中图分类号: 学校代码: 10055
UDC: 密级: 公开

南開大學

硕 士 学 位 论 文

面向大规模网络的 VoIP 流量实时识别研究

A real-time identification system for VoIP traffic in large-scale networks

论文作者 王荣康 指导教师 张建忠教授
申请学位 工学硕士 培养单位 计算机与控制工程学院
学科专业 计算机科学与技术 研究方向 网络与信息安全
答辩委员会主席 评 阅 人

南开大学研究生院
二〇一九年五月

南开大学学位论文使用授权书

根据《南开大学关于研究生学位论文收藏和利用管理办法》，我校的博士、硕士学位获得者均须向南开大学提交本人的学位论文纸质本及相应电子版。

本人完全了解南开大学有关研究生学位论文收藏和利用的管理规定。南开大学拥有在《著作权法》规定范围内的学位论文使用权，即：(1) 学位获得者必须按规定提交学位论文(包括纸质印刷本及电子版)，学校可以采用影印、缩印或其他复制手段保存研究生学位论文，并编入《南开大学博硕士学位论文全文数据库》；(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等场所提供校内师生阅读，在校园网上提供论文目录检索、文摘以及论文全文浏览、下载等免费信息服务；(3) 根据教育部有关规定，南开大学向教育部指定单位提交公开的学位论文；(4) 学位论文作者授权学校向中国科技信息研究所和中国学术期刊 (光盘) 电子出版社提交规定范围的学位论文及其电子版并收入相应学位论文数据库，通过其相关网站对外进行信息服务。同时本人保留在其他媒体发表论文的权利。

非公开学位论文，保密期限内不向外提交和提供服务，解密后提交和服务同公开论文。

论文电子版提交至校图书馆网站：http://202.113.20.161:8001/index.htm。

本人承诺：本人的学位论文是在南开大学学习期间创作完成的作品，并已通过论文答辩；提交的学位论文电子版与纸质本论文的内容一致，如因不同造成不良后果由本人自负。

本人同意遵守上述规定。本授权书签署一式两份，由研究生院和图书馆留存。

作者暨授权人签字：_____

20　　年　　月　　日

南开大学研究生学位论文作者信息

论 文 题 目	面向大规模网络的 VoIP 流量实时识别研究				
姓　　名	王荣康	学号	2120160423	答辩日期	
论 文 类 别	博士　学历硕士　硕士专业学位　高校教师　同等学力硕士				
院 / 系 / 所			专　业		
联 系 电 话			Email		
通讯地址 (邮编)：					
备注：					

注:本授权书适用我校授予的所有博士、硕士的学位论文。由作者填写 (一式两份) 签字后交校图书馆，非公开学位论文须附《南开大学研究生申请非公开学位论文审批表》。

南开大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下进行研究工作所取得的研究成果。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创作的、已公开发表或者没有公开发表的作品的内容。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。本学位论文原创性声明的法律责任由本人承担。

学位论文作者签名：_____ 年 月 日

非公开学位论文标注说明

根据南开大学有关规定，非公开学位论文须经指导教师同意、作者本人申请和相关部门批准方能标注。未经批准的均为公开学位论文，公开学位论文本说明为空白。

论文题目			
申请密级	限制（2 年） 秘密（10 年） 机密（20 年）		
保密期限	20 年 月 日至 20 年 月 日		
审批表编号		**批准日期**	20 年 月 日

南开大学学位办公室盖章（有效）

限制 2 年（最长 2 年，可少于 2 年）

秘密 10 年（最长 5 年，可少于 5 年）

机密 20 年（最长10年，可少于10年）

摘要

近年来，VoIP 应用凭借其丰富的服务和低廉的价格已经成为了重要的通信服务手段。不幸的是，VoIP 服务为人类带来便利性的同时，也造成了诸多的社会问题。诈骗分子利用 VoIP 所提供的便利的服务实施犯罪活动，对我国诈骗案件的侦破造成了极大的困难。我国反 VoIP 作案面临严峻的局势，为了使 VoIP 进一步服务于人类，对恶意 VoIP 的监管是必要的。

网络流量识别是一个较热门的研究领域，主流的流量分类方法包括基于端口的识别方法，基于字节、字符等的模式识别方法，基于行为的分析方法和基于机器学习的识别方法等。由于 VoIP 应用具有加密和 P2P 特性，以上分类方法大部分很难单独应用于 VoIP 流量识别，目前较为成熟的 VoIP 流量识别方法普遍采用多种方法结合的策略。另一方面，VoIP 流量识别要求较高的实时性，一些基于流特征进行识别的方法不能被应用于实时识别。此外，对 VoIP 流量建立特征集的任务也是极其繁琐的。

本文针对上述问题进行研究，第一，系统地归纳了各类 VoIP 识别方法，包括非实时的识别和实时的识别方法，并且从不同的识别层面上作出比较；第二，本文围绕 VoIP 应用类型采用 CLNN（卷积神经网络，长短期记忆网络）模型提取特征，提取特征的过程不但不需要人为参与，并且提取的特征具有更高的精确度。第三，基于本文提出的方法，我们设计并实现了一个 VoIP 实时识别系统，该系统采用 Apahce Storm 作为流处理计算引擎，支持在大规模的网络中采集并识别 VoIP 流量。实验结果表明，该系统可以实时的准确识别 VoIP 流量。

关键词： VoIP；流量识别；实时；卷积神经网络；长短期记忆网络

Abstract

Recently, owing to the service and price advantage, VoIP has become an important communication technology. Unfortunately, while providing convenience services, it also causes some social tragedies. Swindlers handle VoIP services provided by VoIP to carry out criminal activities, which causes great difficulties in solving fraud cases. We are facing severe situation to against VoIP frauds, in order to make VoIP applications serve humans better, it is important to keep VoIP applications under supervision.

Traffic identification has been an active research topic in the past decade, several methods have been given, which can be generalized as port-based, pattern recognition, behavioral analysis, machine learning methods. Due to the encryption and P2P characteristics of VoIP applications, the above identification methods are difficult to be applied to VoIP traffic identification alone, researchers generally adopt a combination of several methods. On the other hand, VoIP identification has strict real-time requirements, the methods based on flow features are not applicable to real-time identification. It is troublesome to extract features for VoIP traffic.

Firstly, various VoIP identification methods are summarized and we compare them from different identification levels; Secondly, we adopt CNN(Convolutional Neural Networks) to extract features for accurate VoIP application identification. These extracted features are not only more reliable than the features extracted by humans, but also greatly improve the identification efficiency; Thirdly, we design a real-time identification system with Apache Storm, it can capture VoIP traffic in a large-scale network and identify their application types with the features we trained. The evaluation results verify that our system can identify VoIP traffic timely and accurately.

Key Words: VoIP; application identification; real-time; CNN; LSTM

目录

第一章 引言

第一节 研究背景

近年来，VoIP（Voice Over Internet Protocol）技术发展迅速。VoIP 是一种语音通信技术，可提供网际协议作为媒介进行语音传输的环境。包括智能手机、计算机和 VoIP 电话在内的各类终端可在蜂窝网络、Wi-Fi 和以太网等各种网络环境中进行语音通信。与传统的公共交换电话网络（Public Switched Telephone Network，PSTN）相比，VoIP 服务具有低通话成本、低建设成本、高扩展性、服务丰富等优势，因此，VoIP 很快获取了人们的青睐。此外，公共交换电话网络可以通过 FXO 语音网关接入 VoIP 网络，从而实现两种网络之间的上下车应用，各种各样的 VoIP 应用也应运而生。

VoIP 服务在创造大量社会价值的同时，也造成了诸多的社会问题。来自通信欺诈管制协会的报告显示，电信欺诈所造成的损失以每年 29% 的速度增长。其发布的一份报告显示，2017 年仅与 VoIP PBX（专用交换机）相关的黑客攻击行为所造成的价值损失就高达 19.4 亿美元[1]。据估算，每年因 VoIP 欺诈事件所造成的全球总损失达 300-500 亿美元。在国内，VoIP 欺诈案件普遍表现为金融诈骗，诈骗分子利用 VoIP 所提供的改号服务冒充证券银行相关工作人员实时金融诈骗。诈骗分子通过将 VoIP 呼叫系统架设于境外代理服务器，以匿名接入的方式对中国境内电话呼叫实施诈骗。VoIP 所提供的服务允许诈骗分子修改主叫号码，操作路由跳板，对我国诈骗案件的侦破造成了极大的困难。由中国信息通信研究院发布的《信息通信行业防范打击通讯信息诈骗白皮书》显示，2016 年全国共立通讯信息诈骗案件 63 万起，上半年造成经济损失 80.4 亿元[2]。《2017 年中国反通讯网络诈骗报告》显示，每 16 个用户中就超过一个收到了诈骗电话[3]。另一方面，VoIP 因其加密性而难以追踪经常被恐怖组织所利用。早在 2005 年，时代周刊就指出 Skype 为大规模技术盲点。2008 年，英国每日邮报报道，阿富汗境内的塔利班恐怖武装使用 Skype 逃避军情六处的侦查[4]。2012 年，印度时报报道，伊斯兰恐怖组织虔诚军已经开发了自己的 VoIP 网络 Ibotel 用以躲避侦查[5]。

为应对以上问题，各国都对 VoIP 技术有不同程度的监管。在我国，对 IP 业务牌照审核有严格的要求，实行适度开放 VoIP 业务的政策。这也意味着未经授权的 VoIP 业务是非法的，并且所有合法的 VoIP 业务也必须处于监管之下以应对任何程度的紧急情况。但是随着技术的进步，越来越多的 VoIP 应用开始采用加密和对等网络的方式，甚至有一些大型厂商采用自己的 VoIP 协议，这些都使得 VoIP 识别任务愈发艰难。大数据时代的到来为 VoIP 识别提供了新的思路，VoIP 数据规模的增加使得越来越多的信息可供挖掘和分析，通过所挖掘的信息识别 VoIP 流量。本文将结合深度学习和大数据处理技术，对大规模网络中的 VoIP 流量进行实时识别，以支撑网络管理员对非法 VoIP 流量进行监管。

第二节 研究现状

在过去的十几年内，网络流量分类一直是一个研究热点。同时因为 VoIP 技术的发展，针对 VoIP 的网络流量分类吸引了很多研究者，许多有价值的工作也随之产生。

同其他流行的应用一样，早期的 VoIP 应用也使用由 IANA（The Internet Assigned Numbers Authority，互联网数字分配机构）所指定的端口号。但是随着 P2P 和 NAT 技术的发展，许多厂商开始在自己的 VoIP 软件中使用这两项技术，VoIP 应用不再局限于使用固定端口号，而是采取动态端口映射的方案。这就使得传统的基于端口号的识别方法不再有效。为了克服这种困难，一些研究人员[6,7] 通过识别某些特定协议如 SIP，RTP 协议所携带的特征来进行 VoIP 流量识别。一些研究[8] 则从应用级别的角度对 VoIP 应用采取 DPI（Deep Packet Inspection，深度包检测）的方式提取相关特征进行 VoIP 识别。这些识别方法仅仅可以识别分析过的 VoIP 应用，不可能做到对更多 VoIP 应用的识别，并且这种方式对于 Skype 这种私有协议是无效的。其次，这些方法不能识别采用了加密技术的 VoIP 应用。

基于统计分析的方法是另一种经常被用于检测 VoIP 流量的方法。诸如数据流的包长度、包到达间隔、持续时间和分组交换率等统计特征被分析用于检测 VoIP 流量[9]。一些研究方法[10] 结合流的统计特征分析和主机行为估计方法，通过设置参数 D 用于过滤非 VoIP 的流量，之后通过 EL（large inter-packet time，最大包到达间隔）和 ES（small inter-packet time，最小包到达间隔）计算出的参数 R 再次准确的识别 VoIP 流量。与此同时，一些最新的研究[11,12] 仍然在

表 1.1 IANA 指定的 VoIP 默认端口

VoIP 协议	端口	传输层协议
SIP	5060	TCP/UDP
SIPS	5061	TCP/UDP
H.323	1300, 1718-1720	TCP/UDP
MGCP	2427	TCP/UDP
MEGACO/H.248	2944, 2945	TCP/UDP
RTP/RTCP	5004, 5005	TCP/UDP

采用基于规则和统计分析的方法检测 VoIP 流量。这些方法需要根据对应的规则设置很多的阈值参数，越来越多的 VoIP 应用会导致这些阈值参数有很大的出入。随着机器学习技术的发展，基于统计分析的方法也被赋予了新的生命力。一些研究[13] 提出了使用机器学习的方法识别加密的 VoIP 流量。他们建立了基于流的特征集并且使用了 3 中机器学习的方法来训练数据集，3 中机器学习方法有不错的识别率，其 C5.0 获得了最优的效果。机器学习的方式减轻了研究人员在设定阈值方面的负担，但是终归还是基于统计分析的方法，并且对于更新速度越来越快的 VoIP 应用来说，每种 VoIP 应用都可能在短时间内更新自己的算法，这样就需要认为的重新进行统计分析来适应 VoIP 的更新。尤其中作者提出的方法是基于整个 VoIP 数据流训练数据集，训练所得的分类器并不适用于实时的 VoIP 流量识别。从流量识别级别的角度来说，上述方法大多应用于检测 VoIP 流量，也就是说，仅仅针对 VoIP 流量与非 VoIP 流量进行识别，不能从应用级别的角度进行识别。

VoIP 流量识别最大的难点是要求更高的实时性，一些研究也从实时识别的角度进行 VoIP 识别的研究工作。为了实时的检测 VoIP 流量，数据包长度在 60-150 字节内的 UDP 数据包将会被判断成 VoIP 数据包[14]。在这篇文章中，作者仅仅致力于提高 VoIP 的服务质量，该方法不追求较高的识别率所以不能被应用于准确的 VoIP 实时识别。一种基于 VoIP 会话熵变和解编码器类型的方法被提出[15]，文章考虑了几种流行的 VoIP 应用并研究了它们的解编码器类型，通过解编码器类型可以初步的识别一些 VoIP 应用。为了更加准确的识别 VoIP 应用，他们研究了 N 个连续数据包之间的长度关系，并且用熵来衡量它们之间的异质性，通过解编码器类型和熵的异质性来识别 VoIP 流量。从实时识别的角度，他们采取了具有 N 个长度的滑窗计算这 N 个数据包的异质性，通过识别 N 个数据包就可以识别整个流的 VoIP 应用类型。相同的实时识别策略也被

文章[16] 采纳，作者通过提取整个双向流的前几个数据包的 PSD（Packet Size Distribution，包长分布）用于识别 VoIP 应用，实验结果显示这种方式可以在双向流建立之后立即识别出 VoIP 应用类型。目前为止所有的 VoIP 实时识别方案都是基于识别子流的方式从而得知整个数据流的应用类型，这种方式的识别精度与所采用的滑窗大小有直接关系，采用的滑窗太小会导致识别精度太低，滑窗太大会降低实时性。尤其因为识别所采用的特征太少，文章[15] 采用两个特征识别数据流，文章[16] 采用一个特征识别数据流，因此识别精度不是很高。

第三节 本文主要工作

本文主要针对 VoIP 流量识别两个难点展开研究，研究的识别方法通用与加密和非加密 VoIP 应用。两个研究难点，其一是识别精度的问题，提高识别精度的根本在于寻找更加精确的特征集；其二是保证识别的实时性，提高实时性要求减少识别所需的数据包长度。本文研究了现有的 VoIP 流量识别的方法，并对卷积神经网络和长短期记忆网络进行了阐述。利用卷积神经网络在频域方面的建模能力和长短期记忆网络在时域方面的建模能力提取精度更高的特征集。本文针对多种 VoIP 应用建立数据集用于有监督的训练。综合研究成果，本文将训练所得分类器应用于实时识别系统，该系统可支撑在大规模网络中实时识别 VoIP 应用。

1）VoIP 流量识别综述。本文对几种常用的流量识别方法进行了归纳和总结，并将其与 VoIP 流量识别做了比较，总结了 VoIP 流量识别的难点。对现有的几种 VoIP 流量识别方法进行了详细介绍，分析了其优缺点和适用性。

2）基于 CLNN 模型的 VoIP 识别方法。本文研究了 CNN 和 LSTM 的适用性，结合了卷积神经网络在频域方面和长短期记忆网络在时域方面的建模能力，通过研究多种网络模型设计了适用于 VoIP 流量的网络模型。该模型可以提取最优的特征集，避免了人工提取特征集的繁杂工作，并且具有良好的识别性能。

3）VoIP 实时识别系统的实现。对 CLNN 模型训练所的的特征集，本文使用了 4 中机器学习方法重新训练分类器，这些分类器较 CLNN 模型具有更好的实时性。为处置大规模网络中的流量，本文采用了 Apache Storm 作为流式计算引擎，拓扑结构中包含多种分类器 Bolt，网络管理员可以根据紧急程度采用不同的识别结果。

第四节 本文的组织结构

本文的结构安排如下：

第一章为引言，主要介绍了 VoIP 流量识别的研究背景和研究现状，阐述了 VoIP 流量识别的必要性和目前的识别方法所存在的不足，最后介绍了本文的主要研究内容。

第二章为相关研究，介绍了 VoIP 相关技术并比较了传统的流量识别与 VoIP 流量识别的异同，详细的分析了现有研究方法的适用范围和优缺点。最后，介绍了本文所采用的卷积神经网络和长短期记忆网络的基本原理。

第三章为基于 CLNN 网络的特征提取相关研究。该章首先分析了 VoIP 流量所携带的可支持识别的特征，其次介绍了本文所采用的用以支撑实时识别的方案。最后介绍了如何构建 VoIP 流量数据集并使用 CLNN 模型训练数据集提取特征集，详细介绍了 CLNN 模型的设计。

第四章为 VoIP 流量实时识别系统的实现。该章介绍了实时系统的总体框架。之后从离线训练和在线识别两个方面介绍了系统的实现。

第五章为实现结果以及性能评估。评估了 CLNN 模型提取的特征集，并对 4 种机器学习训练所得的分类器进行了评估。

第六章总结了全文的工作，分析了本文研究工作的不足之处以及进一步的研究方向。

第二章 相关研究

VoIP 流量识别是从各种网络协议流量中识别出 VoIP 相关流量，并对其进一步分类的过程。VoIP 流量识别主要用于保障网络安全，可以依 VoIP 应用类型对流量作出不同的处置。本章对多种传统流量识别方法进行了总结分析，并比较了其与 VoIP 流量识别的异同，介绍了 VoIP 识别的难点。最后详细阐述了卷积神经网络与长短期记忆网络的基本原理。

第一节 VoIP 简介

VoIP（Voice over Internet Protocol，基于 IP 的语音传输协议）是一种网络通话技术，经由网际协议来达成语音传输，也就是在互联网上进行语音通信。早期的 VoIP 技术是在个人电脑上通过软件的方式实现的，当时的 VoIP 软件只能支持电脑与电脑之间进行通信。2000 年，VoIP 技术在 IP PBX 的加持下，开始向电信领域渗透。传统的公共交换电话网络可以通过 IP PBX 接入 FXO 网关连接到互联网，从而实现传统公共交换电话与网络电话之间的通信，这也导致了 VoIP 技术的飞速发展。VoIP 技术发展至今，作为宽带电话已经从语音质量、价格等各个方面超越了传统的固定电话。

VoIP 协议包括用于信令控制和媒体传输两方面的相关协议。信令控制协议主要用于呼叫建立、呼叫保持和呼叫终止，信令控制协议包括 H.323，SIP，MGCP 和 MEGACO，它们由不同的标准化组织制定，目前广泛被使用的信令控制协议为 H.323 和 SIP。实时传输协议即 RTP 为语音提供端到端的传输服务，其包括 RTP 和 RTCP 两个子协议，是最主流的 VoIP 媒体传输协议。因为本文的 VoIP 识别方法主要基于 RTP/RTCP 流量，本节将对信令控制协议做简单介绍，详细介绍实时传输协议。

2.1.1 信令控制协议

2.1.2 实时传输协议

第二节 传统流量识别与 VoIP 流量识别方法

第三节 神经网络

第四节 本章小结

第五节 传统流量分类方法

作为网络安全和运营管理的重要技术基础，网络流量分类一直是学术界、产业界和网络监管部门共同关注的热点问题之一。网络流量分类就是依据网络协议或者应用对应的某些参数或特征，自动区分出流量归属的协议或应用。当流量分类是二分类的时候，其概念等同于流量鉴别或流量识别。

传统的流量分类方法大致分为四类：基于端口匹配的方法；基于深度包检测的方法；基于主机行为分析的方法；基于统计学习的方法。

2.5.1 基于端口匹配的方法

在互联网发展早期，IANA 为每个应用都定义了标准端口号，比如 HTTP 协议使用 80 端口，SMTP 协议使用 25 端口等，一直沿用至今。因此，早期的流量分类使用端口匹配就可以实现。基于端口匹配的方法就是将端口号和应用服务关联起来，通过检测 TCP 或者 UDP 包头中的源端口号或目的端口号，鉴别该流量所属应用。表2.1显示了常见的端口协议映射。

表 2.1 常见端口协议映射表

Port	Protocol	Port	Protocol
21	FTP	88	Kerberos
22	SSH	109	POP
23	Telnet	110	POP3
25	SMTP	115	SFTP
37	TIME	143	IMAP4
43	WHOIS	161	SNMP
53	DNS	179	BGP
80	HTTP	443	HTTPS

端口匹配的方法对于识别传统应用简单有效。然而随着互联网的发展，新兴应用不再使用统一分配的端口号，而是使用随机的端口号。这一方面体现了互联网应用的灵活性，但也给基于端口的流量识别带来了困难。特别是随着

P2P 应用的发展，绝大多数 P2P 应用使用动态端口技术，非固定端口。而且有些 P2P 应用会使用 80 端口以逃避网络检测或防火墙的拦截。还有一些 FTP 应用也允许用户手工指定端口，不再使用固定的 FTP 端口。这些都使得基于端口匹配的流分类技术的有效性不断下降。目前端口匹配的方法一般只作为粗粒度流分类的依据。

2.5.2 基于深度包检测的方法

随着端口匹配方法的失效，基于深度包检测的方法应运而生。深度包检测是指，除了对传输层以下部分，对应用层部分甚至是应用层的内容进行检测。一般通过在应用层部分寻找特征字符串，对要识别的流量和事先提取出的特征字符串进行匹配，从而实现流量鉴别。通常称这些特征字符串为应用指纹。例如，P2P 应用 BitTorrent 的有些数据包中包含特征字符串 “0x13BitTorrent”。只要将这些应用指纹和应用关联起来，即可实现流量分类。应用指纹不仅指关键字，也可以是正则表达式。使用正则表达式来描述特征通常比传统的精确提取的关键字更加准确有效。

DPI 方法对许多应用流量的识别都很有效，其中也包括变化多端的 P2P 应用。Sen 等人就将其用于 5 种 P2P 应用的流量检测，并且取得了较好的鲁棒性和准确性[17]。DPI 方法也被应用于一些商业产品和开源工具。L7-Filter[18] 就是典型的支持正则表达式匹配的开源流分类工具，它可以支持数百种主流应用的识别。类似的开源工具还有 OpenDPI、nDPI，这些工具都提供了获取应用指纹的能力。文献 [19] 对 6 种 DPI 开源工具从应用协议和具体应用两个层面的分类效果进行了比较。

对基于 DPI 的方法而言，特征选取的好坏直接决定了分类的质量。特征匹配的算法决定了分类的性能。近几年，许多方法致力于实现 DPI 的自动特征提取。Zhang 等人提出了 ProWord 方法[20]，可以无监督的实现协议特征提取。ProWord 首先利用改进的 Voting Experts 算法对协议负载进行分割，然后定义排序规则对候选特征词进行排序，从而选取可能的特征词。实验表明 ProWord 对于文本协议和二进制协议均可提取有效的特征词。中科院计算所的云晓春等人[21] 提出使用主题模型 LDA（Latent Dirichlet Allocation）方法对流量进行建模，提取协议负载的关键词分布作为特征，然后通过机器学习的方法进行分类。该方法改变了以往的流量特征形式，将 DPI 方法和机器学习方法结合起来，开拓了 DPI 方法的思路。

基于 DPI 的流量分类方法是目前准确率较高的方法。但该技术也有一定的限制：（1）应用指纹难找易变：新兴应用层出不穷，应用版本更新速度快，这都需要不断更新指纹数据库，给数据库的维护带来了挑战。（2）对加密流量不适用：加密流量的特征难于辨认，使用 DPI 的方法效率不高。（3）计算代价高：在主干网上流量负载高，对数据包进行深度检测和模式匹配的算法计算量大，这导致时间和空间代价高，因此有许多针对 DPI 的匹配方法进行加速的研究[22,23]。（4）隐私问题：由于 DPI 的方法需要检测载荷内容，可能涉及用户隐私问题。

2.5.3 基于主机行为分析的方法

为了弥补 DPI 方法的不足，有学者提出基于主机行为分析的流量分类方法。该方法不需要检测应用层的内容，只根据主机在传输层的行为特征来进行流量分类。例如检测一台主机同时连接了多少台其他主机，使用哪些传输层协议，使用哪些不同的端口等。该方法主要基于不同应用产生不同行为模式的思想，例如，使用 P2P 应用的主机通常同时使用不同端口连接许多主机，而一台 Web 服务器会在同一端口连接许多不同的客户机。

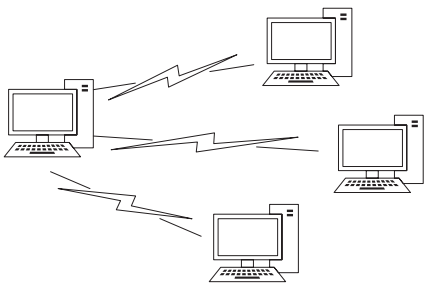

图 2.1 主机行为示意图

Karagiannis 等人 2004 年提出了基于主机连接模式识别 P2P 应用的方法[24]。该方法定义了两个启发式特征，一个是同时使用 TCP 和 UDP 两种传输层协议的 {源 IP，目的 IP} 对，另一个是 {IP，端口} 对。该方法可以高效的识别 P2P 流量，而且还能识别出 DPI 方法未能识别的 P2P 流量。2005 年，Karagiannis 等人又提出了新的方法 BLINC[25]。BLINC 方法在三个层面上分析主机行为：在社会层上分析一台主机一段时间内与哪些其他主机进行通信；在功能层分析一台主机属于提供服务的一方还是请求服务的一方；最后，结合前

两者的结果在应用层结合传输层端口号推断原始应用类型。该方法可以识别出流量的大类别，例如 web、P2P、ftp、mail 等，在实验中识别准确率超过 95%。Schatzmann 等人研究了基于 HTTPS 的 webmail 流量识别，提出利用 webmail 服务器和 webmail 客户端的行为特征以及 webmail 业务的特点作为分类特征，可以识别出 93.2% 的基于 HTTPS 的 webmail 应用的流量[26]。

基于主机行为分析的方法避免了检测数据包的有效负载，具有轻量级的优点。但是该方法也存在一些限制：（1）分类精度通常较低，不能够划分更精细的应用类别。（2）当一些主机使用 NAT 时，对主机的行为分析可能失效，影响分类准确率。

2.5.4 基于统计学习的方法

为了应对加密流量识别等 DPI 方法难以解决的问题，有学者提出了基于统计学习的方法。基于统计学习的方法主要利用流量统计特征实现应用的分类。其基本思想是不同应用流量的持续时间、包长度、包时间间隔等统计信息都不同。因此，通过计算特定应用流量的统计信息，利用各种机器学习算法，可以对流量进行鉴别。和一般的机器学习方法相同，首先提取统计特征，然后利用训练数据建立分类模型，最后利用分类器对未知流量进行分类。

根据机器学习算法的不同可以将基于统计学习的流分类方法分为两类：监督学习的分类方法和无监督学习的分类方法。

监督学习的分类方法需要提前知道数据流或数据包对应的应用，并对数据流或者数据包做好标记。然后用这些标记好的数据样本训练集对分类器进行训练，直到达到一定的识别精度。得到的模型可以对未知应用的流量进行识别。监督学习中的朴素贝叶斯算法、C4.5 决策树算法、SVM 算法等先后用于流分类中，取得了较好的效果。一般的监督学习的分类过程如图2.2所示。

无监督学习就是聚类的过程。无监督学习的方法首先对采集到的数据流量特性进行分析，根据其对应特征采用聚类算法将不同数据流分开，分类结果并没有具体的分类标签。无监督学习的分类方法有 K-Means 聚类算法等。无监督的算法不需要标记数据，这很大程度上解决了流量数据集采集的难题。但是无监督的算法不能直接得出分类标签，使得其使用场景非常有限。也有工作将无监督学习和监督学习结合起来用于流量分类，称为半监督学习。半监督学习的方法通常借助无监督学习的方法进行流量标记。首先用无监督学习的方法对流量样本进行聚类，然后根据已知样本再对各类数据进行标记，这样标记的数据

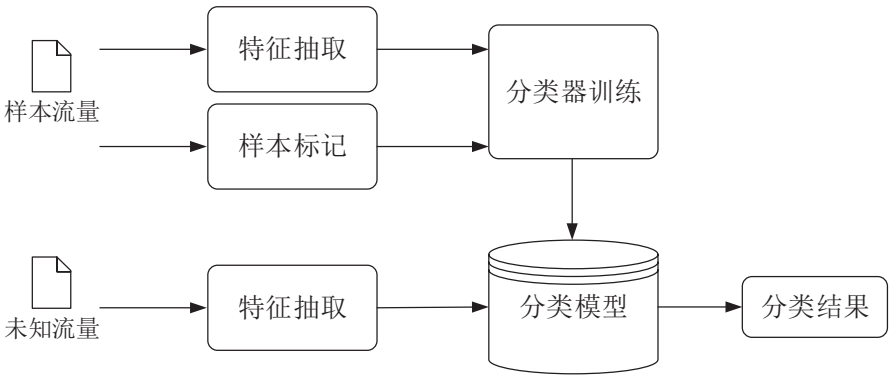

图 2.2 监督学习分类过程

可以用于监督学习的流量分类。

对于基于统计学习的流量分类方法而言，如何选取最少且最有效的流量特征是该分类方法的研究热点之一。2005 年，Moore 等人[27] 提出了可以用于流量分类的 248 个特征，这些特征中有数据包到达间隔的最大、最小、平均值，数据包长度的最大、最小、平均值，TCP 端口等基本的统计特征，还有一些复杂的衍生特征。之后的研究大都选取了这些特征中的一部分用于流量分类。另一个研究热点是分类算法的选取。Moore 等人利用朴素贝叶斯方法对其定义的 10 类流量进行分类，分类准确率高达 95%。2007 年，Moore 等人在其之前研究的基础上采用了贝叶斯神经网络进行流量识别[28]，提高了原来的识别准确率。Li 等人使用了 C4.5 决策树方法和朴素贝叶斯方法，其实验结果表明 C4.5 决策树的分类精度高于朴素贝叶斯方法[29]。Este 等人使用了 SVM 对 TCP 流量进行分类，取得了较高的精度，其划分类别精确到各类应用协议[30]。也有许多研究使用半监督的方法，Bernaille 等人就使用了 K-Means 方法进行流量识别[31]。Ester 等人提出了基于密度的 DBSCAN 半监督方法[32]。Cheeseman 等人提出了基于概率模型的 AutoClass 方法[33]。Erman 等人对这三种半监督方法进行了评估，结果表明 AutoClass 方法准确率最高，但 K-Means 和 DBSCAN 比 AutoClass 聚类速度更快[34]。

基于统计学习的分类方法不需要检测数据包负载，具有轻量级的优点，但是也存在精确度低的问题。基于统计学习的方法目前仍面临一些挑战：（1）对于监督学习的方法来说，难以获得足以代表应用的数据集。（2）多种特征提取计算量大，难以实现实时分类。（3）难以提取有效的流量特征。（4）可移植性较

差，某段时间某个网络的数据集训练的分类器难以适用于不同地点。

第六节 移动应用流量分类方法

随着智能手机的发展，有一些学者开始分析移动应用流量，从而研究移动应用使用行为。2011 年，Xu 等人研究了智能手机应用程序的不同使用行为[35]。该研究基于美国一级蜂窝网络提供商的一周真实流量数据，这些流量数据采集于 2010 年 8 月的一周，并且经过了匿名处理。这些流量中大约包含了 22,000 个移动应用的流量。移动应用使用标准平台 API 时，其产生的 HTTP 流量头部的 User-Agent 域具有标准格式，可以用于代表该移动应用。因此，该研究利用 HTTP 头部的 Use-Agent 域来区分不同移动应用。研究结果显示，对于当时排名前 100 的热门应用，利用 User-Agent 域区分应用的正确率高达 91%。

2012 年，Choi 等人提出一种自动提取移动应用标识符的方法用于移动应用流量鉴别[36]。他们提出在移动手机上安装移动流量代理来监控移动应用流量，从而获取 Ground Truth，并使用 User-Agent 域和 Host 域作为 HTTP 流量的分类标识符，使用最长公共子序列（LCS）方法用于非 HTTP 流量的分类标识符提取。他们的实验结果表明，该方法的平均识别精度可达 99.23%（流大小）和 83.85%（流数量）。

随着移动设备的不断更新，大多安卓应用不再使用 User-Agent 域存放应用标识，而是存放安卓系统版本等没有区分度的信息。使用 Host 字段仍然可以鉴别一些移动应用，但是存在一些限制。首先，有些服务器同时服务于多个移动应用，尤其是在同一个开发者发布多个应用的情况下，Host 字段无法区分这些应用。其次，随着云服务的发展，许多移动应用使用第三方云服务器，多个应用使用同一个服务器更为常见。因此，单纯使用 User-Agent 或者 Host 域的方法在一定程度上已经失效。2013 年，Dai 等人提出使用 HTTP 流量的 Host 域和字符串形式的状态机作为移动应用指纹[37]。为了提取指纹，他们首先利用 UI 模糊技术，自动从多种路径执行移动应用，用来代表该应用的各种行为，同时收集这些流量。然后解析这些 HTTP 流量，把 HTTP 头部分割成不同组件，再根据相似性将这些 HTTP 流量重新聚类，从而形成移动应用指纹。图2.3是应用 Zedge 的状态机指纹实例。他们选择了 10000 个含广告的安卓应用和 6 个不含广告的安卓应用进行了评测，都可以获得较高精度。但是该方法有一定的限制，该方法需要从所有路径执行应用以获取全面的流量，其在生成状态机时也需要

进行大量计算，而且识别流量时需要状态机匹配，总的来说计算代价高。

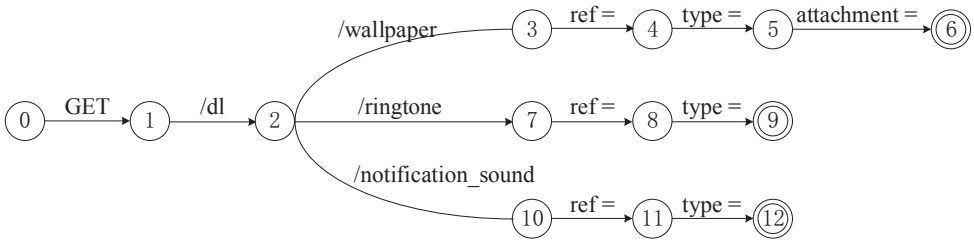

图 2.3 Zedge 的状态机

2014 年，Xu 等人提出了自动生成移动流量指纹的 FLOWR 方法[38,39]。FLOWR 方法基于两点观察：第一，移动应用产生的流量至少有一部分含有可以区分的信息。例如，一些应用会和特定的 Web 服务器相关联，还有一些开发者也会嵌入一些特定信息。第二，流量的时间和拓扑结构有助于应用流量识别。例如，短时间段内从相同设备发出的流量可能来自同一应用。该方法提出以 HTTP 头部的键值对形式为特征。初始状态时，人工提取一些移动应用的键值对指纹，作为初始种子放入指纹库。由于这些种子自身的限制，不一定存在于应用的所有流量中，因此需要进一步添加指纹，以鉴别所有流量。基于第二点观察，FLOWR 利用已有的指纹，提取网络真实流量中短时间段内同时出现的流量特征作为候选指纹。再进行流回归，也就是在真实流量中验证这些候选指纹是否合适，验证通过的指纹也添加到指纹库中用于鉴别特定应用。FLOWR 方法的初始指纹提取非常关键，如果应用覆盖范围小，则提取的指纹也只能识别这些少量应用。因此该研究提出对一些流行广告服务进行逆向工程，提取其在应用流量中嵌入特征字符的模式，获得了覆盖范围比较大的初始指纹库。根据其实验结果，FLOWR 可以鉴别出特定应用的 86-95% 的流量。FLOWR 方法优点在于可以自动学习新的指纹，但其限制在于需要人工参与，对于每个要鉴别的应用都人工提取指纹并不实用。

2015 年，Yao 等人提出了 SAMPLES 方法[40]。该方法首先定义了特征规则，同样利用 HTTP 请求头部各个域作为特征，这些域包含 Host、User-Agent、Method、URL、请求参数等，使用复杂的特征组合用于 HTTP 流的识别。他们结合两种方法获取这些特征，一是逆向分析移动应用的安装包，提取相关静态信息，二是通过分析动态的流量提取指纹。该方法比较系统并具有普适性，但是该方法流量覆盖范围较小。

2016 年，Ranjan 等人提出的 AMPLES 将流识别问题转换为一个信息检索问题[41]。与前面的方法不同的是，该方法不需要执行 app 获取训练流量，只需要分析静态的应用可执行文件。他们从可执行文件中提取出一些可以区分应用的信息，例如，App 标识符、Key-Value 对、URL 等，将不同 app 的这些信息建立成语料库。对于要检测的网络流量，解析出其域名、HTTP 请求参数 Key-Value 对、URL 等信息。将这些信息作为半结构化的文档，发送到搜索引擎中查询，搜索引擎查询之前的语料库，返回一个匹配结果，从而判断出流量所属应用。为了提高流覆盖范围，AMPLES 允许模糊匹配，此时将网络流量映射到一系列应用。

2016 年，Wang 等人强调了域名对于移动应用流量识别的关键性[42]，对于 HTTP 协议而言域名就是 Host 字段，他们发现 73.9% 的应用程序至少有一个功能域名以保证应用的正常运行，根据功能域名他们可以预测出与其相关的应用，F-Measure 在 90% 左右。

除了以上这些深度检测 HTTP 请求报文的方法，还有方法使用流量统计特征实现 Android 应用流量的识别。Alan 等人提出使用 Android 应用启动时产生的流量的 TCP/IP 头部识别应用，这样不仅能够识别 HTTP 流量，还能够识别出加密的 HTTPS 流量[43]。他们将三种针对 Web 流量识别的方法 [44–46] 应用到移动应用流量识别，并分别进行了评估。实验表明，使用同样设备产生的数据集进行训练和测试时，仅用 App 启动时产生的前 32 至 64 个数据包就可以识别应用，准确率达到 88%。然而不同手机、不同应用版本等因素也对识别效果有很大影响，如何提高这些统计方法的鲁棒性有待研究。

第七节　卷积神经网络

随着机器学习方法研究的逐渐成熟，尤其是人工神经网络的发展，研究者提出了深度学习的方法。2016 年，AlphaGo 与李世石的围棋大战引起轰动，AlphaGo 的胜利将深度学习的这一概念带入了众人的视线，一时间深度学习技术成为热点话题。本文提出的方法用到了深度学习中的卷积神经网络，因此本节主要介绍卷积神经网络的研究进展以及其基本原理。

2.7.1 研究进展

卷积神经网络（Convolutional Neural Networks，简称 CNNs）是深度学习技术中具有代表性的一类网络结构，它在图像处理领域取得了很大的成功。它最早起源于人工神经网络。早在上个世纪，就有学者提出了人工神经网络的概念[47]。卷积神经网络是结合了生物学研究的一种人工神经网络模型。1962年，Hubel 等人第一次提出了感受野的概念，他们研究发现，视觉信息从视网膜传递到大脑是通过多个层次的感受野激发完成的[48]。基于这项生物学研究，Fukushima 等人提出了基于感受野的模型神经认知机[49]（Neocognitron），这是感受野在人工神经网络领域的首次应用，也是卷积神经网络最早的模型。1998年，Yann LeCun 等人提出了 LeNet-5[50]，成功将卷积神经网络应用于手写字符的识别。LeNet-5 总共有 7 层结构，其中有两层是卷积层。2012 年，Krizhevsky 等人提出了 AlexNet[51]，进一步把卷积层加深到了 5 层，该模型在 ImageNet 图像竞赛中将图片分类问题的 Top5 错误率降低至 15%，领先第二名 11 个百分点，使得卷积神经网络成为了研究热点。在这之后，不断有人提出新的卷积神经网络模型，比如谷歌的 GoogLeNet、微软的 ResNet 等，进一步提高了图像分类精度。近几年，卷积神经网络也逐渐应用于语音识别[52,53]、自然语言处理[54,55] 等领域。

2.7.2 理论基础

卷积神经网络解决了机器学习中的一大难题，即特征选择。一般的机器学习过程首先要经过数据预处理、特征提取、特征选择，然后再进行模型训练，最后将得到模型用于预测或识别。大多数的研究主要针对模型，也就是机器学习的算法部分。而特征选择也非常关键，算法的准确性一定程度上取决于特征的选取。而机器学习的特征选取通常是由人工完成的。手工选取特征需要很强的专业知识，能不能选取好依赖经验和运气，而且特征选取过程中的测试、调节等工作耗时耗力。卷积神经网络的出现解决了这一问题，它可以自动学习特征，不再需要人工选取特征。而且通过卷积神经网络得到的特征通常比人工设计的特征具有更强的判别能力和泛化能力。

传统的神经网络如图2.4所示，图中的网络有四层，包括输入层、两个隐藏层和输出层。卷积神经网络与普通的神经网络相似，网络由几层的神经元连接而成，这些相邻的神经元之间有部分连接，这些连接对应着一定的权重参数，而

这些权重参数需要通过学习获得。

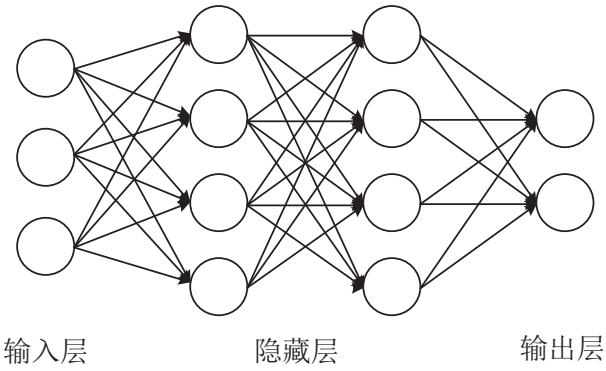

图 2.4 神经网络一般结构

卷积神经网络在图像处理中的巨大成功主要由于其自身的几个特点。传统的神经网络使用全连接的方式，这使得两层神经元之间连接数量多，意味着参数量多，训练非常耗时。而卷积神经网络受到生物视觉研究的启发，通过局部连接、权值共享等方法减少了参数，加快了训练速度。

1）局部连接

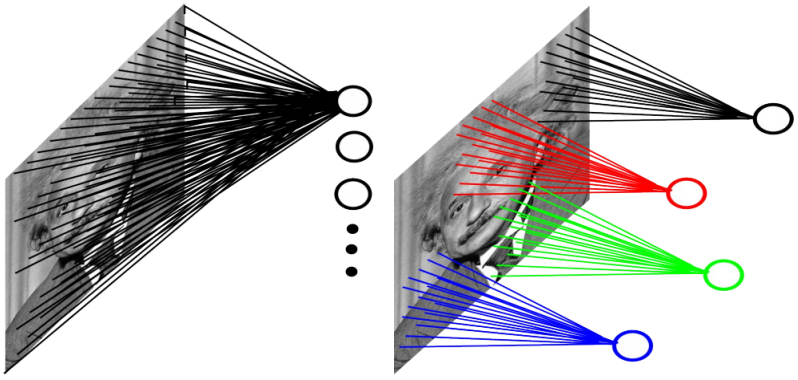

图 2.5 全连接和局部连接

卷积神经网络的设计模仿了生物视觉的处理行为，采用了局部连接的方式，先学习局部特征，然后再综合这些特征。如图2.5所示，200×200 的图像，左图全连接情况下 40 万隐藏单元对应有 160 亿个参数；右图局部连接情况下，每个单元和 10×10 的局部图像相连接，40 万个隐藏单元对应的参数只有 4000 万个。参数的数量大大减少。

2）权值共享

尽管局部连接已经减少了参数数量，然而参数数量仍然较多。因此卷积神经网络还采用了权值共享的方式。在图2.5右图中，每个神经元都对应 10×10 的参数，40 万个隐藏单元对应不同参数时，参数总量是 4000 万个。如果 40 万个隐藏单元共享参数时，只需要 10×10 的参数。这 10×10 的参数可以称为卷积核或过滤器，本质上就是图像特征的抽象。之所以可以权值共享，主要原理是适用于局部图像的特征有可能在其他地方也适用，使用相同的卷积核就是在一幅图像的所有位置探测同样的特征。卷积过程中，卷积核就像一个筛子，将图像中符合条件部分的筛选出来。

使用一个卷积核仅能提取图像的一种特征，要多提取出一些特征可以增加多个卷积核，比如 100 个卷积核可以学习 100 个特征，而总共需要学习的权重参数也只有 $10\times10\times100$ 个。

3）网络结构

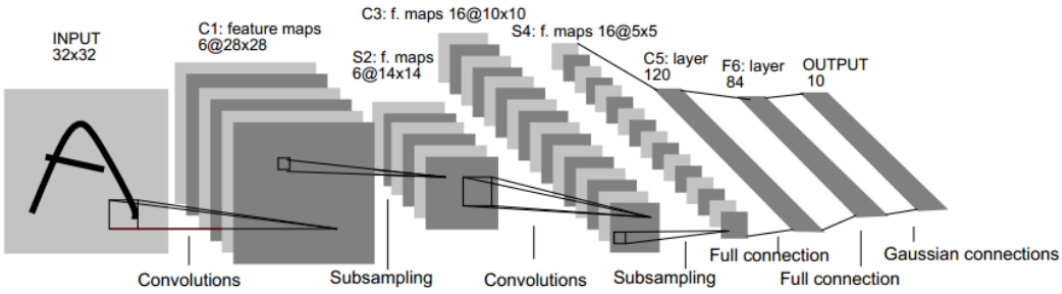

图 2.6 LeNet-5 结构

图2.6是 Yann LeCun 等人提出的 LeNet-5 结构，一个经典的卷积神经网络结构。从中可以看出，CNNs 主要由两种网络构成，一种是卷积层，一种是池化层（子采样层）。其中，卷积层的作用是提取特征，而池化层通常在卷积层后面，对卷积层得到的特征图的局部区域取平均值或者最大值，其作用是降低卷积层输出的特征向量维度，同时改善过拟合的状况。整个网络由多个卷积层和池化层构成，最后使用全连接层，多层卷积的主要目的是从局部到整体、从低级到高级的逐步提取全局特征。

第八节 本章小结

本章首先介绍了传统的流量分类方法的基本原理，并分析了各类算法的优缺点和使用场景。其次，本章还详细介绍了近几年具有代表性的关于移动应用流量分类方法的研究，描述了这些方法的分类机制，并分析了这些方法的优缺点。最后，本章还对卷积神经网络的研究进展进行了介绍，分析了卷积神经网络的基本原理，并通过图像识别领域中经典卷积神经网络的例子，说明了卷积神经网络的一般结构。

第三章 基于 CLNN 模型的 VoIP 流量实时识别研究

在第二章我们提到，本文针对在媒体传输阶段所产生的流量进行特征提取。第一，由于加密技术的发展，SSL/TLS，WEP 和 WAP/WAP2 等加密技术被广泛的应用于信令控制阶段。但是由于媒体传输阶段要求更好的服务质量，考虑到加密技术需要耗费更多的计算资源，大多数 VoIP 应用并没有在其媒体传输阶段使用加密技术；第二，对于少量在媒体传输阶段使用机密技术的 VoIP 应用，因为媒体传输阶段所产生的流量要远远超过信令控制阶段，在大数据技术的支持下，媒体传输阶段的流量相比信令控制阶段可以提取更多的特征用于 VoIP 流量识别；第三，VoIP 在信令控制阶段所采用的应用协议种类繁多，下层协议也包括 UDP 和 TCP，而在媒体传输阶段普遍采用 UDP 进行传输，上层协议也普遍采用 RTP 进行传输，对媒体传输阶段的流量进行识别可以设计更加通用的识别方案。

为了提取有效的特征进行 VoIP 流量识别，本章将从分析 VoIP 媒体流量所携带的特征出发，分析使用 CLNN 模型提取特征的可行性，进而介绍本文进行实时识别所采用的方案以及设计的 CLNN 网络模型。最后将会介绍本文如何使用提取的特征集进行在线实时识别。

第一节 VoIP 特征分析

3.1.1 基本网络特征

RTP/RTCP 是 UDP 的上层协议，其报头可以作为识别 VoIP 流量的基本网络特征。由于本文所研究的 VoIP 流量主要针对于语音流量。RTP 中的 M 和 PT 标记位都可以作为特征用于区分视频和语音流量。RTCP 在对 RTP 媒体传输质量提供反馈信息的同时，会根据与会者的数量调整自己的发送速率。RTP 和 RTCP 数据包普遍采用临近端口号可以作为 VoIP 流量识别的另一特征。

负载类型（PT）是 VoIP 流量最主要的网络特征，用于指明发送端对有效负载采用的解编码器类型。由于语音在不同的通信网络中由不同的解编码器解编码，如果发送端在会话或者广播的中途决定改变编码方式，可以通过设置 RTP 报头中的负载类型位来通知接收端，接收端也可根据负载类型位进行解编码的

工作。RTP 报头中的负载类型预留位为 7 位，意味着 RTP 可以支持 128 种不同的负载类型。VoIP 应用会在不同的网络中基于语音质量和带宽要求等相关条件选择合适的语音解编码器，每种 VoIP 应用所支持的语音解编码器不同。针对本文捕获的 VoIP 流量，表格3.1展示了我们分析所得的 VoIP 应用所使用的部分解编码器类型。RTP 所支持的更多的负载类型可以在 IANA 文档[56] 进行查看。从解编码器的另一角度来看，不同的解编码器会使用不同的比特率传输语音，其与 VoIP 的各项统计特征直接相关，我们将在章节3.1.2进行详细介绍。

表 3.1 VoIP 应用使用的解编码器

应用类型	解编码器类型
Skype	dynamic (96-127), SILK, G.729, PCM A/U, GSM
UUCall	G.723, G.728
KCCall	PCM A/U, ILBC, GSM, G.729
Jumblo	PCM A/U, dynamic (118)
Zoiper	PCM A/U, GSM
Xlite	PCM A/U, iLBC, GSM, Speex
Eyebeam	PCM A/U
ExpressTalk	PCM A/U, Speex
Bria	PCM A/U, Speex, GSM

3.1.2 统计特征

本节所涉及的各项统计特征都与 VoIP 应用所使用的解编码器类型直接相关。许多解编码器是专门为 VoIP 设计的，PCM A/U 解编码器作为最常用的 VoIP 语音解编码器，这两种编码器都生成以 8kHz 采样的 64kbps 的恒定比特率（Constant Bit Rate，CBR）语音流，与其他解编码器相比可以提供较好的语音质量，但同时也较其他具有低比特率的解编码器占用更多的带宽资源。G.723 和 G.729 作为最流行的语音解编码器，占用的带宽比较低，其中 G.723 支持在 5.3kbps 和 6.3kbps 两种码率下工作，G.729 支持在 8kbps 的码率下工作。G.728 也是一种使用恒定比特率的解编码器，它使用 16kbps 的恒定比特率运行。iLBC 是对丢包具有处理能力的专为包交换网络通信设计的解编码器，优于目前流行的 G.729 和 G.723，它可以以 13.33kbps 和 15.20kbps 的比特率工作。GSM 和 Speex 是两种可变比特率（Variable Bit Rate，VBR）解编码器，支持宽范围的比特率，也被广泛的应用于 VoIP 通话技术。多数解编码器都采用 20ms 的打包周期，G.729 采用 10ms 的打包周期，iLBC 在 13.33kbps 下采用 30ms 的打包周

期。

各种解编码器比特率和打包周期的不同，直接造成了 VoIP 流量所占带宽和包到达间隔的不同，同时也间接造成了数据包大小的不同。

3.1.2.1 数据包大小分布

数据包大小分布（Distribution of Packet Size，PSD）是进行 VoIP 流量识别最常用的统计特征。相关工作中介绍的研究方法[16] 直接采用 PSD 作为 VoIP 流量识别的标准，研究方法[15] 提到的数据包长度的熵变化也是数据包大小分布的表现形式。

3.1.2.2 包到达间隔分布

第二节 基于 CLNN 模型的特征提取研究

3.2.1 实时识别基础

为了进行实时识别以达到流量控制的目的，可以通过在 VoIP 会话建立的初期识别出该 VoIP 会话。如果一个 VoIP 会话在可以接受的时间内被识别出，我们认为该识别满足了实时性的要求。假设，在一个组织内部是禁止使用某种 VoIP 应用的，如果我们可以在几秒钟之内准确的识别出一个 VoIP 流，之后在网关处拦截属于该 VoIP 流的全部流量，这就达到了我们实时识别的目的。

根据 VoIP 选择的解编码器的不同，构造并发送数据包的时间在 10ms-30ms 之间，这意味着 VoIP 应用在 1 秒中可以发送 33-100 个数据包供我们进行识别。本文将通过为 VoIP 子流构建数据集并提取特征集的方式，通过所提取的特征集识别 VoIP 子流从而确定整个流所属的应用类型。

3.2.2 VoIP 子流数据集构造方法

本文采用有监督的学习方法训练数据集获取特征集，构造和标记数据集对于很多研究人员来说是一项繁琐的工作。本节我们将介绍构造适用于实时识别的带标签的数据集的方法。

3.2.2.1 数据集构造

数据集构造过程由收集原始数据、标记数据和拆分子流三个步骤构成。

为了获取收集原始 VoIP 流量，我们在真实网络环境中部署了多台电脑用于收集实验数据集。标记数据集的过程实际上是随着收集原始数据完成的，我们在 Windows 操作系统下采用了基于进程的流量收集工具 QPA，在 Linux 系

统下通过确定进程标识符，并按照进程标识符进行流量收集的方式。收集原始数据的过程实际上已经将原始数据按照进程名称或者进程标识符进行标记。

拆分子流的过程是将 VoIP 完整流拆分成多个子流的过程，我们将拆分后的单个子流叫做子流 ^{k}，k 代表该子流中所包含的数据包个数。为了保证子流数据集的随机性，我们设计了基于滑窗的采样方法。从索引为 0 的数据包开始，我们采用大小为 k 的滑窗选择子流 ^{k} 进入我们的数据集，滑窗的步长是在 [1, k] 中随机选取的。这种方式不但保证了数据集的随机性，同时也节省了大量的训练所需时间。图片3.1展示了滑窗大小为 10 时选择 4 个子流 ^{10} 的过程。

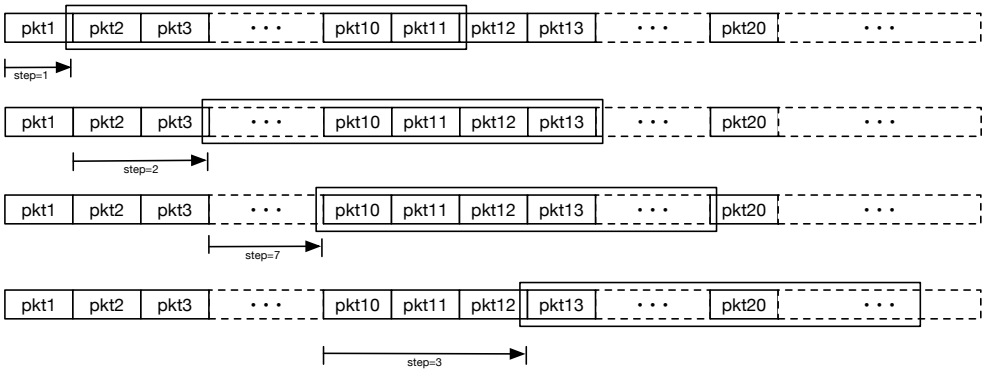

图 3.1 滑窗大小为 10 时选择 4 个子流 ^{10}

3.2.2.2 数据预处理

数据预处理的过程是将数据集处理成可供 CLNN 模型学习的训练数据的过程。本文提出的方法致力于识别 RTP/RTCP 数据流所携带的特征进而识别出 VoIP 流量。由于网络地址转换（Network Address Translation，NAT）技术的发展，对于我们来说，IP 报头所携带的信息是没有识别价值的。UDP 报头所携带的端口号信息对于我们识别 RTP/RTCP 数据包有一定的价值，因此我们保留 UDP 端口号信息。

数据预处理的过程主要包括五个步骤：1）移除网络层和传输层头部；2）为数据包添加传输层端口号；3）添加 2 个时间戳，一个为距离首个数据包的时间差，另一个为距离上个收据包的时间差；4）按照 ASCII 码将子流 ^{k} 转换为矩阵的格式；5）对所得的矩阵进行归一化操作。根据我们的经验，RTP/RTCP 数据包的长度普遍在 50-210 之间，因此我们将矩阵的列设为 256。矩阵的行为 k，即一个子流 ^{k} 所包含的数据包个数。

3.2.3 CLNN 模型研究

基于以上特征分析和构建的子流数据集，本节将介绍本文所使用的 CLNN 模型以及有监督学习的过程。

3.2.3.1 CLNN 模型结构

随着人工智能技术的发展，出现了越来越多优秀的深度学习模型。Lecun, Y. 等人[57] 提出的模型是最古老的 CNN 模型；LeNet 模型作为 CNN 模型的发展被 Rumelhart, David E[58] 等人提出；Krizhevsky, Alex[59] 等人提出的 AlexNet 以作者的名字命名，它在 2012 年举办的 ImageNet 竞赛中获得了冠军；Simonyan, Karen[60] 等人随后提出了 VGG 模型；Szegedy, Christian[61] 等人提出的 GoogleNet 模型在网络设计上做出了大胆的尝试，也获得了更好的性能；He, Kaiming[62] 等人设计了一个具有 152 层的被称作 ResNet 的网络模型。

我们研究了以上神经网络模型，同时吸收 CLDNN 模型[63] 的设计思想设计了本文所使用的 8 层 CLNN 模型。正如大多数的深度学习模型，本文所使用的 CLNN 模型的大多数想法来自于 AlexNet。唯一的不同就是我们将 LSTM 模型加入到卷积神经网络模型中以为语音流提供更强的在时序方面的建模能力。CLNN 模型由 2 层 LSTM，3 层卷积层和 3 层全连接层构成。

在我们的实验中，输入由大小为 k×l 的矩阵组成，k 代表子流 ^{k} 中包含的数据包个数，l 代表矩阵的列数，本文所采取的默认值为 256。我们将 CNN 层的过滤核大小调整为 5×5，调整每层的填充类型为 SAME。结构中前 7 层的 DropOut 值都设置为 0.5。3 层全连接层的前两层使用 ReLU 作为激活函数，最后一层使用 Softmax 作为激活函数。全部 LSTM 层使用 Tanh 作为其激活函数，全部卷积层使用 ReLU 作为激活函数。在我们的 CLNN 模型中，我们去掉了多余的池化层，仅仅在第二层和第五层加入了窗口大小为 2×2 的池化层。

3.2.3.2 CLNN 网络模型的学习过程

对于 CLNN 模型的最后一层全连接层，我们使用 Softmax 作为其激活函数。在我们的实验中，一共有 11 中类型的流量，对于未知的 VoIP 流，我们通过计算其先验概率确定其应用类型。Softmax 函数表示如下：

$$\hat{P}(V_i) = softmax(V_i) = \frac{e^{V_i}}{\sum_{j=1}^{n} e^{V_j}} \tag{3.1}$$

其中 $(V_1,V_2,\cdots,V_i,\cdots,V_n)$ 代表第八层的输入矩阵，$(\hat{P}(V_1),\hat{P}(V_2),\cdots,\hat{P}(V_i),\cdots,\hat{P}(V_n))$ 代表第八层的输出结果，它代表一个未知子流 ^{k} 对于 11 中类型流量的先验概率分布，它是一个大小为 $1\times n$ 的向量。在我们的实验中，n 为 11，意味着我们的 CLNN 模型可以识别 11 中类型的流量。

在学习的过程中，我们的目标是找出最优的权值以获得最大的似然估计。寻求最大似然估计就意味着需要将多分类的对数损失函数最小化，计算损失的函数如下：

$$CCE(V_i) = -\sum_{i=1}^{n} P(V_i)\times\log(\hat{P}(V_i)) \tag{3.2}$$

其中 $P(V_i)$ 代表 V_i 的真实概率，它是通过训练数据集中所设置的标签所得的目标矩阵。如果 j 是该子流 ^{k} 所对应的应用类型，则 $P(V_j)=1$。另外的，当 $i\neq j$ 时，$P(V_i)=0$。

在本文中，我们使用随机梯度下降（Stochastic Gradient Descent，SGD）优化器来最小化损失函数，并且在每个迭代的过程中使用牛顿动量（Nesterov Momentum）来更新梯度值。梯度增量可通过如下公式计算：

$$\Delta X_t = \tau M_{t-1} - \eta\nabla f(X_{t-1}+\tau M_{t-1}) \tag{3.3}$$

其中，τ 用于表示动量因子，η 用于表示学习率。$g_t=\nabla f(X_{t-1}+\tau M_{t-1})$ 用于计算过度点的梯度，ΔX_t 表示实际下降位移，X_t 表示 t 时刻的位置，M_t 表示 t 时刻的动量。

在我们的实验中，学习率会在每轮计算后进行衰变，衰变规则如下：

$$\eta_i = \eta_{i-1}\times\frac{1}{1+\rho\times i} \tag{3.4}$$

其中，ρ 表示衰减因子，i 表示当前正在进行第 i 轮计算。

3.2.3.3 特征集

获得更加精确的特征集用于识别 VoIP 流量是线下训练的目标。在我们的实验中，通过 CLNN 模型所提取的针对语音识别的特征集是不可读的。该特征集将会被用于使用 SVM，random forest，decision tree 和 naive bayes 等机器学习方法训练分类器。经过 4 中机器学习方法所训练的分类器均获得了较高的准确率，并且它们将会被用于我们的在线识别阶段。

第三节 在线实时识别方法研究

本节将会介绍如何使用训练所得的分类器在大规模网络中进行 VoIP 流量的实时识别。本文包括 3 个主要组件用于构建我们的实时识别系统，它们分别是流量捕获器、VoIP 流过滤器和分类器。流量捕获器用于捕获 UDP 数据包并按照 IP 地址和 UDP 端口号进行分流；VoIP 流过滤器用于过滤掉明显的非 VoIP 流以提高系统的实时性；分类器是通过我们提取的特征集经过 SVM 方法训练所得，用于识别 VoIP 流的应用类型。

3.3.1 流量捕获器

流量捕获器维护一个待识别流数据库并且具有将捕获的数据包按照 IP 地址和 UDP 端口号分发到对应的待识别流的能力。图片3.2展示了流量捕获器如何在真实网络中根据 IP 地址和 UDP 端口号分流的过程。如果刚到的数据包不能在数据库中找到对应的待识别流，流量捕获器则会使用该数据包的 IP 地址和 UDP 端口号作为键值创建一个待识别流；反之，如何可以找到对应的待识别流，则将该数据包处理后加入到该待识别流。当一个待识别流的数量达到 k 后，这个待识别流将会被送到下一个 VoIP 流过滤器组件中。

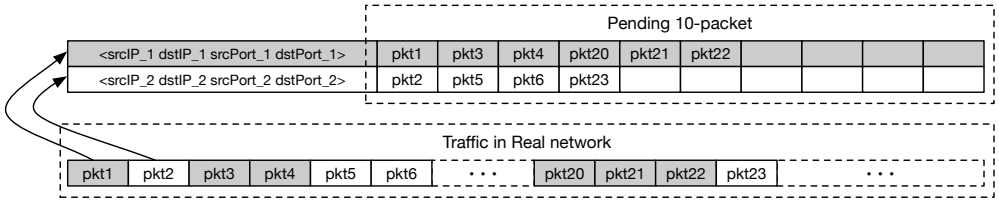

图 3.2 根据 IP 地址和 UDP 端口号进行分流

流量捕获器保证所捕获的数据包均为 UDP 数据包，并且它可以移除待识别流队列中的死流，死流是在有限时间内未达到 k 值的待识别流。

3.3.2 VoIP 流过滤器

VoIP 流过滤器作为实时识别系统的第二个重要组件，它是一个基于规则的过滤器。它接收由流量捕获器发送过来的待识别流，并对该待识别流作出初步的判断。如果初步判断的结果表明该待识别流为非 VoIP 流，因为非 VoIP 流对于我们没有任何意义，我们将不继续对该流做任何处理；如果该待识别流通过了我们设定的全部规则，则该待识别流将会被发送到分类器组件进行进一步识

别。

我们从数据集中选择了一系列基准数据，并根据我们长期研究 VoIP 流量的经验设定了几个阈值参数用于过滤非 VoIP 流量。相关算法和参数分别在算法3.1和表格3.2中展示。

表 3.2 算法3.1中使用的基准值和阈值参数

	符号	详细描述
1.	min_mpl	全部子流中的最小平均数据包长度
2.	max_mpl	全部子流中的最大平均数据包长度
3.	min_pl	最小数据包长度
4.	max_pl	最大数据包长度
5.	min_miat	全部子流中的最小平均包到达间隔
6.	max_miat	全部子流中的最大平均包到达间隔
7.	min_iat	最小包到达间隔
8.	max_iat	最大包到达间隔
9.	λ_1, λ_2	阈值分别为上限和下限
10.	ρ	比例阈值

3.3.3 分类器

通过 VoIP 流过滤器的待识别流将会被输入第三个组件分类器中，分类器所使用的特征集是在离线训练过程中经过训练所得。我们使用了 4 中机器学习的方法重新训练了分类器，我们将具有最好的性能的 SVM 分类器应用于实时识别系统。分类器将 k 维矩阵作为输入并输出一个大小为 $1 \times n$ 的向量，该向量代表此待识别流可能属于每种 VoIP 应用的概率，最大的概率所对应的 VoIP 类型将作为识别结果写入数据库。与此流具有相同键值的 VoIP 数据包将通过查询数据库获得识别结果，供网络管理员进行下一步操作。在识别过程中，一个 VoIP 流会在数据包个数达到不同 k 值的时候被送到对应的分类器，这些分类器都会给出它们的识别结果和它们判定该流属于该识别结果的概率。

第四节 本章小结

本章首先从基本网络特征和统计特征两个角度分析了 VoIP 流量所携带的可供识别的特征，证明了使用媒体传输阶段所产生流量进行 VoIP 流量识别的可行性。之后，本章分析了进行实时识别的基本条件，并介绍了针对 VoIP 实时识别构建其专用的子流数据集和数据预处理的基本方法。基于以上研究，本文

算法 3.1: 用于识别 VoIP/非 VoIP 流的算法

```
Input: 子流 k
Output: VoIP/非 VoIP
if λ1min_mpl < 子流 k 的平均数据包长度 < λ2max_mpl and
 λ1min_miat < 子流 k 的平均包到达间隔 < λ2max_miat then
    for 数据包 in 子流 k do
        if (数据包长度 < λ1min_pl ) then
            ct_pl_l ← ct_pl_l + 1;
        if (数据包长度 > λ2max_pl ) then
            ct_pl_u ← ct_pl_u + 1;
        if (包到达间隔 < λ1min_iat ) then
            ct_iat_l ← ct_iat_l + 1;
        if (包到达间隔 > λ2max_iat ) then
            ct_iat_u ← ct_iat_u + 1;
    if ct_pl_l < ρk and ct_pl_u < ρk and ct_iat_l < ρk and ct_iat_u
     < ρk then
        return VoIP;
    else
        return 非 VoIP;
else
    return 非 VoIP;
```

针对 VoIP 语音流量设计了在频域和时域两个方面具有良好建模能力的 CLNN 模型，该模型可以自动提取 VoIP 流量的抽象特征，在 VoIP 识别方面具有良好的识别性能。本章最后介绍了在大规模网络中进行实时识别的方法，介绍了实时识别系统的三个重要组件及其对应特性。

第四章 VoIP 流量实时识别系统实现

本章根据第三章提出的基于 CLNN 模型进行 VoIP 流量识别的方法，设计并实现了一个可在大规模网络环境中进行 VoIP 流量识别的实时系统。本章首先对 VoIP 实时识别的整体框架做了介绍，之后对系统的离线训练和在线识别过程分别做了详细介绍。

第一节 框架设计

图片4.1展示了本文 VoIP 流量实时识别系统的总体框架。图中展示了两个阶段的具体步骤，离线训练阶段所获的的分类器作为实时识别阶段的第三个组件分类器，在实时识别阶段可以对 VoIP 流作出准确识别。

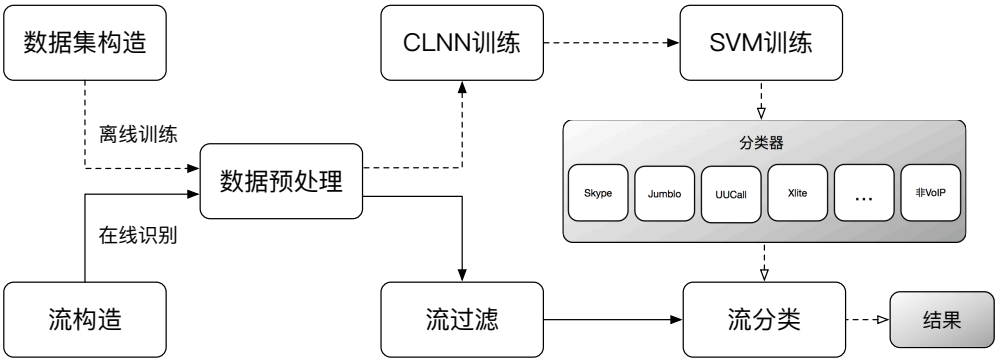

图 4.1 VoIP 流量实时识别框架

4.1.1 离线训练阶段

离线训练阶段包括数据集的构造、数据预处理、CLNN 模型训练和 SVM 训练等四个过程，其目的是通过深度学习提取 VoIP 子流的特征集用于训练适于实时识别的分类器。上文我们提到，构造数据集的过程是会将我们采集的原始数据按照应用类型进行标记并且将 VoIP 完整流按照不同的 k 值进行分流以适用于实时识别；之后，预处理过程会对数据集及其所对的标签进行预处理工作，数据集中的 VoIP 子流将会被处理成可输入 CLNN 模型的矩阵，其对应的标签将会被作为目标矩阵一同输入到 CLNN 模型进行训练；CLNN 模型接收数据集

执行训练过程，经过不断的迭代过程，得到具有最小损失的模型，通过该模型提取出的特征集将会用于 SVM 训练分类器；使用 SVM 训练所得的分类器的准确率没有 CLNN 模型高，但是其具有更良好的实时性。如图4.1中所示，分类器可以识别出各类型的 VoIP 应用流量，同时也可对非 VoIP 的媒体传输流进行识别。

4.1.2 在线识别阶段

在线识别阶段包括流构造、数据预处理、流过滤和流分类几个过程。章节第三节中我们提到的 3 个用于实时识别的组件分别用于执行流构造、流过滤和流分类几个过程。图片4.2展示了流量捕获器、VoIP 流过滤器和分类器工作的基本流程。其中待识别子流中键值为 key_3 的子流由于数据包数量达到了 k，它将会被送往 VoIP 流过滤器执行过滤操作。通过过滤器后将会由分类器作出识别结果存入数据库。图中展示了两个被流量捕获器捕获的数据包其键值分别为 key_1 和 key_2，由于 key_2 数据包未在数据库中查到其对应的识别结果，它将会被用于构造待识别子流等待识别。而 key_1 数据包在数据库中查到其识别结果为 Skype，该数据包会被判断为 Skype 数据包。

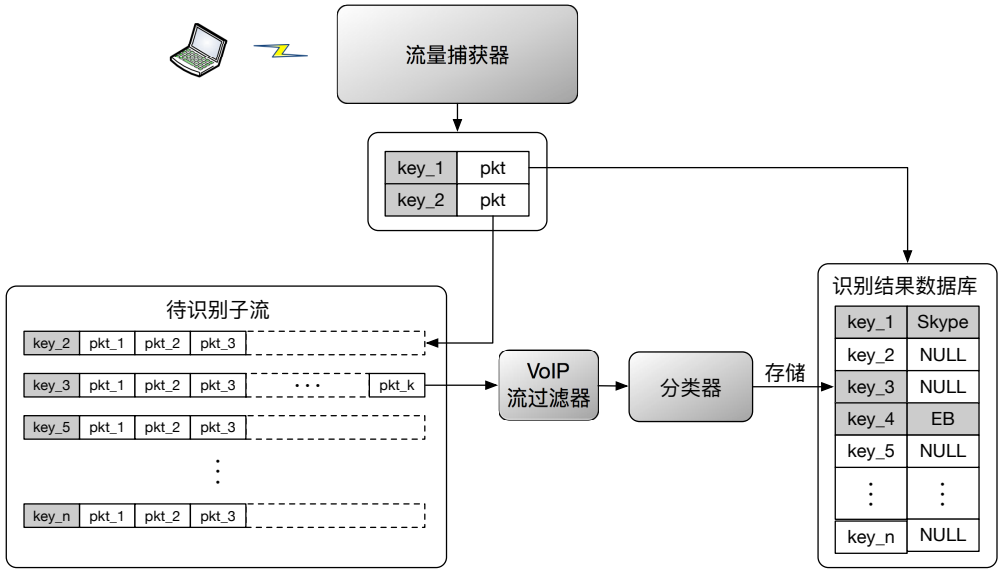

图 4.2 流量捕获器、VoIP 流过滤器和分类器工作流程示意图

第二节 数据集构造

深度学习是在海量数据的环境中提取有价值的信息，足够海量的数据是深度学习的基础。构造带标签的数据集对于研究人员来说是一项繁重的任务，另外由于 VoIP 流量的隐私性，国内外没有公开的海量数据集供我们使用。本文所涉及的落地应用主要针对于可以直拨 PSTN 的 VoIP 应用，因此采集了 10 种流行的 VoIP 应用流量和其他非 VoIP 流量用于训练深度学习模型。

4.2.1 原始数据收集

为了保证所捕获的 VoIP 原始流量的多样性，避免因流量过于单一造成所提取的特征集不具备足够的泛化能力，本文使用多台计算机在多种操作系统下进行流量的收集工作，并使用基于线程的流量捕获工具进行流量标记工作。

为了获得较为纯净的数据集，本文在原始数据收集阶段使用基于线程的方法进行捕获工作。在 Windows 操作系统下，我们使用 QPA 进行流量捕获。QPA 是一款开源的抓包、提供正则表达式识别引擎、按流量类型自动归类、能实时快速的分析软件，我们主要使用其基于进程的工作方式进行流量采集工作。由于 QPA 未在 Linux 环境下提供相应软件，我们使用 tcpdump 工具执行流量捕获工作。当 VoIP 应用开始通话后，我们首先分析其使用的端口号，针对该端口号使用命令 “tcpdump -i eth_name trans_protocol port port_id” 进行流量捕获。基于线程和端口的方法使得所捕获的流量可以很容易的按照应用类型进行标记用于深度学习。来自视频应用和游戏的流量都将被标记为非 VoIP 流量。

4.2.2 数据预处理

捕获的原始数据是以 pcap 文件的格式存储的，每个 pcap 文件存储的内容为一次 VoIP 通话所生成的流量。数据预处理就是要将原始 pcap 文件处理成可供 CLNN 模型学习的训练数据，一是要按照章节3.2.2中介绍的方法进行子流的划分用于提取适于实时识别的特征集，二是要将子流处理成可输入 CLNN 模型的矩阵格式。

为了采取多个数据集适应不同程度的实时识别，本文按照 k 值的不同训练了多个 CLNN 模型。k 值包括 2，4，6，8，10，20，40 和 100，以上多个 k 值所训练出的模型具有不同程度的实时性。其识别准确率与 k 值成正比，实时性与 k 值成反比。图片4.3展示了以 pcap 文件作为输入预处理后以子流作为输出的过程。输出的子流以文本文件的方式进行保存。

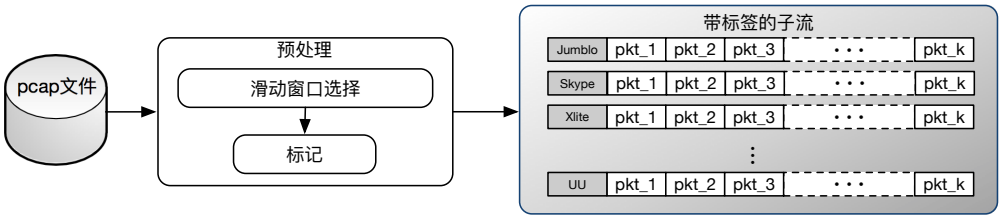

图 4.3 数据预处理详情

按照我们所提出的将 VoIP 子流转化成矩阵的方法，图片4.4展示了一个用于训练的 Skype 子流 ^{100} 图片格式的表现形式。

图 4.4 Skype 子流 ^{100} 的图片表现形式

第三节 CLNN 模型离线训练

基于 CLNN 模型的离线训练过程是进行 VoIP 流量识别的核心。训练过程中，CLNN 模型可以通过对训练数据的学习，将低维层次的特征转化成高维层次的特征，从而构造出用于 VoIP 流量识别的特征集。CLNN 模型通过训练数据集自学习过程中，会在正向传播的过程中计算损失，并通过随机梯度下降的方式逐层向前反馈进行反向传播，从而最小化损失。

本文使用 Keras 库搭建 CLNN 模型，Keras 是一个基于 Theano 和 TensorFlow 的深度学习库，其具有高度模块化和可扩充的特性，并且可以支持 CPU 和 GPU 的无缝切换。Keras 高度的支持 CNN 和 RNN 的结合，本文使用的 CNN 和 LSTM 的策略可以很简单的使用 Keras 实现。Keras 提供的函数式模型是用户定义多输出模型、非循环有向模型或具有共享层的模型等复杂模型的途径，本文使用函数式模型进行了 CNN 和 LSTM 的结合，从而创建了

具备更强的提取 VoIP 流量特征能力的模型。

CLNN 模型经过不断的迭代学习后，输出结果就是用于 VoIP 子流识别的一系列权重值。CLNN 模型使用这些权重值进行正向计算从而可以计算出 VoIP 子流的识别结果。

4.3.1 模型基本结构

CLNN 模型在学习过程中会生成大量的参数，大量的参数也是 CLNN 模型准确识别的基础。本节将以 k=100 对 CLNN 模型每层所使用的具体参数进行详细介绍。之后列出本文训练的 8 种模型生成的权值参数个数以及保存权值的 HDF5 文件大小。

表 4.1 CLNN 模型参数详情

<table>
<tr><th colspan="3">$CNN + Dense$</th><th>LSTM</th></tr>
<tr><td rowspan="3">Conv1</td><td>输入</td><td>$100 \times 256 \times 1$</td><td rowspan="24">输入：100×256
单元：100
权值个数：142800
Dropout：0.5</td></tr>
<tr><td>参数</td><td>卷积核：$5 \times 5 \times 48$；步长：2</td></tr>
<tr><td>输出</td><td>权值个数：1248；Dropout：0.5</td></tr>
<tr><td rowspan="3">Conv2</td><td>输入</td><td>$50 \times 128 \times 48$</td></tr>
<tr><td>参数</td><td>卷积核：$3 \times 3 \times 128$；步长：2</td></tr>
<tr><td>输出</td><td>权值个数：55424；Dropout：0.5</td></tr>
<tr><td rowspan="3">Conv3</td><td>输入</td><td>$25 \times 64 \times 128$</td></tr>
<tr><td>参数</td><td>卷积核：$3 \times 3 \times 192$；步长：1</td></tr>
<tr><td>输出</td><td>权值个数：221376；Dropout：0.5</td></tr>
<tr><td rowspan="3">Conv4</td><td>输入</td><td>$25 \times 64 \times 192$</td></tr>
<tr><td>参数</td><td>卷积核：$3 \times 3 \times 192$；步长：1</td></tr>
<tr><td>输出</td><td>权值个数：331968；Dropout：0.5</td></tr>
<tr><td rowspan="3">Conv5</td><td>输入</td><td>$25 \times 64 \times 192$</td></tr>
<tr><td>参数</td><td>卷积核：$3 \times 3 \times 128$；步长：1</td></tr>
<tr><td>输出</td><td>权值个数：221312；Dropout：0.5</td></tr>
<tr><td rowspan="3">Dense1</td><td>输入</td><td>204800</td></tr>
<tr><td>参数</td><td>单元：2048</td></tr>
<tr><td>输出</td><td>权值个数：419432448；Dropout：0.5</td></tr>
<tr><td rowspan="3">Dense2</td><td>输入</td><td>2048</td></tr>
<tr><td>参数</td><td>单元：100</td></tr>
<tr><td>输出</td><td>权值个数：204900；Dropout：0.5</td></tr>
<tr><td rowspan="3">Dense3</td><td>输入</td><td colspan="2">200</td></tr>
<tr><td>参数</td><td colspan="2">单元：11</td></tr>
<tr><td>输出</td><td colspan="2">权值个数：2211</td></tr>
</table>

表格4.1展示了 CLNN 模型每层所采用的参数详情。表格所示由两部分组

成，第一部分是由 CNN 和 Dense 组成的用于提取频域特征的模型，第二部分是 LSTM 层用于提取时域特征。两部分在 Dense3 也就最后一个全连接层进行组合，Dense3 将频域特征和时域特征进行组合最大患的提取 VoIP 流量携带的特征。以第一层 Conv 举例，输入为 $100\times256\times1$ 的矩阵，卷积核大小为 5×5，深度为 48。通过 $W=(r_{filter}\times c_{filter}\times d_{input}+b)\times d_{filter}$ 可以计算该层权值个数为 1248，其中每个神经元包含 $5\times5\times1$ 权值和一个偏差。以第一层 Dense 层举例，输入为 204800 维的向量，输出为 2048 维的向量，通过 $W=(dim_{input}+b)\times dim_{output}$ 可以计算其权值个数为 419432448。LSTM 层输入为 100×256 的矩阵，输出单元为 100，可以通过 $W=4\times dim_{output}+r_{input}+1\times dim_{output}$ 计算其权值个数，其中 4 代表每个 LSTM 单元包括 4 个激活函数。

4.3.2 Keras **搭建模型**

本文使用 Keras 提供的 Model 类搭建函数式模型，需要导入 Keras 提供的 models 包。使用

model=Model(input=subflows,output=[model])

创建模型。之后，向该模型中添加隐层，本文主要使用了 3 种隐层，卷积层、全连接层和 LSTM 层。依次可以使用

model = Conv2D(filters, kernel_size, strides, activation)(model)

model = Dense(units, activation)(model)

model = LSTM(units, activation)(model)

添加到 CLNN 模型中。另外，本文选择性使用了池化层（MaxPooling2D）和失活层（Dropout）加速训练的过程。在维度转换时，本文使用了平滑层（Flatten）和变形层（Reshape）提供的支持。

本文搭建的 CLNN 模型最主要的特性是针对频域特征和时域特征同时进行提取，最后使用全连接层综合两个类型的特征。以下代码段展示了如何将 CNN 和 LSTM 提取的特征进行组合。

```
input = Input(input_shape)
conv = create_cnn_layers(input)
lstm = create_lstm_layers(input, input_shape[0])
model = concatenate([conv, lstm],axis=1)
model = Dense(nb_class, activation='softmax')(model)
```

4.3.3 迭代训练

CLNN 模型迭代训练是通过训练数据集不断调整权值寻找具有最小误差的权值的过程。

本文使用随机梯度下降的算法进行权值的更新，同时在随机梯度下降时引入冲量，冲量用于表示上一梯度对当前梯度的贡献率。梯度下降算法采用小批量更新的策略，每一批样本数据计算一次损失函数并更新参数。随机梯度下降优化器可通过 SGD 类进行定义。

sgd = SGD(lr=0.0, decay=0.0, momentum=0.9, nesterov=True)

为 CLNN 模型配置损失函数和优化器类型通过 compile 函数完成，

model.compile(loss='categorical_crossentropy', optimizer=sgd)

以上配置所使用的损失函数为针对多类的对数损失函数。

上文中定义随机梯度下降函数时没有为 SGD 指定学习率，因为本文采取每轮训练完成后都要对学习率进行更新的策略。以下代码段为对上一章中介绍的公式3.4的具体实现。

```
def step_decay(epoch):
    initial_lrate = 0.01
    drop = 0.5
    epochs_drop = 5.0
    lrate = initial_lrate *
        math.pow(drop,math.floor((1+epoch)/epochs_drop))
    print(lrate)
    return lrate
lrate = LearningRateScheduler(step_decay)
```

训练中使用 EarlyStopping 优化训练行为，当损失不再减小时则提前停止训练过程。

earlystopping = EarlyStopping(monitor='val_loss', patience=1)

我们在训练中对全部数据指定进行 20 轮的计算，当然，在满足 EarlyStopping 的条件后训练也会提前结束。每 200 个样本计算后反向传播进行梯度的更新。以下函数最后的 callbacks 参数展示对学习率更新以及 EarlyStopping 的调用。

表 4.2 8 个 CLNN 模型生成的权值个数及 HDF5 文件大小详情

CLNN-k	**输入**	**权值个数**	HDF5(MB)
k=2	$2\times256\times1$	17,818,697	87
k=4	$4\times256\times1$	17,820,823	87
k=6	$6\times256\times1$	34,600,197	154
k=8	$8\times256\times1$	34,602,387	154
k=10	$10\times256\times1$	51,381,825	221.6
k=20	$20\times256\times1$	84,947,847	355.8
k=40	$40\times256\times1$	168,859,507	691.3
k=100	$100\times256\times1$	420,613,687	1619.3

model.fit(X_train, Y_train, batch_size=200,validation_data=(X_val, Y_val),epochs=20,callbacks=[early_stopping,lrate])

4.3.4 模型保存

经过训练得到的最终模型需要进行存储，模型分为两个部分进行存储。一是需要对 CLNN 模型的基础架构进行存储，我们使用 json 格式的文件进行存储；二是要对训练后的模型的权重参数进行保存，我们使用可提供层次数据格式存储的 HDF5 进行存储。在实时识别阶段我们将使用该模型进行 VoIP 流量的识别。表格4.2展示了本文训练所得的 8 个 CLNN 模型生成的权值个数及存储为 HDF5 文件得大小详情。

第四节 VoIP 流量实时识别系统

VoIP 流量实时识别系统是部署在大规模真实网络出入口节点上的用于在线识别 VoIP 流量的分布式系统。为了应对大规模真实网络上复杂多变的环境，我们采用 Apache Kafka 流处理平台接入真实网络环境。Kafka 流处理平台使用 ZooKeeper 作为其后端分布式服务器管理、协调 Kafka 代理，可以为处理实时数据提供高吞吐、低延迟的性能，解决了真实网络中流速不均衡造成的难题。其次，Kafka 除了为我们提供负载均衡的环境之外，也作为我们的数据备份模块备份网络流量。为了达到实时计算的目的，我们采用 Apache Storm 作为流式计算框架执行计算任务。用户通过自定义 Spout 和 Bolt 指定信息源和计算措施从而达到批量、分布式处理流式数据的目的。本节我们将分为 Kafka 和 Storm 两个模块进行介绍。

4.4.1 Kafka 模块

Kafka 模块是用于直接对接真实网络环境的基础模块，其可以按照不同交换机和路由器创建标题，供 Storm 模块进行消费。

图片4.5展示了真实网络如何接入 Kafka 集群的结构图。本文实验中将局域网中的路由器接入 Kafka 集群模仿真实网络环境中的交换机。

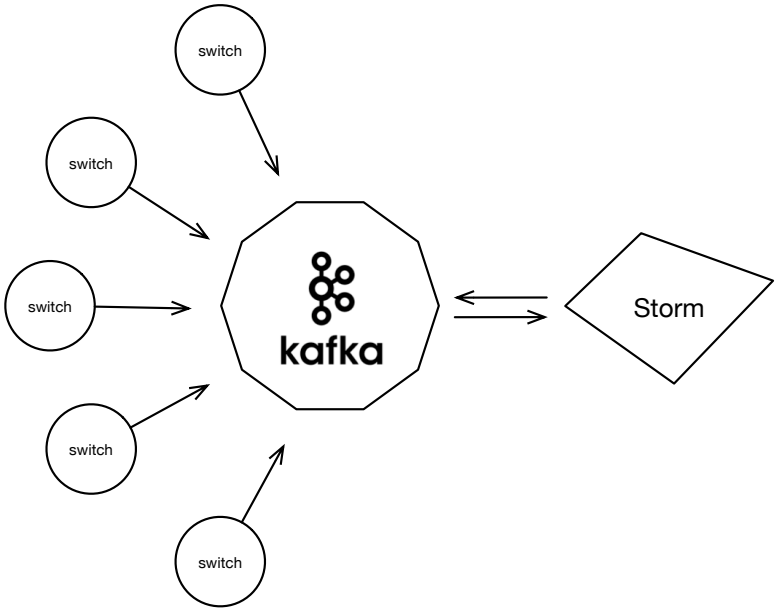

图 4.5 真实网络环境接入 Kafka 集群结构图

Kafka 提供良好的 Java 编程接口，我们主要使用其提供的生产者接口将真实网络中的流量发送到 Kafka 集群中。以下分别为发送单条数据和发送多条数据所对应的接口。

public void send(KeyedMessage<K,V> message)

public void send(List< KeyedMessage<K,V> > messages)

4.4.2 Storm 模块

Storm 模块是支持实时识别的核心模块，其对来自 Kafka 集群的动态的、连续的以及大数据量的流量作出及时的处理。

本文使用 Kafka Spout 作为消费者客户端消费 Kafka 集群中未处理的流量，同时 Kafka Spout 也作为 Storm 拓扑结构的数据源将流量吐到 Storm 拓扑中进

行处理。其次，我们使用两个 Bolt 分别作为流量捕获器的 UDP 识别和分流部分，后继 Bolt 可能是一个 VoIP 流过滤器也可能直接是一个分类器，这根据不同的 k 值作出不同调整。具有较小 k 值的 VoIP 流不易进行初步 VoIP 流过滤操作，其次较小的 k 本身要求要高的实时性，因此不经过 VoIP 流过滤器直接送入分类器进行识别。最后一个 Bolt 提供写入 Redis 数据库的操作。图片4.6展示了 Storm 模块的拓扑结构图。其中分类器 Bolt 对应多种 CLNN 模型所构成的分类器，它们产生的结果和该结果所对应的可能性都将输出到 Redis 数据库中，网络管理员可以根据实时性和准确度要求拿取符合要求的结果。

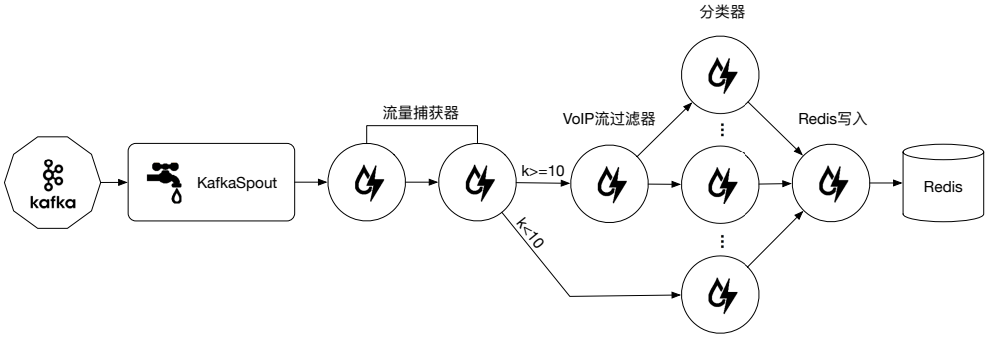

图 4.6 Storm 模块拓扑结构图

以下代码段展示了 Storm 拓扑的搭建过程。

```
TopologyBuilder tb = new TopologyBuilder();
tb.setSpout("spout", new TrafficSpout(), 3);  // executor num
tb.setBolt("udpbolt", new UDPBolt()).shuffleGrouping("spout");
tb.setBolt("pendingflowsbolt", new PendingFlowsBolt())
        .fieldsGrouping("udpbolt", new Fields("key"));
tb.setBolt("classifierbolt10", new ClassifierBolt10())
        .shuffleGrouping("pendingflowsbolt", "10");
tb.setBolt("filterbolt", new FilterBolt())
        .shuffleGrouping("pendingflowsbolt", "100");
tb.setBolt("classifierbolt100", new ClassifierBolt())
        .shuffleGrouping("filterbolt");
tb.setBolt("rediswriterbolt", new RedisWriterBolt())
        .shuffleGrouping("classifierbolt10")
```

```
.shuffleGrouping("classifierbolt100");
```

第五节 本章小结

本章依托第三章设计的移动应用流量识别算法，实现了一个移动应用流量识别系统。本章第一节首先系统的介绍整体框架，分为离线训练阶段和在线识别阶段做了简单介绍；第二节对本文采用的构建数据集的方法和数据集详细信息做了介绍；第三节对本文所使用的 CLNN 模型的结构做了详细介绍，并从编程的角度介绍了模型的搭建、训练和输出的过程；最后第四节介绍了为支撑大规模网络环境中的识别，我们采用的 Kafka 和 Storm 框架，并使用这两种框架开发了适用于本文所提出的方法的应用环境。

第五章 实验结果和性能评估

本章将从 CLNN 模型准确率和 VoIP 识别系统的实时性两个方面进行评估。针对本文所采取的 VoIP 流量，使用本文所搭建的 CLNN 模型进行训练和验证，并分析了 8 个不同 k 值训练出的 CLNN 模型的识别准确率。同时，为证明本文所提取的特征集的有效性，我们使用 4 种机器学习的方法通过该特征集训练数据集并进行了对比。最后，从实时性的角度分析实时系统的可用性。

第一节 实验基础

5.1.1 实验环境

实验环境包括开发环境和运行环境两个部分，开发环境用于训练 CLNN 模型和实时系统开发，运行环境用于搭建分布式服务器、运行 Kakfa 和 Storm 集群、存储识别结果和回放 VoIP 流量。

表格5.1展示了本文开发环境和软件。本文深度学习方面的内容基于 Python 开发，使用的深度学习框架为 Keras。本文使用 GPU 环境加速深度学习过程，使用 CUDA 作为其运算平台。分布式集群相关使用 Java 进行开发。

表 5.1 开发环境

	项目	具体环境及版本
硬件环境	CPU	四核 3.6GHz Intel Core i7
	GPU	NVIDIA GeForce GTX 1070
	RAM	8GB
软件环境	操作系统	CentOS7
	GPU 运算平台	CUDA Toolkit 8.0
	深度框架	Keras
	编程语言	Python 2.7.10, Java 1.8.0

表格5.2展示了本文 VoIP 实时识别系统的运行环境。本文使用 ZooKeeper 提供分布式协调服务，Kafka 和 Storm 作为运行在 ZooKeeper 之上的分布式应用提供存储和计算功能。键值数据库使用 Redis 提供分布式查询功能。本文为模拟真实网络环境的 VoIP 流量，使用了 tcpreplay 软件回放我们采集到的 VoIP

流量。

表 5.2 运行环境

	项目	具体环境及版本
硬件环境	服务器	3 台, RAM 8GB, 磁盘空间 200GB
软件环境	操作系统	CentOS7
	分布式系统	ZooKeeper 3.4.12
	集群服务	Kafka 2.0.0, Storm 1.2.2
	数据库	Redis 5.0.3
	回放软件	tcpreplay

5.1.2 数据集

本文针对 10 种流行的 VoIP 应用和多种非 VoIP 的视频和即使游戏应用进行了流量的捕获工作。为了提升识别的泛化能力，我们部署了 8 台计算机，其中 6 台计算机安装了 Windows 操作系统，2 台计算机安装了 Linux 操作系统，2 台 Linux 计算机安装了提供支持在 Linux 环境下使用的 4 种 VoIP 应用，6 台装载 Windows 操作系统的计算机均安装了 10 种 VoIP 应用。其次我们捕获了视频应用、视频通话应用和即时游戏应用的 UDP 流量作为非 VoIP 应用流量。为了保证 VoIP 通话线路的泛化性，我们从国内包括河北省、吉林省、广东省和四川省多个地方拨打电话用于流量获取工作。

本文将采集到的数据分成两个数据集，数据集 1 用于训练 CLNN 模型，数据集 2 用于进一步验证 CLNN 模型并且用作回放流量验证实时系统的性能。5.3展示了数据集 1 的统计信息和经过处理后用于训练的子流 ^{10} 个数。数据集 2 不做子流处理，其流量大小与数据集 1 相同。训练过程中数据集 1 中的 80% 子流用作训练 CLNN 模型，20% 的子流用于训练过程中验证准确率。

5.1.3 训练参数

训练过程中使用的 SGD 算法是一种学习率非自适应的算法，学习率的设置对实验结果有直接的影响。我们结合深度学习一般经验并按照实验结果进行调整，初始学习率设置为 0.01，衰减规则按照本文以上所述每两轮训练进行一次衰减。nesterov 动量因子设置为 0.9，它会吸收上一梯度 90% 的贡献。我们针对 8 个 k 值分别对数据集训练了 20 轮，每 200 个样本进行一次梯度更新。

表 5.3 数据集 1 详细信息

VoIP **应用类型**	**平台**	**数据集大小** (MB)	**子流数** (k=10)
Skype	Windows, Linux	3908.8	669984
UUCall	Windows	2709.4	691566
KCCall	Windows	3128.8	808224
ALTCall	Windows	2795.2	692002
Jumblo	Windows, Linux	3704.6	871468
Zoiper	Windows, Linux	4418.1	677114
Xlite	Windows, Linux	5165.3	638288
Eyebeam	Windows	4524.7	616773
ExpressTalk	Windows	4633.1	602637
Bria	Windows	4476.0	598724
non-VoIP	Windows	11324.5	973522

第二节 CLNN 模型评估

5.2.1 评估指标

在多分类问题中，不同的场景需要不同的评估指标。一般来说，准确率（Accuracy）就可以评估一个分类模型的性能，用于衡量分类正确的比例。可以使用如下公式计算准确率：

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} 1_{\hat{y}_i = y_i} \tag{5.1}$$

其中，$\hat{y}_i$ 表示被分类模型预测的类别，y_i 表示样本的真实类别。准确率是一个适用范围最广的评估指标，可以反映系统的全局识别性能，但是准确率在反映分类器对正负样本的识别能力上有着不足。本文引入真正类率（True Positive Rate, TPR）来表达分类器所识别出的正实例占所有正实例的比例，负正类率（False Positive Rate, FPR）在表达能力上与 TPR 是互补的关系。对于二分类问题，即将样本分为正类和负类两种情况的问题来说，会出现 4 种分类结果：如果一个实例是正类并且也被预测成正类，即为真正类（True Positive, TP），如果实例是负类被预测成正类，称之为假正类（False Positive, FP）。相应地，如果实例是负类被预测成负类，称之为真负类（True Negative, TN），正类被预测成负类则为假负类（False Negative, FN）。通过这种分类方式，我们重新表达准确率如下：

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

表 5.4 8 个模型对训练数据集和验证数据集的识别率

模型 k	损失	训练集	验证集
k=2	0.015865	0.995560	0.996130
k=4	0.007670	0.997700	0.997860
k=6	0.004933	0.998520	0.999340
k=8	0.004393	0.998690	0.999130
k=10	0.001159	0.999650	0.999650
k=20	0.000546	0.999880	0.999880
k=40	0.002920	0.999120	0.999460
k=100	0.006689	0.997620	0.998730

评估指标 TPR 和 FPR 计算如下：

$$TPR = \frac{TP}{TP+FN} \tag{5.3}$$

$$FPR = \frac{FP}{FP+TN} \tag{5.4}$$

5.2.2 识别性能评估

根据以上 CLNN 模型架构及实验基础中使用的参数，我们对 11 类流量分为 8 个 k 值进行了训练得到了 8 个 CLNN 模型。本节将对这 8 个模型按照第一节中介绍的评估指标进行评估。

表格5.4展示了 8 个模型对训练数据集和验证数据集识别的准确率。训练数据集来自数据集 1 中的 80%，验证数据集来自数据集 1 中的 20%。我们可以看到其识别率均接近于 1。

因为训练过程中普遍存在过拟合的现象，甚至在验证阶段可能不会表现出来。因此我们使用数据集 2 进行全面的验证。图片5.1中的前十个图表展示了使用数据集 2 进行验证的 10 种 VoIP 应用流量所对应的 TPR，后两个图表展示了数据集 2 的 FPR 和 TPR 分布。很明显的，在前 4 张图表中我们可以看到一些异常值，分别是使用模型-20 进行识别 Skype 流量、模型-4 识别 Xlite 流量、模型-6 识别 Eyebeam 流量和模型-2 识别 Bria 流量时，这表明这四种模型在训练时针对这几种流量出现了过拟合现象。我们在实时识别过程的过程中暂时放弃使用这几个模型识别出的结果对应四种流量的结果。在未来的研究过程中，第一步我们需要将数据集 2 合并到数据集 1 中再次训练，第二步我们考虑加入 L1 和 L2 正则层避免过拟合现象。抛除以上几个异常值，综合显示 CLNN

表 5.5 VoIP 实时识别系统识别 100 个流所耗费的时间

Model	k=2	k=4	k=6	k=8	k=10	k=20	k=40	k=100
Time(s)	27.55	20.12	29.09	30.53	48.00	73.62	141.33	309.51

模型有优秀的识别性能。

图片5.2展示了针对两个数据集识别时识别正确和识别错误的结果所对应的概率分布。如图所示识别正确的结果所对应的识别概率普遍接近于 1，而识别错误的结果所对应的概率分布在 0.2-0.98 之间，普遍要低于识别正确的结果所对应的概率。该概率将作为实时识别系统初步判断识别结果的阈值。

第三节 实验对比

5.3.1 特征集对比

由于 VoIP 数据集的特殊性，本文所使用的识别方案无法与已存的识别方案直接进行对比。我们使用本文 CLNN 模型提取的特征集与文章[13] 中的所使用特征集进行对比，分别使用 SVM，随机森林，C5.0 决策树和朴素贝叶斯四种机器学习方法对两种特征集训练了分类器。图片5.3展示了 CLNN 模型提取的特征集与文章[13] 中使用特征集的对比结果。

5.3.2 其他方法对比

第四节 实时性评估

为了验证本文搭建的 VoIP 流量实时识别系统的实时性，本文使用数据集 2 模拟真实网络环境中的流量进行实时性验证。我们使用 tcpreplay 软件回放所收集到的 VoIP 流量，tcpreplay 可以将我们收集到的 pcap 文件进行回放，并且其支持在回放的过程中修改网络层报头。我们通过 tcpreplay 工具在流量纯净的网卡上进行回放工作，我们记录从收到 VoIP 流的第一包到识别出结果所耗费的时间。表格5.5展示了实时识别系统识别 100 个 VoIP 流所耗费的时间。可以看到，当我们使用 k 值为 100 的分类器进行识别平均识别每个流需耗费 3 秒钟，使用 k 值为 2 的分类器耗费 0.3 秒钟。对于不同情形下的识别需求，网络管理员可以按需对识别结果作出应对。

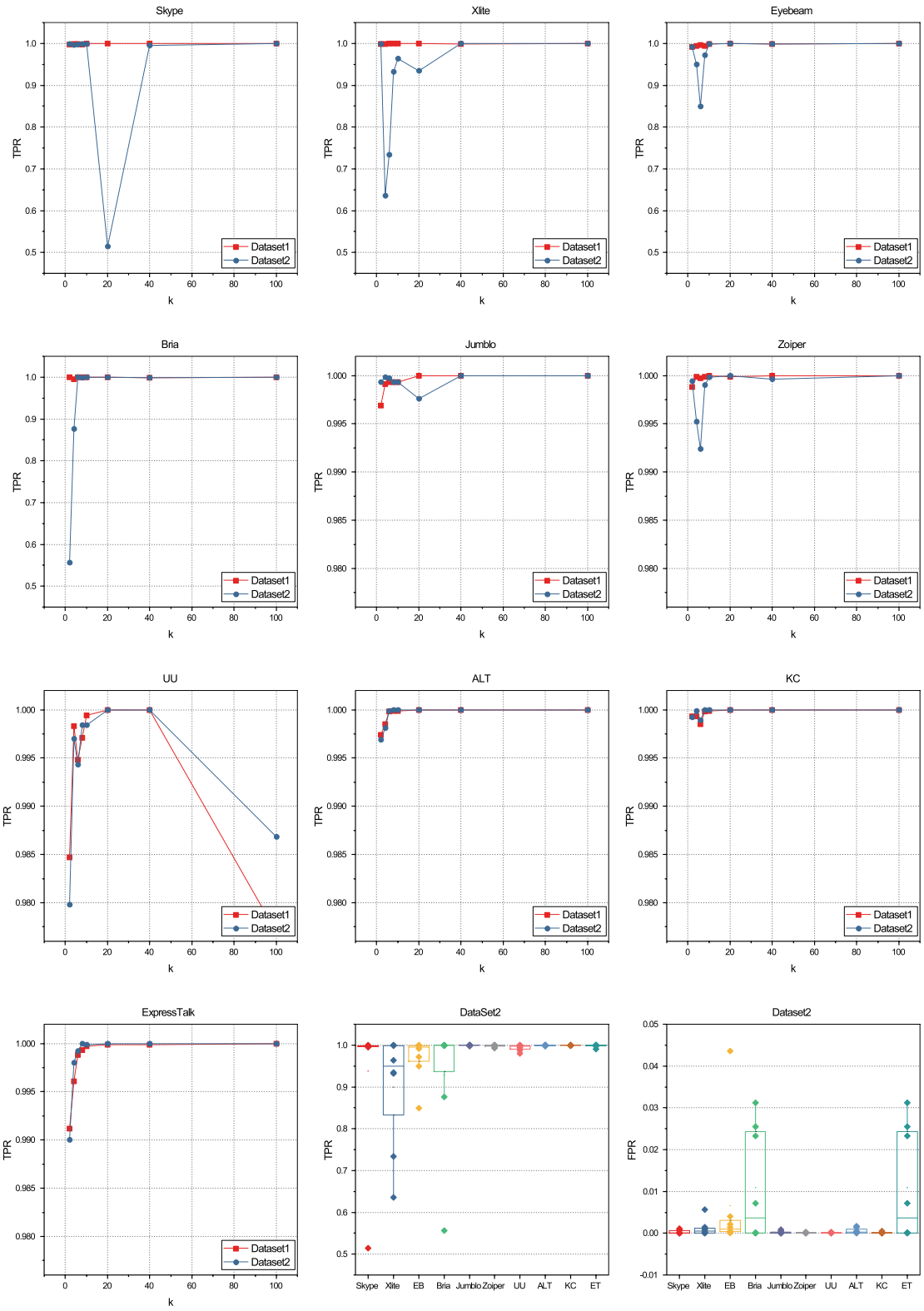

图 5.1 8 种 CLNN 模型的 TPR 和 FPR 结果

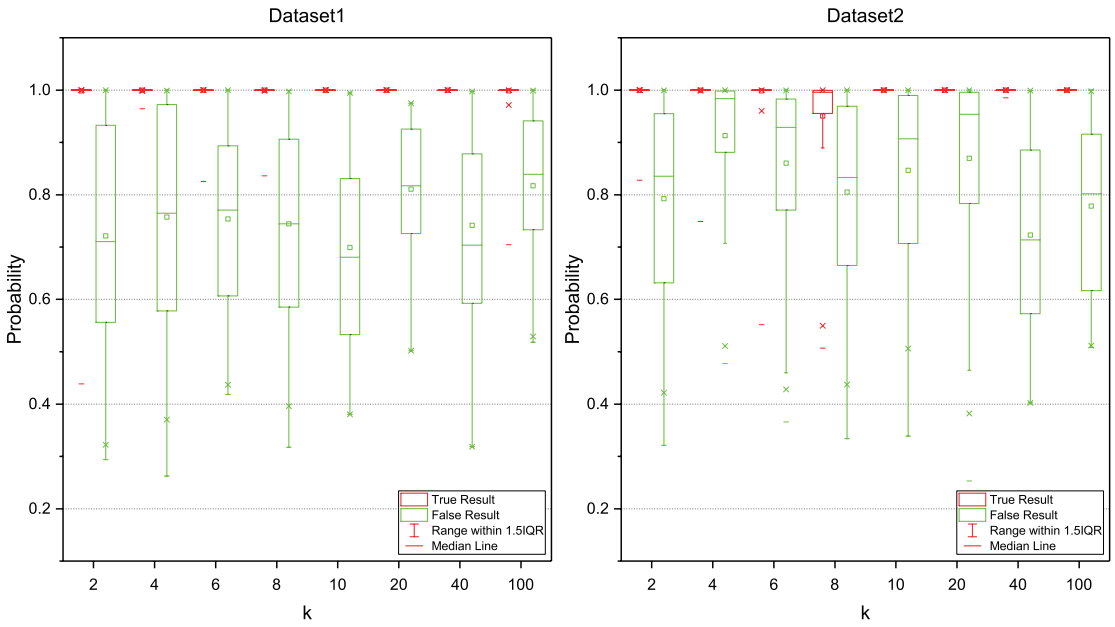

图 5.2 识别结果的概率分布状况

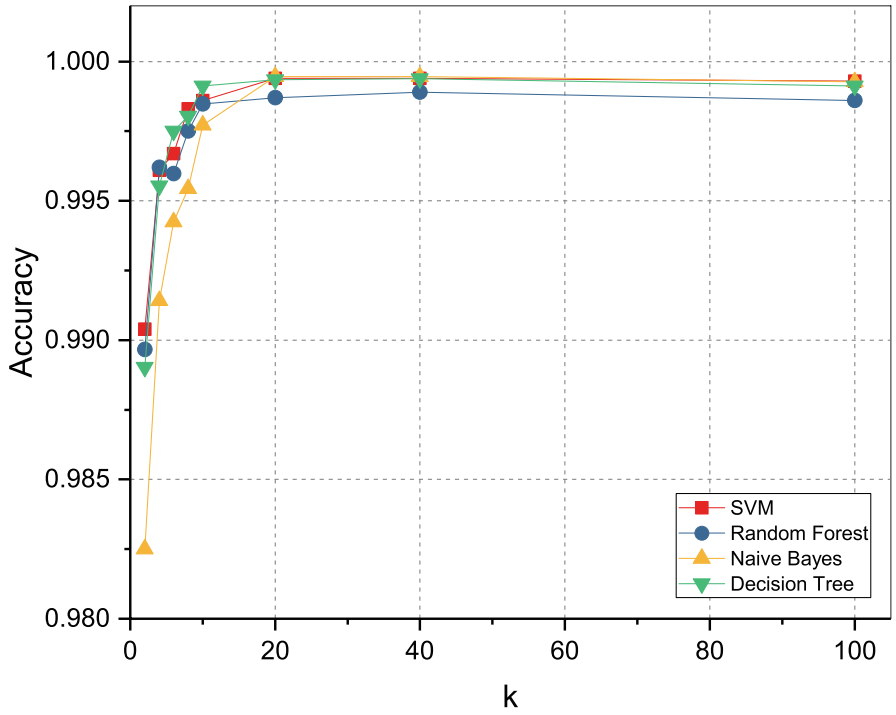

图 5.3 CLNN 模型提取特征集与文章[13] 中特征集对比

第五节 本章小结

本章首先介绍了 CLNN 模型的训练环境和 VoIP 流量实时识别系统的运行环境，之后对训练数据集和深度学习所使用的参数做了详细介绍。然后针对 CLNN 模型的识别性能从不同的层面做了评估，证明了 CLNN 模型的有效性。其次，我们还将本文所提出的方法与其他文章中的识别方法做了比较。最后，本文对 VoIP 流量实时识别系统的实时性作了评估。

第六章 总结与展望

VoIP 应用的广泛使用势必会造成诸多的社会问题，为了使 VoIP 应用是可控的，对于 VoIP 应用流量的有效监管是必要的。由于 VoIP 应用的加密性和 P2P 的特性，VoIP 流量的识别难度要远远高于其他流量的识别。本文总结整理了现存的 VoIP 流量识别方法，并分析了这些方法的优缺点。在此基础上，我们针对 VoIP 应用媒体传输流量进行了分析，设计了可以自动提取媒体传输流量的 CLNN 模型，该模型可以从时域和频域两个维度上最大限度的提取其特征用于 VoIP 识别。其次，本文提出了针对 VoIP 子流的实时识别方法，设计了可以应用于大规模真实网络的 VoIP 流量实时识别系统。同时，我们对于文中使用的各种方法进行了实验验证，证明了方法的可靠性。

第一节 论文总结

本文的主要工作如下：1）本文研究了 VoIP 流量识别的关键技术。对于 VoIP 流量识别方法分为多个层次进行了介绍，分析了每种方法的针对点和适用性范围。

2）本文研究了与 VoIP 流量识别相关的各方面的知识。包括 VoIP 相关技术、流分类、深度学习等多个方面的内容，在此基础上，本文分析了 VoIP 流量所携带的各种可能用于进行识别的特征。之后结合卷积神经网络和长短期记忆网络模型在时域频域方面的建模能力，设计了可以自动提取 VoIP 流量特征的网络模型。本文同时在 VoIP 识别实时性方面作出要求，设计了基于子流进行识别的方法。结合深度学习方面的能力，该模型在保证识别精度的同时也保证了识别要求的实时性。

3）针对本文所提出的方法，我们设计了并实现了一个可以应用于大规模真实网络的 VoIP 流量实时识别系统。该系统采用 Storm 作为其流计算引擎，Kafka 作为其负载均衡系统，保证了在大规模真实网络中的识别性能。在 VoIP 识别系统中引入了 CLNN 模型作为分类器，该分类器保证了对 VoIP 流量的识别精度。其次，我们对实时识别的性能通过加入 VoIP 流过滤器进行提升，从而提高了识别系统的实时性能。

第二节 问题和展望

尽管本文提出的方法在移动流量分类实验中表现出了良好的识别效果，但距离实际应用还有一定的距离，其中还有一些问题需要解决，这也是进一步研究的方向：

1）本文的实验均针对离线数据，并没有对真实网络流量进行实时监测和识别。要移植到真实网络节点中，还需要进一步提高识别速度。进一步简化模型，以及实现高效的并行化方法都是下一步研究方向。

2）本文针对 HTTP 协议进行分类，然而忽略了 HTTP 协议的固有特性。尽管分类模型取得了良好的效果，但其实模型的学习曲线更加曲折。如果能利用 HTTP 的特征，事先提取部分有用信息，再结合卷积神经网络进行分类，这样卷积神经网络的学习更加具有针对性，可能会取得更好的分类效果，也能减少一定训练时间。

3）本文只针对 20 个移动应用进行了实验，实验数据具有一定的局限性。为了验证模型的适用性，采集更为广泛的应用流量也非常关键。而且本文只针对 HTTP 流量进行了识别，为了更全面的识别移动应用流量，如何利用已经识别出的 HTTP 流量进一步识别其他关联流量也值得研究。

参考文献

[1] 2017 Global Fraud Loss SurveyCFCA, https://zh.scribd.com/document/368471387/2017-Global-Fraud-Loss-Survey-CFCA-pdf.

[2] 信息通信行业防范打击通讯信息诈骗白皮书（2018）, http://www.caict.ac.cn/kxyj/qwfb/bps/201807/P020180711616875493434.pdf.

[3] 2017 年中国反通讯网络诈骗报告, https://gw.alicdn.com/bao/uploaded/LB16jHhkhPI8KJjSspfXXcCFXXa.pdf.

[4] Taliban Using Skype Phones to Dodge MI6, The Daily Mail, https://www.dailymail.co.uk/news/article-1055611/Taliban-using-Skype-phones-dodge-MI6.html.

[5] Lashkars own Skype frazzles Indian intelligence, Times of India, https://timesofindia.indiatimes.com/india/Lashkars-own-Skype-frazzles-Indian-intelligence/articleshow/12934037.cms.

[6] Curtis J, Cleary J, McGregor A, et al. Measurement of voice over IP traffic. In: Passive and Active Measurement conference, 2000.

[7] Chen M, Zhang G-X and Bi J-P. Research of SIP-based VoIP traffic identification methodology. Appl. Res. Comp, 2007, 24.

[8] Sen S, Spatscheck O and Wang D. Accurate, scalable in-network identification of p2p traffic using application signatures. In: Proceedings of the 13th international conference on World Wide Web, 2004: 512 $\sim$ 521.

[9] Idrees F, Aslam U. A generic technique for Voice over Internet Protocol (VoIP) traffic detection. 2008.

[10] Li B, Jin Z and Ma M. VoIP traffic identification based on host and flow behavior analysis. In: Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on, 2010: 1 $\sim$ 4.

[11] Rathore M, Paul A, Ahmad A, et al. High-speed network traffic analysis: detecting VoIP calls in secure big data streaming. In: 2016 IEEE 41st Conference on Local Computer Networks (LCN), 2016: 595 $\sim$ 598.

[12] Saqib N A, SHAKEEL Y, Khan M A, et al. An effective empirical approach to VoIP traffic classification. Turkish Journal of Electrical Engineering & Computer Sciences, 2017, 25(2): 888 $\sim$ 900.

[13] Alshammari R, Zincir-Heywood A N. Identification of VoIP encrypted traffic using a machine learning approach. Journal of King Saud University - Computer and Information Sciences, 2015, 27(1): 77 $\sim$ 92.

[14] Yildirim T, Radcliffe P J. VoIP traffic classification in IPSec tunnels. In: International Conference on Electronics and Information Engineering, 2010: V1 ∼ 151 ∼ V1 ∼ 157.

[15] Gomes J V, Inacio P R M, Pereira M, et al. Identification of Peer-to-Peer VoIP Sessions Using Entropy and Codec Properties. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(10): 2004 ∼ 2014.

[16] Qin T, Wang L, Liu Z, et al. Robust application identification methods for P2P and VoIP traffic classification in backbone networks. Knowledge-Based Systems, 2015, 82: 152 ∼ 162.

[17] Sen S, Spatscheck O and Wang D. Accurate, scalable in-network identification of p2p traffic using application signatures[C]. In: Proceedings of the 13th international conference on World Wide Web, 2004: 512 ∼ 521.

[18] l7-filter[EB/OL].
http://l7-filter.sourceforge.net.

[19] Bujlow T, Carela-Español V and Barlet-Ros P. Independent comparison of popular DPI tools for traffic classification[J]. Computer Networks, 2015, 76: 75 ∼ 89.

[20] Zhang Z, Zhang Z, Lee P P, et al. Proword: an unsupervised approach to protocol feature word extraction[C]. In: INFOCOM, 2014 Proceedings IEEE, 2014: 1393 ∼ 1401.

[21] Yun X, Wang Y, Zhang Y, et al. A semantics-aware approach to the automated network protocol identification[J]. IEEE/ACM Transactions on Networking (TON), 2016, 24(1): 583 ∼ 595.

[22] Ikeuchi K, Wijekoon J, Ishida S, et al. GPU-based multi-stream analyzer on application layer for service-oriented router[C]. In: Embedded Multicore Socs (MCSoC), 2013 IEEE 7th International Symposium on, 2013: 171 ∼ 176.

[23] Vasiliadis G, Koromilas L, Polychronakis M, et al. GASPP: A GPU-Accelerated Stateful Packet Processing Framework[C]. In: USENIX Annual Technical Conference, 2014: 321 ∼ 332.

[24] Karagiannis T, Broido A, Faloutsos M, et al. Transport layer identification of P2P traffic[C]. In: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, 2004: 121 ∼ 134.

[25] Karagiannis T, Papagiannaki K and Faloutsos M. BLINC: multilevel traffic classification in the dark[C]. In: ACM SIGCOMM Computer Communication Review, 2005: 229 ∼ 240.

[26] Schatzmann D, Mühlbauer W, Spyropoulos T, et al. Digging into HTTPS: flow-based classification of webmail traffic[C]. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, 2010: 322 ∼ 327.

[27] Moore A W, Zuev D. Internet traffic classification using bayesian analysis techniques[C]. In: ACM SIGMETRICS Performance Evaluation Review, 2005: 50 ∼ 60.

[28] Auld T, Moore A W and Gull S F. Bayesian neural networks for internet traffic classification[J]. IEEE Transactions on neural networks, 2007, 18(1): 223 ∼ 239.

[29] Li W, Canini M, Moore A W, et al. Efficient application identification and the temporal and spatial stability of classification schema[J]. Computer Networks, 2009, 53(6): 790 ∼ 809.

[30] Este A, Gringoli F and Salgarelli L. Support vector machines for TCP traffic classification[J]. Computer Networks, 2009, 53(14): 2476 ∼ 2490.

[31] Bernaille L, Teixeira R, Akodkenou I, et al. Traffic classification on the fly[C]. ACM SIGCOMM Computer Communication Review, 2006, 36(2): 23 ∼ 26.

[32] Ester M, Kriegel H-P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]. In: Kdd, 1996: 226 ∼ 231.

[33] Cheeseman P, Kelly J, Self M, et al. Autoclass: A Bayesian classification system[C]. In: Readings in knowledge acquisition and learning, 1993: 431 ∼ 441.

[34] Erman J, Arlitt M and Mahanti A. Traffic classification using clustering algorithms[C]. In: Proceedings of the 2006 SIGCOMM workshop on Mining network data, 2006: 281 ∼ 286.

[35] Xu Q, Erman J, Gerber A, et al. Identifying diverse usage behaviors of smartphone apps[C]. In: Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, 2011: 329 ∼ 344.

[36] Choi Y, Chung J Y, Park B, et al. Automated classifier generation for application-level mobile traffic identification[C]. In: Network Operations and Management Symposium (NOMS), 2012 IEEE, 2012: 1075 ∼ 1081.

[37] Dai S, Tongaonkar A, Wang X, et al. Networkprofiler: Towards automatic fingerprinting of android apps[C]. In: INFOCOM, 2013 Proceedings IEEE, 2013: 809 ∼ 817.

[38] Xu Q, Andrews T, Liao Y, et al. FLOWR: a self-learning system for classifying mobileapplication traffic[J]. ACM SIGMETRICS Performance Evaluation Review, 2014, 42(1): 569 ∼ 570.

[39] Xu Q, Liao Y, Miskovic S, et al. Automatic generation of mobile app signatures from traffic observations[C]. In: Computer Communications (INFOCOM), 2015 IEEE Conference on, 2015: 1481 ∼ 1489.

[40] Yao H, Ranjan G, Tongaonkar A, et al. Samples: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic[C]. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, 2015: 439 ∼ 451.

[41] Ranjan G, Tongaonkar A and Torres R. Approximate matching of persistent LExicon using search-engines for classifying Mobile app traffic[C]. In: Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on, 2016: 1 ∼ 9.

[42] Wang W, Bickford J. WhatApp: Modeling mobile applications by domain names[C]. In: Wireless and Mobile Computing, Networking and Communications (WiMob), 2016 IEEE 12th International Conference on, 2016: 1 ∼ 10.

[43] Alan H F, Kaur J. Can Android Applications Be Identified Using Only TCP/IP Headers of Their Launch Time Traffic?[C]. In: Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2016: 61 ∼ 66.

[44] Sun Q, Simon D R, Wang Y-M, et al. Statistical identification of encrypted web browsing traffic[C]. In: Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on, 2002: 19 ∼ 30.

[45] Liberatore M, Levine B N. Inferring the source of encrypted HTTP connections[C]. In: Proceedings of the 13th ACM conference on Computer and communications security, 2006: 255 ∼ 263.

[46] Herrmann D, Wendolsky R and Federrath H. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier[C]. In: Proceedings of the 2009 ACM workshop on Cloud computing security, 2009: 31 ∼ 42.

[47] McCulloch W S, Pitts W. A logical calculus of the ideas immanent in nervous activity[J]. The bulletin of mathematical biophysics, 1943, 5(4): 115 ∼ 133.

[48] Hubel D H, Wiesel T N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex[J]. The Journal of physiology, 1962, 160(1): 106 ∼ 154.

[49] Fukushima K, Miyake S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition[M]. In: Competition and cooperation in neural nets. Springer, 1982: 267 ∼ 285.

[50] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278 ∼ 2324.

[51] Krizhevsky A, Sutskever I and Hinton G E. Imagenet classification with deep convolutional neural networks[C]. In: Advances in neural information processing systems, 2012: 1097 ∼ 1105.

[52] Abdel-Hamid O, Mohamed A-r, Jiang H, et al. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition[C]. In: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, 2012: 4277 ∼ 4280.

[53] Abdel-Hamid O, Mohamed A-r, Jiang H, et al. Convolutional neural networks for speech recognition[J]. IEEE/ACM Transactions on audio, speech, and language processing, 2014, 22(10): 1533 ∼ 1545.

[54] Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. Journal of Machine Learning Research, 2011, 12(Aug): 2493 ∼ 2537.

[55] Zhang X, Zhao J and LeCun Y. Character-level convolutional networks for text classification[C]. In: Advances in neural information processing systems, 2015: 649 ∼ 657.

[56] Real-Time Transport Protocol (RTP) Parameters, https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml.

[57] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278 ∼ 2324.

[58] Rumelhart D E, Hinton G E and Williams R J. Learning representations by back-propagating errors. Readings in Cognitive Science, 1988, 323(6088): 399 ∼ 421.

[59] Krizhevsky A, Sutskever I and Hinton G E. ImageNet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems, 2012: 1097 ∼ 1105.

[60] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Computer Science, 2014.

[61] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1 ∼ 9.

[62] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. In: Computer Vision and Pattern Recognition, 2016: 770 ∼ 778.

[63] Sainath T N, Vinyals O, Senior A, et al. Convolutional, long short-term memory, fully connected deep neural networks. In: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, 2015: 4580 ∼ 4584.

[64] Statistical Report on China's Internet Development[EB/OL]. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/.

图清单

表清单

致谢

感谢您使用本模板。

个人简历