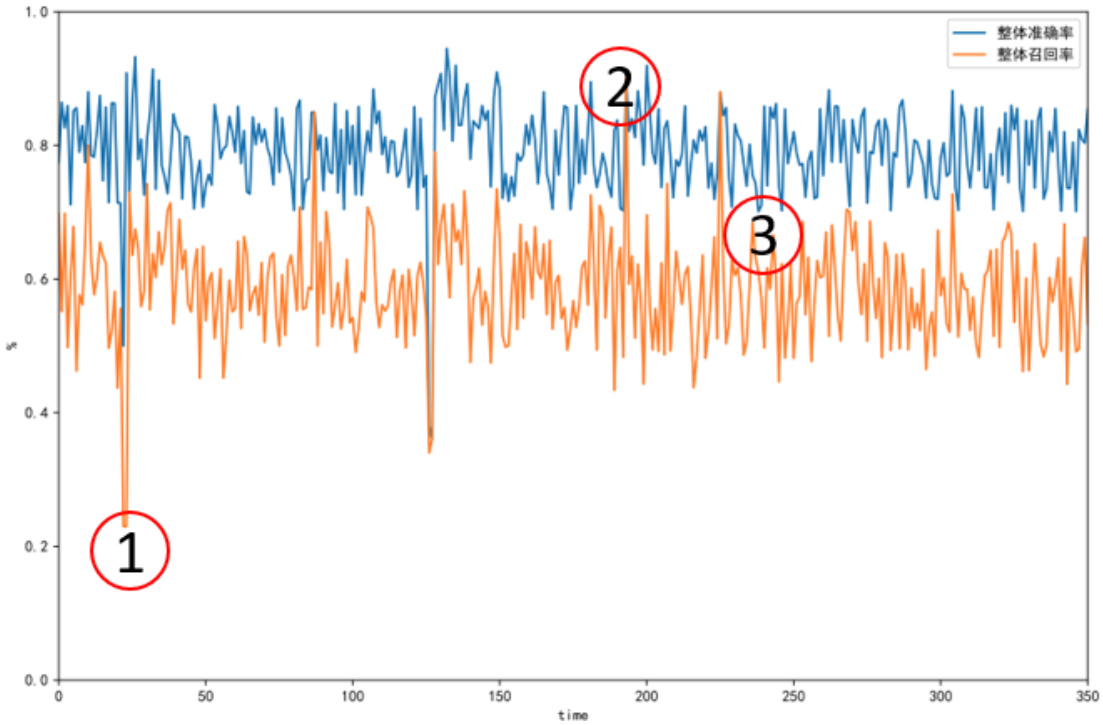


图 4.13　异常准召随时间变化图

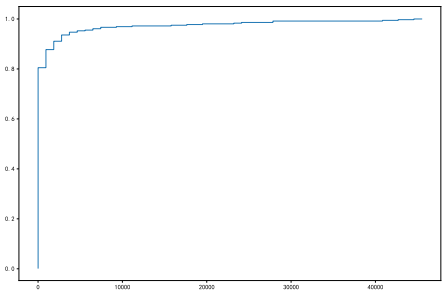


图 4.14　异常数量 CDF 图

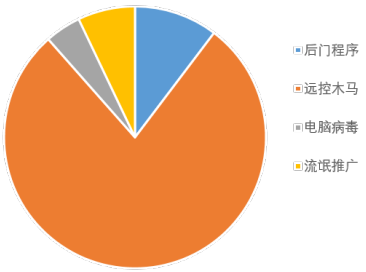


图 4.15　时刻 1 的异常分布

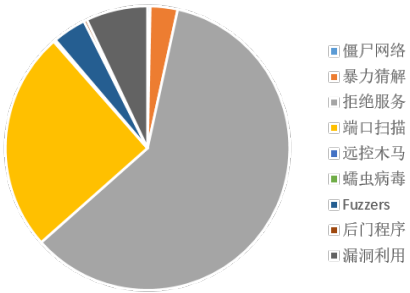


图 4.16　时刻 2 的异常分布

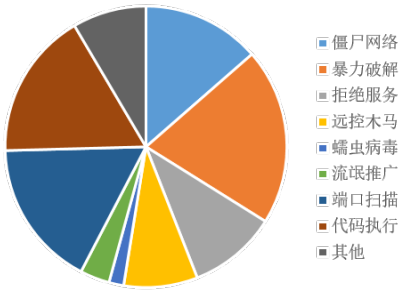


图 4.17　时刻 3 的异常分布

第 5 章　异常检测系统在检测端口扫描中的应用

在应用上述异常检测系统进行检测时，我们发现根据特征分析以及检测结果，该系统能够更有效地检测出 DDoS、PortScan 等异常。

本文发现在校园网 vlan-pool 架构下，出现大量 PortScan 时，由于外部主机试图访问内网不存在的主机，会在子网内部引发 ARP 报文，导致时延增高。PortScan 本身是个安全问题，但是由于 vlan-pool 对 ARP 报文的放大效应，引发了一定的性能问题。发现该问题后，为了定量分析 PortScan 对无线网性能的具体影响，本文首先搭建了一个小型子网进行了模拟仿真的受控实验，然后在校园网范围内进行了被动测量。验证端口扫描会对性能产生影响后，我们采用了封禁 ip 和增加缓存两种策略，有效降低了系统时延。

5.1　受控实验

为了验证端口扫描引发的大量 ARP 对无线网性能的定量影响，本文设计了一组受控实验，该实验的架构图如5.1所示，包含 1 个 AP 和 10 台终端设备，每台设备有着不同的物理速率。其中 4 台终端通过有线网连接到 AP，用于控制 ARP 报文的数量，6 台终端通过无线网连接到 AP，用于控制局域网的整体负载。本文通过 iperf 控制 AP 的负载，nmap 控制扫描流量的数量。值得注意的是，该局域网是一个 IEEE 802.11ac 网络，将其安装在一个近似无干扰的环境中。AP 在 5G 上以 100 Mbps 的数据速率运行，功率为 80mW。各终端的物理速率如表5.1所示，物理速率是指空口物理层的传输速率，是终端与 ap 协商的值，不是固定值；该表给出了各个终端大部分时间所处的物理速率。

表 5.1　每台终端的物理速率

devName	dev1	dev2	dev3	dev4	dev5
Physical Rate	100	144	433	650	702
devName	dev6	dev7	dev8	dev9	dev10
Physical Rate	780	780	866	866	866

本文将实验分成了九组，各组实验的参数设置如表5.2所示。平均传输速率 (Mbps) 代表不同终端运行 iperf 产生的流量负载，而扫描范围代表不同终端运行 nmap 产生的扫描负载。比如 192.168.0.1/24 代表扫描 256 个 ip 地址，192.168.0.1/21

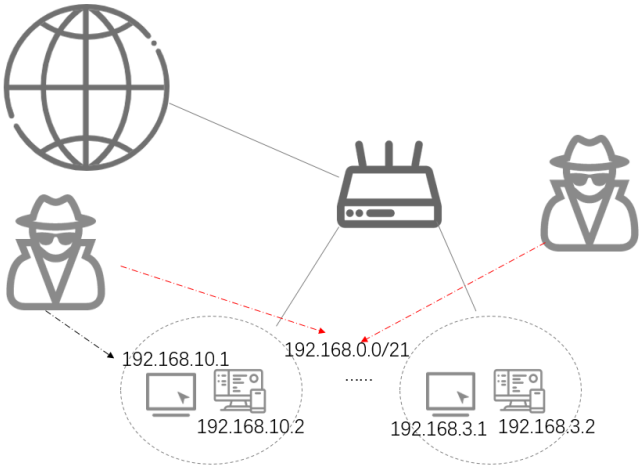


图 5.1　受控实验架构图

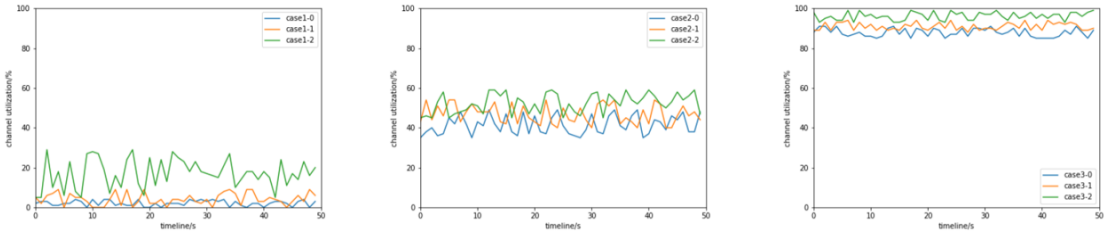
代表扫描 2048 个 ip 地址。也就是说，case1-x 代表低流量负载场景，case2-x 代表中等负载，case3-x 代表高负载。casex-1 代表无扫描环境，casex-2 代表中度扫描，casex-3 代表严重扫描。随着不同级别的流量负载和端口扫描负载，本文发现不同实验中子网具有不同的性能。

表 5.2　每组实验的参数设置

	平均传输速率 (Mbps)	扫描范围
case1-0	0	0
case1-1	0	16
case1-2	0	256
case2-0	87	0
case2-1	87	16
case2-2	87	256
case3-0	470	0
case3-1	470	16
case3-2	470	256

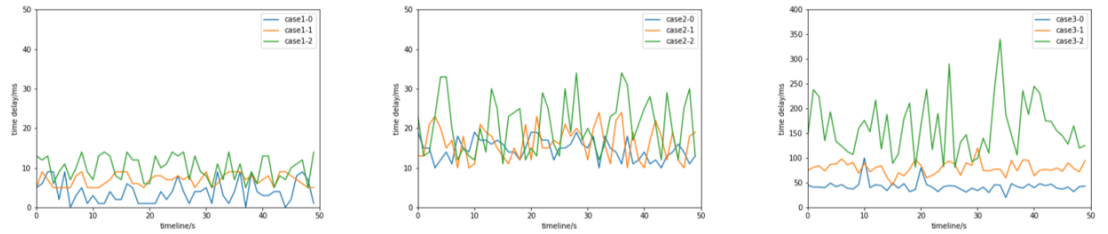
在每组实验中，我们收集了大量 arp 数量、单播报文数量、端对端延迟、丢包率、信道利用率等指标的数据。其中单播报文数量、信道利用率等指标是使用 AirMagnet[?] 进行测量得到，端对端延迟是使用 tcping 测量得到，丢包率是从 iperf 输出日志得到。不同组实验下信道利用率的比较如图 5.2所示，随着端口扫描流量的增加，信道利用率会增加 10-20%。图 5.3是不同组别的实验中时延的对比情况，在网络处于中低负载时，尽管端口扫描增加，时延变化不大，仅在 10-30ms 之间

波动。但是在高负载时，随着 arp 数量的增加，时延出现 100ms 以上的增加。这是因为 arp 数量的增加挤占了数据报文所需要的信道资源，导致数据报文丢包变多，从而时延增加。这点可以从图 5.4中看出，在高负载情况下，端口扫描增加会出现丢包率增高的现象。



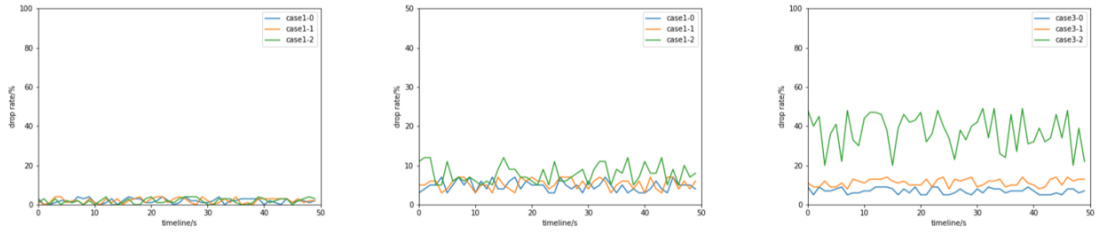
(a)　(b)　(c)

图 5.2　信道利用率对比



(a)　(b)　(c)

图 5.3　时延对比



(a)　(b)　(c)

图 5.4　丢包率对比

5.2　测量实验

在进行受控实验后，我们在清华大学校园网中进行了大规模的测量，以验证端口扫描对无线网性能的影响。在本文中，本文收集了一周内的两种数据进行测量分析和评估，广播/单播包数据用于 ARP 分析，SNMP 数据用于计算和衡量无线网的性能。广播/单播包数据由 AirMagnet 采集得到，SNMP 数据通过 SNMP 轮询

工具周期性抓取所需要的对象标识符得到。

简单网络管理协议（SNMP）是 TCP/IP 协议簇的一个应用层协议，被广泛应用于网络管理中[?]。针对一个给定的对象标识符（Object Identifier, OID），可以通过发起 SNMP 请求以键值对的形式获取对象的值，用以监控设备或者网络的状态。

在本节中，我们主要采集以下几个 OID：

- 信道利用率（Channel Utilization）：表示对于一个无线接入点来说，传输报文占用的时间比例。
- 传输的帧数（Transmitted Frames）：表示在一个数据采集周期内，无线接入点成功传输的 802.11 数据帧的数量。该 OID 定义为 $Tran_{num}$。
- 重传的帧数（Retransmitted Frames）：表示在一个数据采集周期内，由于无线链路层的丢包或者包错误（packet error）导致链路层重传的总帧数。该 OID 定义为 $Retry_{num}$。
- 失败的帧数（Failed Frames）：表示在一个数据采集周期内，在经历了一定次数的链路层重传尝试后最终传输失败的总帧数。该 OID 定义为 $Failed_{num}$。
- 时延（Latency）：终端侧的无线用户发送报文的平均延迟时间。该值为一段时间内的平均值。

利用上述部分 OID 我们可以计算出 MAC 层丢包率，即当无线 MAC 层发生丢包时，会重传相应的数据包，直到成功传输或重传次数达到预设阈值。其定义如下：

$$loss_{mac} = \frac{Retry_{num} + Fail_{num}}{Trans_{num} + Retry_{num} + Fail_{num}} \tag{5-1}$$

我们将采集到的数据按照 arp 数量进行对比，发现当网络中出现大规模端口扫描时，丢包率和时延都有增高。图 5.5和图 5.6分别展示了某个 AP 下 arp 数量和时延、丢包率的关系。其中两图的横坐标为时间轴，每个坐标点代表 5 分钟，其中以 78 刻度作为分界线，左右两个区间内各自连续，纵坐标为双坐标轴，分别表示时延、丢包率、arp 数量，均为 5 分钟内的平均值。从图中可以看出，arp 报文数量增加到四倍时，丢包率从原来的 3% 增加到 20%，而时延则提高到 100ms 以上。

5.3　优化策略

在以上基础上，我们采用了两种优化策略。

1. 在出口网关处进行端口扫描 ip 的封禁，由第四章的异常检测系统提供端口扫描 ip 的列表，然后运维人员对这些 ip 地址进行封禁操作。
2. 在 AC 中添加当前子网已分配 ip 地址的缓存，该缓存为关联在当前 AP 上的设备 mac 与对应 ip 地址的映射表。判断流程如图5.7所示，每当 AP 收到一个

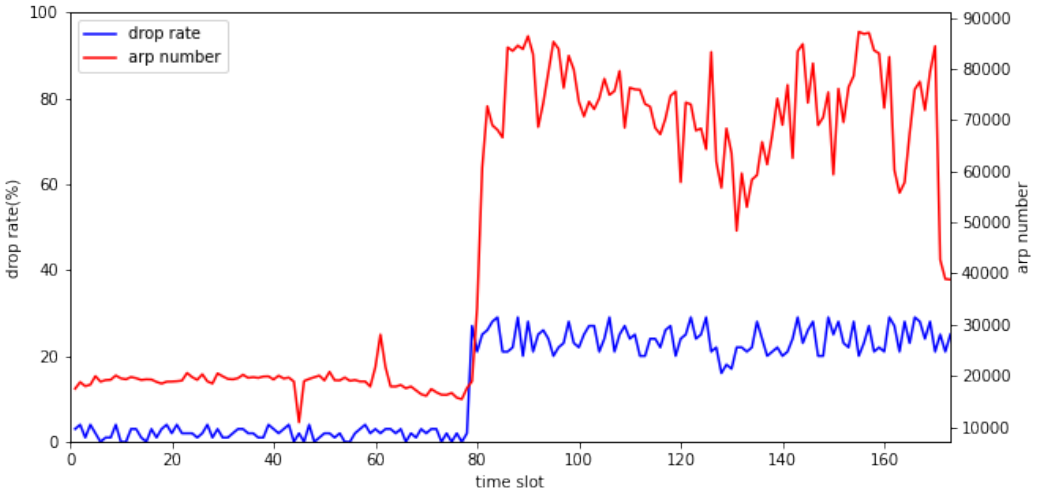


图 5.5　arp 数量和丢包率的关系

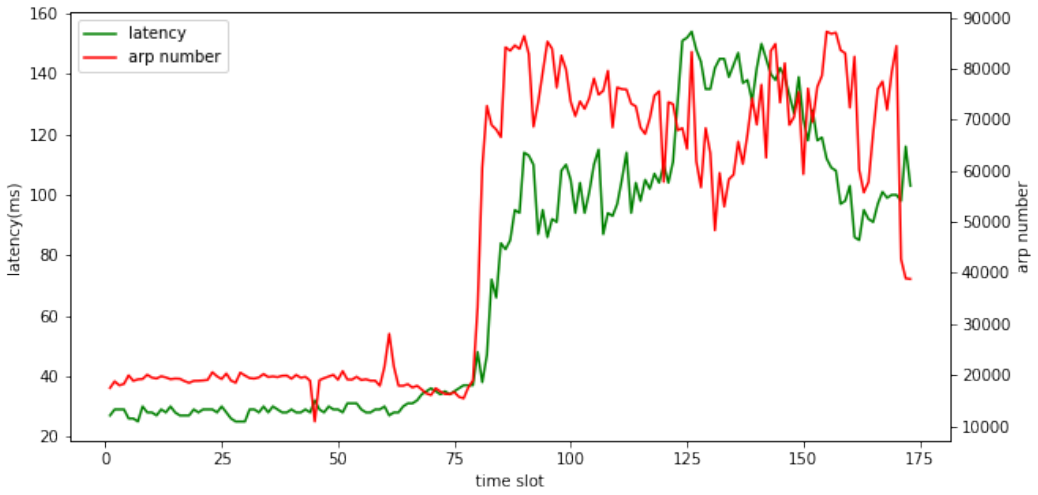


图 5.6　arp 数量和时延的关系

报文，首先查看 ip 地址是否存在于映射表中，如果在则说明该 ip 已分配给某台设备，直接转发该报文，不会产生 arp；如果 ip 地址不在映射表中，则需要判断表中是否存在未分配 ip 地址的设备 mac，如果存在，需要下发相应的 arp 报文，如果不存在，则直接将报文丢弃。因为当出现漫游设备时，映射表中该设备 mac 可能没有对应的 ip 地址，即此时 n 个设备关联在 AP 上，只有 m（$m < n$）个 ip 地址已分配。

经过仿真验证，这两种策略均能有效降低 arp 数量，并且在外部扫描依然存在的情况下，子网平均时延由 150ms 下降到 50ms 左右。

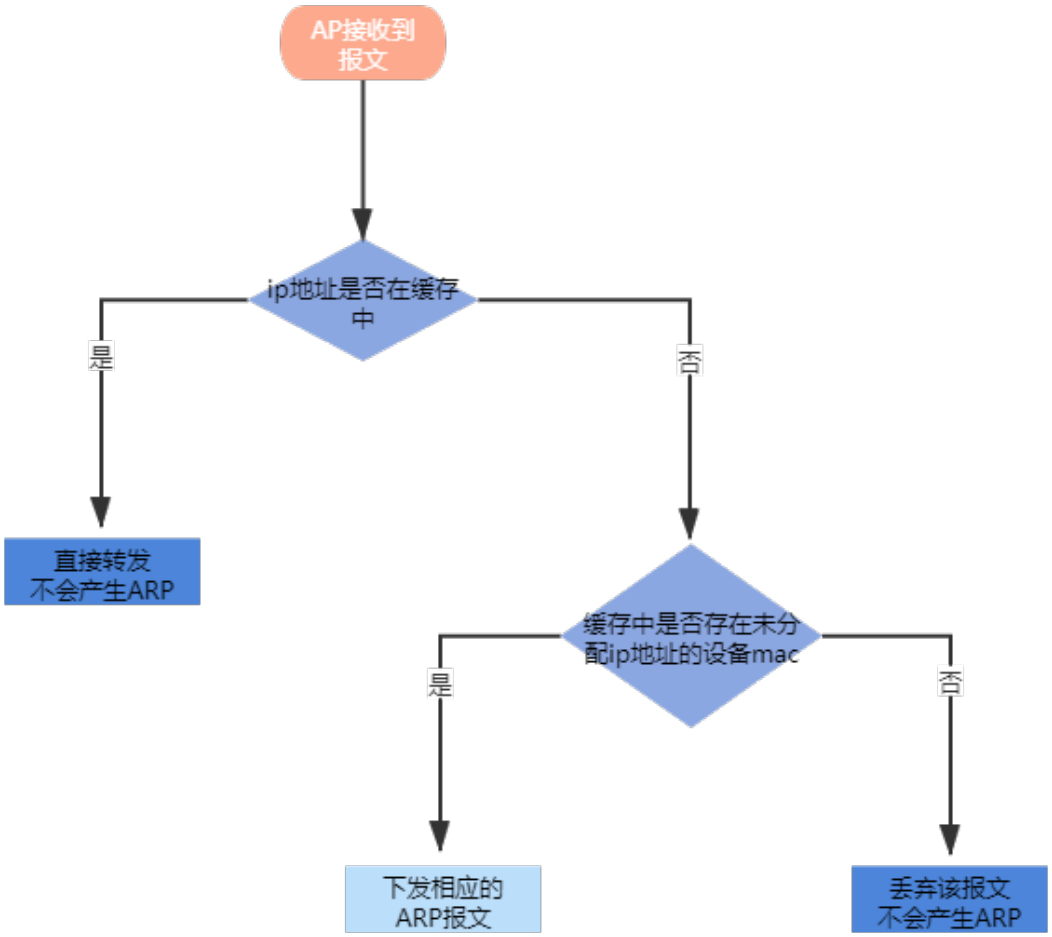


图 5.7　增加缓存的优化策略

5.4　本章小结

通常端口扫描被视为一个安全问题，在前述章节中这类异常可以有效地被检测系统检测到，但是本章发现端口扫描会对无线网性能产生一定影响。当发生对子网中不存在的 ip 进行扫描时，由于 vlan-pool 架构中一个 AP 可以工作在多个 vlan 下，该 AP 会在所有 vlan 中发送该 ip 地址相关的 ARP 报文，产生 ARP 泛洪。本章通过受控实验和测量实验，验证了该现象，并且使用封禁 ip 和增加缓存策略，有效缓解了该问题。

第 6 章　总结与展望

6.1　总结

随着互联网的规模越来越大，网络流量的规模也急剧膨胀，网络应用的种类日益多样化，因此管理网络的难度也越来越大。传统的基于专家知识、人工生成规则的自动化运维具有很多缺陷，一是无法适应快速变化的网络流量环境，过度依赖规则库，而规则库需要不断人为添加新的规则；二是面对复杂网络情况时，很难根据流量的实时变化动态更新基线。

异常检测作为智能运维的关键环节，现有的异常检测算法具有很多缺陷，如面对海量数据规模时，无法或很难做到实时性；应用类型多导致的流量特征复杂，因此很难达到较高的检测率和较低的误报率；大多数异常检测算法仅能报告是否发生异常，难以对确定异常种类和定位异常来源给出指导性意见。

本文以清华大学校园网为例，进行网络流量异常检测算法和系统的研究。清华大学校园网是全球规模最大、架构最复杂、流量场景最多变的校园网之一，具有以下几个特点：

1. 用户规模大，流量峰值高。每天有 10 万台设备活跃在线，同时在线设备最多为 7 万台，峰值流量约为 30Gbps.
2. 用户应用类型多，远比一般的企业网复杂，在网络环境中几百种应用同时使用，这给数据分析带来了很多困难。
3. 异常流量是常态。扫描流量、攻击流量占比多。

为了应对这些困难和挑战，本文在以下几个方面对流量异常检测进行了研究：

1. 本文设计了一种基于特征关系图的循环神经网络算法 (FG-RNN)。在复杂的网络流量环境下，流量特征种类繁多，且特征之间的相关性会随着流量变化而变化。原有的异常检测算法通常直接利用提取的特征进行训练，本文提出的 FG-RNN 算法有效利用了特征之间的相关性信息，将其加入到神经网络的训练过程中。经过在广泛应用的开源数据集、清华大学校园网真实数据集下分别进行的实验验证，本文提出的算法检测结果优于现有的基于机器学习、深度学习的算法。
2. 本文对基于特征关系图的循环神经网络算法进行了流式改造，使之能够满足当前实时异常检测的需求。为了应对海量的清华大学校园网流量数据和高速数据流，本文设计了一个基于 Spark Streaming 的实时流量异常检测系统。该系统分为输入模块、模型模块、检测模块三部分，输入模块负责将流量数据

进行窗口划分、特征选择、特征抽取、合并计算，得到的特征矩阵与预训练得到的关系矩阵一同送到模型中进行训练，模型中的参数周期性更新，最后由检测模块对当前流量进行判别，并给出异常流量的类别。

3. 基于该实时异常检测系统，本文在大规模真实数据集上用多个不同的衡量指标对 FG-RNN 和现有神经网络算法进行实验对比，FG-RNN 算法效果均优于其他算法。
4. 本文对系统检测出的端口扫描进行了定量分析，发现其不仅是安全问题，同时会对无线网性能产生影响，严重时会造成 100ms 以上的时延增加。最后通过两种优化策略，可以有效地缓解该问题。

6.2　未来研究和展望

本文通过对清华大学校园网流量进行研究，提出了一种基于特征关系图的循环神经网络算法（FG-RNN），并设计和实现了一个基于 Spark Streaming 的实时异常检测系统。但是其仍然有很多不足之处，本人认为未来的研究可以从以下几个方面着手。

1. 本文 FG-RNN 算法中使用比较简单的基于皮尔森相关系数的关系矩阵计算方式，该方式并不一定是最佳的关系图计算方式。在未来的工作中，可以尝试其他挖掘方法，如 Network Embedding 等。
2. 本文中的异常检测系统只经过了小规模验证，可扩展性需要进一步测试，并且需要更大量的数据和更长时间的运行以验证系统稳定性。
3. 本文中的算法在处理某些特定异常时，召回率极低，例如无法检测出远控木马这类异常，可以从提升召回的角度尝试优化模型。
4. 由于标注样本的缺失，部分异常种类占比很低，因此本文在处理这些特殊异常时效果不佳，未来可以尝试数据增强等方式。
5. 与商业化的网络运维中心（Network Operation Center）平台中的威胁告警系统相比，基于各种算法的流量检测系统通常人力成本、机器开销更低，但是也有很多不足，如难以涵盖特殊情况，未来可以考虑将两者的优势进行结合。

致　谢

衷心感谢我的导师李风华老师，李老师不仅在学术上给予我耐心详细的指导，还在生活上给予我支持，在我面对困难颓废退缩时给我指明方向，一次次地耐心鼓励让我重拾信心。李老师丰富的学识、严谨的科研风格和平易近人的生活态度使我终身受益。

感谢课题组的其他老师，为我提供丰富的科研资源、开放的科研环境，使我开阔了眼界，增长了能力。

感谢实验室的荣成浩、庄淑颖、陈昕、徐超、沈钲晨等同学，我从和他们日常的讨论中获益良多，很多 idea 都迸发在激烈的探讨中。同时感谢宋兆杰、于淼、赵正品、蓝天鸣、李果等同学，很荣幸和你们一起度过三年的研究生生涯。

特别感谢一直以来对我默默支持的父母和女友梁道萍，在求学之路上他们给予我无私的帮助，让我有勇气面对压力克服挫折。

最后，衷心感谢各位专家老师百忙之中抽出宝贵时间参与本文的评审工作。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

个人简历、在学期间完成的相关学术成果

个人简历

1993 年 5 月 20 日出生于河北省定州市。

2011 年 9 月考入吉林大学机械科学与工程学院机械工程及自动化专业，2015 年 7 月本科毕业并获得工学学士学位。

2018 年 9 月考入清华大学网络研究院攻读网络空间安全专业硕士至今。

在学期间完成的相关学术成果

学术论文：

[1] Gao T, Li F, Yuan Q, et al. A New Factor Affecting the Performance of Wireless Networks: Port Scan[C]//2021 17th International Wireless Communications & Wireless Networks Conference (IWCMC).(已被 IWCMC 录用，EI 收录)

[2] Wang H, Gao T, et al.Hopping on Spectrum: Measuring and Boosting a Large-scale Dual-band Wireless Network[J]//USENIX ATC '21.(在投)

指导教师学术评语

面向智能运维的流量异常检测是目前网络研究领域的开放性问题。论文选题基于校园网真实运行环境，实现校园网流量异常检测，具有一定的理论意义和现实价值。

论文完成的主要工作包括:

1. 提出了一种基于特征关系图的循环神经网络算法，通过引入特征关系图，进一步强化多维数据中的重要维度数据对机器学习效果的影响，提高流量异常检测算法的准确率和召回率。
2. 基于该神经网络算法，针对校园网真实流量环境，设计和实现了基于 Spark-Streaming 的异常流量检测系统，该系统通过引入特征关系图更新机制来避免传统机器学习算法的时效性问题。
3. 论文还对流量异常检测过程中发现的扫描攻击行为进行了分析，发现了扫描攻击行为对无线网络性能的影响，并提出了针对扫描攻击的防御策略。

论文工作表明，作者在本学科掌握了较为坚实的基础理论和系统的专门知识,具有独立承担专门技术工作的能力。论文论述清晰，表述规范，达到了硕士学位论文水平。

建议组织硕士学位论文答辩。