# 基于注意力机制的序列模型在网络流量分类及保护上的应用

(申请清华大学工学博士学位论文)

培 养 单 位 ： 计算机科学与技术系

研 究 生 ： 肖 文 韬

指 导 教 师 ： 导师姓名

二〇二一年十一月

# Title

Thesis Submitted to

**Tsinghua University**

in partial fulfillment of the requirement

for the degree of

**Doctor of Philosophy**

in

by

**Name of author**

Thesis Supervisor: Name of supervisor

**November, 2021**

# 摘　要

不需要。

**关键词：**流量分类；注意力机制；序列模型；GAN

# Abstract

No need.

**Key Words:** Recommendation System; click-through rate; deep learning; xDeepFM; AutoInt; FGCNN

# 目 录

# 插图和附表清单

# 第 1 章　研究背景和意义

## 1.1　研究背景

网络流量分析是一个完整的过程，从拦截流量数据开始，到发现互联网网络中的关系、斑点、异常和错误配置，以及其他事项。特别是，流量分类是该领域策略的一个子组，其目的是根据用户的要求，将互联网流量分为预定义的类别，如正常或异常流量、应用程序的类型（流媒体、网络浏览、VoIP 等）或应用程序的名称（YouTube、Netflix、Facebook 等）。网络流量分类在保证网络安全和网络系统稳定运行方面发挥着重要作用，引起了学术界和工业界的广泛关注。网络流量分类之所以很重要，是因为有以下几个原因：a) 故障排除任务：主要目的是定位有问题的网络设备、设备/软件的错误配置、定位数据包丢失点、网络错误等。在这个领域，识别或分类网络中的应用程序的名称或类型，有助于处理一些事先的问题。b) 安全性：避免恶意软件或防止对私人信息的入侵。c) 服务质量（QoS）管理，以保证终端用户对应用或服务的总体接受度。例如，从流量中识别不同的应用，对于管理带宽资源和确保 QoS 要求至关重要。

在过去，流量分类依赖于基于端口的方法，即每个应用程序通过其注册的和已知的端口来识别，这些端口由互联网分配号码管理局（IANA）定义。这种方法现在已经变得不可靠和不准确，因为除其他因素外，具有未注册或随机生成的端口的新应用程序的扩散。另一种在该领域获得大量普及的方法被称为深度包检测（DPI）。DPI 在数据包有效载荷和存储的签名集之间进行匹配，以对网络流量进行分类。然而，当隐私政策和法律阻止访问数据包内容时，以及在协议混淆或封装的情况下，DPI 会失败。为了克服前述问题，机器学习成为一个合适的解决方案，不仅适用于流量分类任务，还适用于预测和新知识发现等[1-6]。在这种情况下，网络流量的统计特征通常从网络流中提取出来，并存储起来以生成历史数据。通过这种方式，可以用这些历史数据来训练不同的机器学习模型，并且可以用这些模型来分析新进入的流量。这些方法首先在数据包层面或流量层面提取特定协议的特征，然后使用这些特征来构建流量分类的分类器。在数据包层面，每个数据包都将被分类。在流量层面上，分类的粒度是流量。各种有监督和无监督的方法已经被用来对许多网络协议进行分类。然而，这些方法面临两个挑战性的问题。首先，由于存在巨大的新应用，手工寻找对所有应用都有效的新特征是很耗时和容易出错的。其次，现代应用的网络流量是如此的复杂和动态，以至于传统的挖掘方法不能很

好地概括出特定应用的准确指纹。

近十年来我国的互联网行业一直走在世界的前列，各式各样的信息科技创新模式和技术在我国互联网行业内生根发芽，且正以不可思议的速度发展。跟随互联网的发展和大数据科技的快速崛起，国内外的应用程序和网站呈现爆发式的增长。截至 2021 年 6 月[7]，我国三家基础电信企业的移动电话用户总数达 16.14 亿户，较 2020 年 12 月净增 1985 万户。其中，5G 手机终端用户达 3.65 亿户，较 2020 年 12 月增加 1.66 亿户。截至 2021 年 6 月，我国网民规模达 10.11 亿，其中手机网民规模达 10.07 亿，较 2020 年 12 月增长 2092 万，网民使用手机上网的比例为 99.6%。互联网给人们提供了越来越多的个性化资讯，并在工作、娱乐，生活中服务于大众，受到了人们的欢迎。在这一庞大的用户体量下，各种应用程序和网站的需求呈现爆发式增长。截至 2021 年 6 月，我国国内市场上监测到的应用程序数量为 302 万款，而且每天都会有一大批新的应用程序出现。截至 2021 年 6 月，我国网站数量为 422 万个。

科技是把双刃剑，互联网给人们生活带来便利的同时，也受到了恶意攻击者的关注。如今不管是应用程序还是网站大多存在安全隐患。以 Android 应用程序为例，根据移动互联网系统与应用安全国家工程实验室和爱加密移动应用大数据平台提供的数据，截止 2021 年 9 月底大数据平台共计收录 Android 移动应用程序 347 万款，其中 70% 以上存在高危漏洞威胁[8]，这类应用程序容易遭受不同形式的攻击。据调查，几乎所有（89%）的漏洞都可以被恶意软件利用[9]。正由于目前的应用程序漏洞总量特别庞大，这就有让黑客窃取个人密码、金融信息或隐私数据的可能。如多数应用程序存在滥用各类共享权限的情况，容易被不法分子利用，导致隐私的泄露。通过应用程序发起恶意攻击的攻势越来越猛烈。据统计，每年至少新增 150 万种恶意移动软件，至少造成超过 1600 万件的恶意移动软件攻击事件[10]。

2021 年 4 月，全球领先的网络安全解决方案提供商 Check Point® 软件技术有限公司发布了《2021 年移动安全报告》[11]。报告全面概述了移动恶意软件、设备漏洞及国家级网络攻击的趋势，并介绍了如何抵御当今和未来复杂的移动威胁。报告显示，2020 年，几乎每个组织都经历了至少一次移动恶意软件攻击。这些攻击中有 93% 源于互联网，它们试图诱骗用户通过受感染的网站或网址安装恶意应用，或者窃取用户凭据。46% 的组织至少有一名员工下载了威胁网络和数据的恶意应用。由于芯片组的缺陷，全球至少有 40% 的移动设备存在固有的漏洞，需要紧急打补丁。目前全世界正处于新冠疫情的影响之下，这些恶意软件通常会隐藏在生成提供疫情相关信息的应用中。诸如此类的应用程序用户可能遭受不同形式的恶

意攻击，包括流量分析攻击以及通过恶意应用程序攻击来窃取用户的个人隐私或者个人财产。

## 1.2　研究意义

# 第 2 章　国内外研究现状

## 2.1　网络流量分类

学术界关于网络流量分类的研究采用了很多种不同的技术。在本节中，我们主要关注基于机器学习的方法。尽管越来越多的应用程序使用 SSL/TLS 来保护他们的流量，但仍有大量的信息可以从加密的流量中得知，如数据包的长度、方向、元数据和 TCP 时间戳。我们把这种信息称为侧信道数据。同时，有效载荷内容也在从网络流量中获取有用信息方面发挥着重要作用，即使它们是加密的。根据流量分类中使用的信息，这些方法可分为两类。1）基于侧信道数据的方法，和 2）基于有效载荷内容的方法。

### 2.1.1　基于侧信道数据的方法

基于旁路数据的方法主要采用数据包头和流量的信息。典型的侧信道数据包括根据流量中的源和目标 IP 地址的数据包长度和方向等。整个网络流量中的侧信道数据序列也包含有用的信息。因此，现有的研究通常首先获取侧信道数据序列，然后将其转化为适合机器学习的特征[3]。特别是，可以从侧信道数据中提取两类特征用于流量分类。

#### 2.1.1.1　统计特征

许多统计数据可以从侧通道数据序列中获得，包括计数、最大、最小、平均、标准差、方差、倾斜、中位绝对偏差、峰度、百分比等。这些数字是由整个流或子流计算出来的。一个子流程是具有某些约束的流程的一部分。例如，突发是一种子流，由方向相同的连续数据包组成[12]。Roughan 等人使用最近的邻居、线性判别分析和二次判别分析方法，根据统计数据对流量进行分类[13]。Bernaille 等人观察到，TCP 连接的前几个数据包的大小和方向很重要。根据这一观察，他们提出了一个基于简单 K-means 的模型[14]。有许多基于集群技术的方法。Zhang 等人[15] 利用流量相关信息，通过流量标签传播和包括 K-means 聚类在内的复合分类技术来识别未知应用。Luis Sacramento 等人[16] 在进行聚类时发现，恶意流量位于真实流量中的小聚类，并实现了一个名为 FlowHacker 的新型网络入侵检测系统（NIDS）。他们的方法从流量中自动提取特征并利用两阶段聚类来迭代检测攻击。

此外，Taylor 等人实现了一个应用分类框架[3]，它利用数据包长度序列的统计

特征，对智能手机应用进行指纹识别。他们评估了其在不同设备、应用程序版本和时间上的鲁棒性。最近，在区块链的新兴领域[17-18]，如智能合约和去中心化的应用程序，Shen 等人[19] 识别了以太坊[20] 上的去中心化应用程序（DApps）的加密流量。他们通过利用多项式和径向基等核函数，融合了三个不同维度的统计特征（即数据包长度、突发和时间序列）。对于数据包长度，Kampeas el at. 提出了一种有效的采样策略[21]，以满足软件定义网络（SDN）环境中的实时分类需求，该策略只使用有效载荷长度为零的数据包，如 TCP 流中的 ACK，以恢复整个数据包长度序列。他们利用流量中的前 $k$ 个 a-APDUs 作为特征，其中 a-ADPU 是在同一方向传输的连续数据包的数据字节长度之和。

#### 2.1.1.2　序列特征

同一流量中的数据包之间的关系信息在网络流量分类中也起着至关重要的作用。由于侧信道数据可以从加密的网络流量中提取并形成时间序列，因此使用基于马尔可夫模型的方法自然是合适的。Anderson 等人[22] 利用三种特征对加密的恶意软件流量进行了二元分类：长度和时间的统计特征、由流量的前 50 个数据包的大小构建的一阶马尔可夫链以及 TLS 握手元数据的特征。随后，多属性马尔可夫概率指纹（MaMPF）在[23] 中被提出。作者发现，数据包的长度是受幂律分布影响的。因此，他们首先采用幂律划分，将数据包长度序列转化为更具辨识度的长度块序列（LBS）。然后，为消息类型序列（MTS）和 LBS 建立了两个马尔可夫模型。最后，这些模型的输出概率被送入机器学习分类器。

由于循环神经网络（RNN）适合于序列数据的建模，Liu 等人设计了一个新的深度学习模型，名为流量序列网络（FSNet）[24]，用于使用数据包长度序列的加密流量级分类。FS-Net 利用自动编码器架构与重建机制来促进特征学习，其中编码器和解码器都是两层 Bi-GRU。

**总结**：大多数现有的方法都是基于从数据包头中提取的特定侧信道特征建立模型。为了利用更粗略和更广泛的侧信道特征，而不是现有研究中使用的手工制作的特征，我们的模型从网络流量中自动学习最合适的侧信道特征。

### 2.1.2　基于有效载荷内容的方法

数据包的有效载荷可被视为一串字节或比特。为了利用有效载荷提供的信息，传统方法试图找到与网络流量相匹配的签名。这些签名通常是 $n$-grams 或关键词的形式，网络流量的识别利用了有效载荷中签名的出现频率。除了这些传统的基于签名的方法，自然语言处理（NLP）的技术，如主题模型也被使用。同时，深度学习也被引入，以学习复杂但有用的数据包的表示。

#### 2.1.2.1　基于签名的方法

基于签名的方法使用从有效载荷中提取的签名从网络流量中识别应用协议。Deri 等人开发了一个开源的高速深度包检测工具，名为 nDPI[25]，它支持超过 500 种不同的协议。通过使用从数据包头和有效载荷中提取的字节级的签名，DPI 可以在不降低网络吞吐量的情况下进行流数据中的字符串检测。它取得了很高的准确性和效率，但需要对每个协议有预先的了解。为了克服这一限制，Hubballi 等人提出了一种基于比特级 DPI 的签名构建方法，称为 BitCoding[6]。为了自动匹配所有具有相同标签的样本的比特序列（即数据包有效载荷的内容），他们使用数据包中有效载荷的前 $k$ 比特（如果第一个数据包的长度小于 $k$ 比特，则使用流），从计数中产生一个压缩的签名。该签名与正则表达式类似，只有三个运算符，即 .、$*$ 和 $\{n\}$。他们还使用 Relaxed Hamming Distance[6] 来测量签名的重叠度，可以通过增加 $k$ 来减少。

此外，Haffner 等人利用三种机器学习算法，即 Naive Bayes、AdaBoost 和 Maximum Entropy，从 TCP 流的前 $n$ 个字节中自动提取签名[26]。

#### 2.1.2.2　基于主题模型的方法

由于 NLP 中的词-文档关系和网络流量中的字节-负载关系的相似性，一些研究使用自然语言处理（NLP）中的主题模型技术来寻找网络流量分类的有力关键词。主题模型的目的是发现文档的抽象关键词。同样地，一些研究人员致力于寻找通用的关键词来识别网络流量中的数据包。Securitas[27] 是第一个基于主题模型的方法，它应用 Latent Dirichlet Allocation（LDA）和词包模型来提取协议的特征。该特征是具有相同协议的有效载荷中的关键词的分布。然后，采用支持向量机（SVM）、C4.5 决策树或贝叶斯网络来做分类。此外，Xiao 等人[5] 使用动态集群主题模型来获得集群中最常见的关键词，用于流量分类模型。

#### 2.1.2.3　基于深度学习的方法

深度学习已经被广泛应用于许多重要领域，包括流量分类和网络攻击检测[28]。在智能家居软件定义网络（SDN）的场景中，Wang 等人提出了一个受 SDN 启发的家庭网关框架（SDN-HGW）[4]，提供细粒度的应用级流量分类。他们开发了名为 DataNets 的加密数据分类器，该分类器由三个虚构的深度学习模型组成，即多层感知器、卷积神经网络（CNN）和堆叠的自动编码器。同时，Lotfollahi 等人提出了 DeepPacket[29]，具有类似的深度学习模型。一维（1D）CNN 是在他们的实验中取得最佳性能的拟议模型之一。它将一维字节矢量化的数据包转化为二维的张

量。Giuseppe Aceto 等人利用多模态深度学习[30] 和融合[31] 技术来提高性能。这些模型要么错过了网络流量的特点，要么涉及许多高开销模块，引入了不必要的计算开销。因此，我们设计了一个新颖的神经网络，它适合网络流量的特点，并且从实验结果来看，它的性能优于 DeepPacket[29]。

**总结**：与基于侧信道数据的方法相比，基于有效载荷内容的方法涉及更多流量信息，即数据包的有效载荷内容。由于网络流量行为的复杂性，传统的机器学习方法很难提取特征。我们设计了一种新颖的基于深度学习的流量分类方法，不需要提取特征。

## 2.2　网络流量保护

网络流量保护系统对数据包进行修改（增加数据包或负载、延迟或重路由），以防止使用攻击者使用网络流量分类技术分析网络流量。广义上讲，这些保护措施要么是根据专家设计的噪音启发式方法来混淆网络流量，要么是利用对抗扰动来逃避机器学习分类器。

### 2.2.1　基于流量混淆的保护

这些防御措施旨在混淆痕迹以增加分类的难度。这种混淆是在应用层或网络层进行的。由于这些防御措施没有专门针对攻击者的分类器，它们通常对最先进的网络流量分类攻击提供较低的保护（<60%）。

为深度神经网络（DNN）模型提供解释，对于它们在安全敏感领域的使用至关重要。已经提出了大量的解释模型，以帮助用户了解 DNN 的内部工作原理：DNN 是如何对给定的输入作出具体决定的？改进的可解释性被认为是通过让人类参与决策过程来提供一种安全感。然而，由于其数据驱动的性质，可解释性本身有可能受到恶意操纵的影响，到目前为止，人们对此知之甚少。

在应用层混淆中，防御者将随机性引入 HTTP 请求或 Tor(洋葱网络) 路由算法中[32-33]。应用层防御通常会做出强大的假设，而这些假设在实践中往往是不现实的，例如目标网站实施定制的 HTTP 协议[32] 或只允许攻击者观察单一 Tor 入口节点的流量[33]。这些防御措施对基于 DNN 的攻击，如 DF 和 Var-CNN[34-35]，提供不到 60% 的保护。

在网络层混淆中，防御者在网络痕迹中插入假数据包，使网站识别更具挑战性。首先，早期的防御措施[36-37] 使用恒定速率填充来减少由时间间隔和流量引起的信息泄漏。然而，这些方法导致了巨大的带宽开销（>150%）。其次，最近的工作通过在数据包之间有大的时间间隔的位置插入数据包（WTF-PAD[38]）或专注于跟

踪的前面部分来减少开销，这已经被证明包含了最多的信息（FRONT[39]）。然而，WTF-PAD 和 FRONT 在面对强势网络流量分类攻击时，分别只取得了 9% 和 28% 的保护成功率。最后，超级序列防御[40-41] 试图找到一个超级序列，它是一个较长的数据包痕迹，包含不同网站痕迹的子序列。最强的超级序列防御，Walkie-Talkie，对 DNN 攻击实现了 50% 的保护。总的来说，面对强大的网络流量分类攻击，网络层的混淆防御要么引起极大的开销（>100%），要么提供低保护（<60%）。

### 2.2.2　基于对抗扰动的保护

Goodfellow 等人首先提出了针对 DNN 的规避攻击，即攻击者通过向 DNN 添加小的、对抗性的扰动[42]，导致 DNN 对输入进行错误分类。这种攻击在许多领域都得到了广泛的研究，例如，计算机视觉[43-44]，自然语言处理[45-46]，和恶意软件检测[47-48]。最近的网络流量保护[49-50] 使用对抗性扰动来击败基于 DNN 的攻击。

基于对抗性扰动的网络流量保护所面临的挑战是，对抗性扰动是针对给定的输入计算的，在网络流量保护背景下，输入是完整的网络轨迹。因此，计算保护网络连接所需的对抗性扰动，需要防御者在连接之前就知道整个轨迹。这一限制使得防御者在实时保护用户痕迹时不切实际。Mockingbird[50] 建议，我们可以使用最近的痕迹数据库来计算扰动，并在新的痕迹上使用它们。

在一份最新的研究成果中，Nasr 等人[51] 提出通过预先计算未见踪迹的通用对抗扰动来解决这一限制，从而实现对实时踪迹的保护。然而，意识到这种防御的攻击者也可以计算这些通用扰动，并针对这些扰动对他们的模型进行对抗训练以提高稳健性。这种对策导致其保护率大幅下降到 76%。

**总结**：目前最先进的网络流量保护方法是基于对抗扰动的保护，然而已有的研究成果要么因为复杂度太高、或者需要完整（包含未来的还没有发送的部分数据包）的网络流信息而无法实用，要么会因为攻击者使用对抗训练而失效。我们将在本研究中实现实时的、可以实际部署在真实场景的、能够抵抗对抗训练的网络流量保护方法。

## 2.3　序列模型在自然语言处理上的应用

由于网络流量可以被视为网络应用之间的语言对话，所以 NLP 中的技术已经被应用于流量分类，特别是自然语言分类中的深度学习网络。其中，卷积神经网络（CNN）和循环神经网络（RNN）是两个广泛使用的深度学习网络。RNN[52] 保持着对历史信息的内部记忆，对于从连续的数据中捕捉上下文特征是很有效的。长短期记忆（LSTM）[53] 是一种特殊而强大的 RNN。它可以通过引入遗忘、输入和

输出门来删除或增加记忆。在 LSTM 的基础上，门控循环单元（GRU）[54] 将遗忘门和输入门合并为一个” 更新门”，并做了一些其他的改变。

为了捕捉文档中句子之间的连贯性，层次递归神经网络语言模型（HRNNLM）[55] 采用了 RNN 来进行文档建模。通过两步训练过程，HRNNLM 在句子层面和单词层面都具有收敛性。Lai 等人介绍了一个用于文本分类的递归卷积神经网络（RCNN）[56]。与传统的神经网络相比，递归结构能以较少的噪音捕捉到上下文信息。RCNN 可以自动决定哪些词在文本分类中更重要。

此外，分层网络的创建是为了反映分层注意力网络（HANs）中的文档结构[57]。特别是，HAN 采用了两个注意层，以确保每个句子和单词在文档表示中具有不同的重要性。

**总结**：这些使用序列模型的研究在自然语言处理中取得了良好的性能。序列模型擅长对系列进行建模，学习其中的上下文语义信息，同时注意力机制的引入，在可解释性上和模型的性能上都带来了重大的提升。而网络流量不管是数据流还是数据包都可以表示为一个序列，并且序列间的上下文语义信息丰富。受此启发，我们设计了基于注意力机制的序列模型，用于网络流量分类和保护。

# 第 3 章　研究目标和拟解决的关键问题

## 3.1　研究目标

整体的研究围绕网络流量安全展开, 总体分为两个方向:(1) 网络流量分类;(2) 网络流量保护。在本研究中，我们将探索基于注意力机制的序列模型在上述两个方向的应用。

## 3.2　拟解决的关键问题

- **包级别和流级别的高精度网络流量分类模型**：网络流量分析技术在网络监控与管理、用户行为分析等领域具有重要应用，区分流量所属的网站和应用是流量分析的首要关键步骤。如何从原始流量字节中自动学习特征表示，以及如何高精度地对各种网站和应用的流量进行分类是至关重要的问题。
- **基于深度学习的网络流量分类模型的可解释性**：虽然基于深度学习的方法在许多网络安全的研究中取得了可观的进展，但是基于深度学习的方法仍然饱受可解释性差的问题，这使得这类方法的实际部署存在很多的困难。为基于深度学习的模型提供解释，对于它们在安全敏感领域的使用至关重要。可解释性帮助用户了解基于深度学习的方法的的内部工作原理：深度学习模型是如何对给定的输入作出具体决定的？可解释性能够通过让人类（用户）参与决策过程来提供一种安全感。因为如果无法解释这类方法的运行原理，方法的使用者无法评估和控制这些方法的误差上界和行为。
- **针对基于网络流量分类的攻击的网络流量保护算法**：通过网络流量分类，攻击者可以监听用户的流量窥探出用户访问的应用和网站。这导致用户隐私的泄漏。即使用户使用虚拟专用网络（VPN）和洋葱网络（Tor）等手段也无法避免被攻击。这促使学术界开始研究针对基于网络流量分类的攻击的网络流量保护算法，而已有的方法在存在着无法实际部署、资源消耗过大、保护效果差等问题。
- **高吞吐量**：因为在实际应用中，我们需要将网络流量分类和保护的算法部署到真实设备中，并服务于大量的用户流量。这要求我们提出的模型达到满足实时处理的模型执行速度，即高吞吐量。

## 3.3　预期创新点

针对上述研究目标和关键问题，我们提出的下一代智能网络流量分类及保护技术在实际应用方面和理论研究方面都有创新，预期在以下几个方面达到创新：

- 高精度、高吞吐量、高可解释性的网络流量分类方法。
- 针对基于网络流量分类的攻击的网络流量保护算法。
- 基于注意力机制的序列模型在网络流量分类及其保护上的理论研究。
- 在真实数据集上的充分实验证明我们提出的方法的有效性。

# 第 4 章　研究内容和方法

## 4.1　研究框架和研究内容

## 4.2　网络流量分类

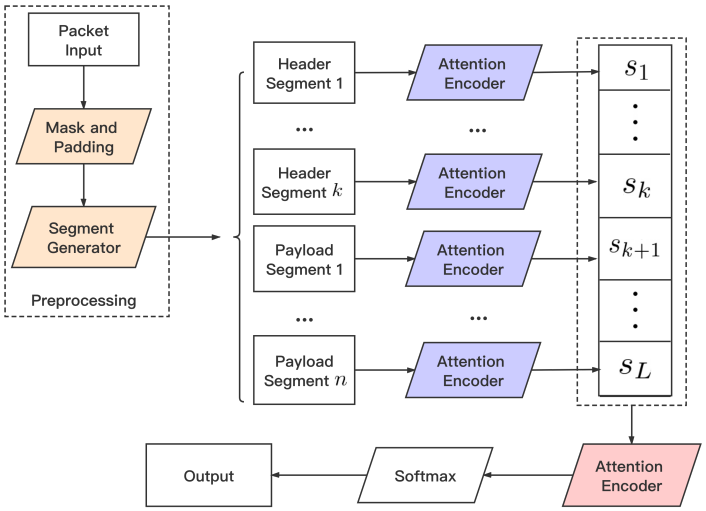


图 4.1　EBSNN 整体架构

在本节中，我们将详细介绍字节段神经网络（EBSNN）的架构。EBSNN 的初始目标是将单个数据包划分为相应的应用或网站。EBSNN 在数据包级的整体结构如图 1 所示。在我们的模型中，每一个单包都经过预处理后送入段生成器，并分解成一系列段，包括头段和有效载荷段。随后，注意力编码器将每个段转化为段向量。然后，将数据包的所有这些向量组合起来，放入另一个注意力编码器中，得到整个数据包的表示向量（在图 4.1中，不同颜色的注意力编码器具有不同的 RNN 层）。最后，将表示向量引入 softmax 函数，计算出数据包的预测类。本方法的详细内容将在以下几个小节中描述。第 4.2.1小节介绍了预处理的细节，特别是分段生成器的细节；第 4.2.2小节给出了所提出的模型的形式定义；第 4.2.3小节描述了我们所提出的模型在网络流级分类中的扩展。

### 4.2.1　预处理

为了将原始数据转化为适合模型的合适表示方式，我们首先需要对原始数据包进行预处理。整个数据包在应用层上被分割成头文件和有效载荷。我们首先读取二进制格式的数据包，并将其呈现为一个范围为 [0,255] 的 8 位整数序列。然后，

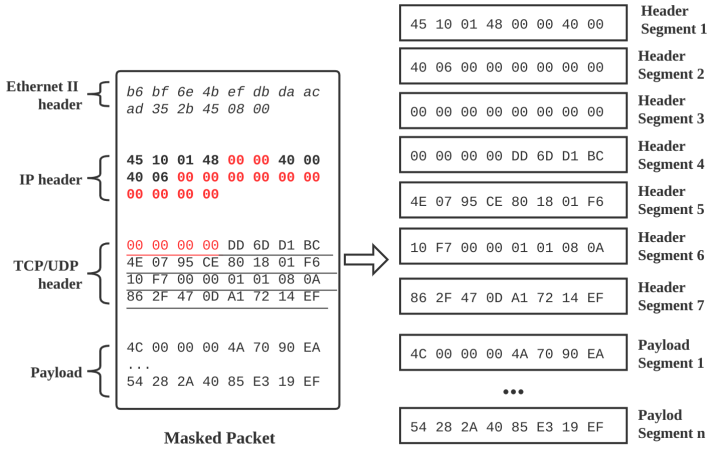


图 4.2　掩蔽后的包的段生成器实例

段生成器将该序列分解成固定长度 N 的字节段，称为段长。字节段有两种，即从头部产生的头部段和从有效载荷产生的有效载荷段。为了得到有效载荷段，在大多数情况下，有效载荷的字节总数不是段长的倍数，在有效载荷的末尾垫入 0，字节构造一个段，因此段的数量为 $\lceil M/N \rceil$，其中 M 为有效载荷的长度。段生成器提供了原始有效载荷的简单表示。有效载荷段中的每个字节都可以看作是一个字符。因此，一个有效载荷段类似于自然语言中的一个句子，那么一个数据包就可以看作是一个文档。

与 BSNN[58] 相比，我们在 EBSNN 中加入了包头的侧信道特征。除了有效载荷段外，我们还从包头中获取头段。数据包头由以太网 II 头、IP 头和 TCP/UDP 头组成。虽然包头信息可以被利用来进行流量分类，但并不是所有的包头位都是有用的，部分包头位 (如 IP 头中的源地址和目的 IP 地址) 可能会导致过拟合。因此，我们首先对报头中的一些字段进行屏蔽，然后从报头中生成报头段。

1. Ethernet II 头: 由于该头只包含 EtherType 以及源地址和目的 MAC 地址，我们干脆放弃整个 Ethernet II 头。
2. IP 头: 一些不提供有用信息的字段, 如校验 (checksum) 和识别 (identification)。另外，IP 地址是个人数据。因此，我们做一些掩码工作，即把 IP 头中的一些字段用 0 代替：IP 标识（IP 头中的 32~37 位）、IP 校验和（80~95 位）、源 IP 地址（80~95 位）、目的 IP 地址（96~127 位）。之后，包括填充在内的段生成器与有效载荷段的段生成器相同。
3. TCP/UDP 头: 与 IP 头类似，我们将源端口（0~15 位）和目的端口（16~31 位）用 0 代替。

按照上述的方式，我们得到 IP 头段、TCP/UDP 头段和有效载荷段。每一个段都是一个整数序列，在输入一个注意力编码器之前其长度为 N。注意力编码器的

目标是寻求数据包的最佳表示，并理解数据包。与手工制作的侧通道特征不同[59]，我们提出的 EBSNN 模型通过使用随机梯度下降优化器（如 Adam[60]）自动学习和选择比手工制作方法更合适的侧通道特征。

图 4.2是段生成的一个例子。图中，左边部分是十六进制的屏蔽数据包，其中两个数字代表一个字节。图中斜体部分为以太网 II 头，在预处理时将其删除。数据包中粗体部分表示 IP 头，下划线部分表示 TCP 头。红色的零是数据包中被屏蔽的内容。在这个例子中，数据包被分成了 N=8 个短段（见图中右边部分），其中，一个段涉及 8 字节。在图中，有 7 个头段和 n 个有效载荷段。

### 4.2.2　模型

EBSNN 通过采用分层 RNN 单元, 结合每个字节和每个片段的上下文信息, 实现对输入序列的合理表示。此外，它还使用关注层来找出哪部分对整个数据包是重要的。整个数据包的集合可以表示为 $A = \{\{a_{(i,n)}\}_{(n=1)}^{N}\}_{(i=1)}^{L}$，其中 $a_{(i,n)}$ 表示段 i 中的第 n 个字节，包括头段和有效载荷段，L 是段的数量。为了将 RNN 单元应用到数据包中, 我们需要将字节嵌入到向量中。如上所述, 字节 $a_{(i,n)}$ 是介于 0 到 255, 之间的整数, 可以将其转化为 256 维的独热向量。然后，将独热向量嵌入到稠密向量 $x_{(i,n)}\ in\mathbb{R}^{E}$ 中。为了从片段中获取信息，我们采用了一个 RNN 结构，其注意力机制的灵感来自于 HAN[57]。这个结构在下文中被称为注意力编码器。起初，我们得到:

$$[h_{i,1}, \cdots, h_{i,N}] = \mathsf{RNN}^{(1)}([x_{i,1}, \cdots, x_{i,N}]), \tag{4-1}$$

其中 $\mathsf{RNN}^{(1)}(\cdot)$ 表示隐藏大小为 $r$ 的 RNN 层（如 LSTM 或 GRU）。相对于 $x_{i,n}$ 的注意力输出是 $h_{i,n} \in \mathbb{R}^{f}$，其中 $f$ 是 $r$ 乘以 RNN 的方向数。注意力编码器中的 RNN 结构可以是单向的，也可以是双向的。

接下来, 将一个网段中所有字节的输出, $h_{i,1}, \cdots, h_{i,N}$ 合并为一个段矩阵, $\boldsymbol{H}_i = [h_{i1}, \cdots, h_{iN}]^{\top} \in \mathbb{R}^{N \times f}$，以代表整个段。具体来说，由于不同的字节在流量分类中可能具有不同的重要性，因此采用关注机制对输出进行整合。

形式上，对于 $\boldsymbol{H}_i$ 的注意力输出 $s_i \in \mathbb{R}^{f}$ 为:

$$\begin{aligned} s_i &= \text{Attention}(q, \boldsymbol{H}_i) \\ &\coloneqq \boldsymbol{H}_i^{\top} \cdot \text{softmax}(\text{FC}(\boldsymbol{H}_i) \cdot q), \end{aligned} \tag{4-2}$$

其中 FC(·) 表示全连接层: $\text{FC}(\boldsymbol{H}_i) \coloneqq \tanh(\boldsymbol{H}_i \boldsymbol{W}_g + b_g) \in \mathbb{R}^{N \times f}$。在上式中, $q \in \mathbb{R}^{f \times 1}$ 是段中信息量最大的字节的抽象表示，它是随机初始化的。将向量归一化为概率

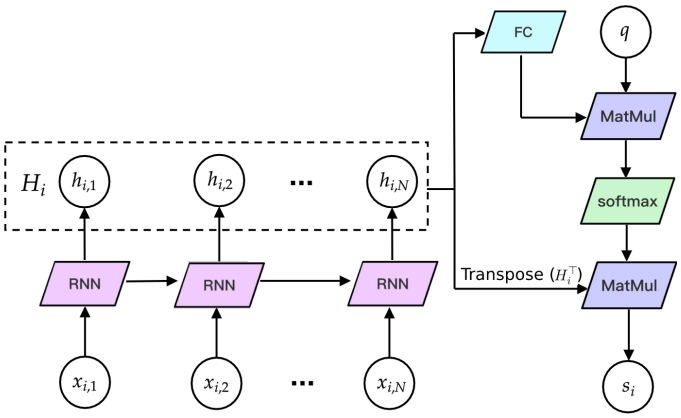


图 4.3　注意力编码器的结构

分布，其定义为:

$$\text{softmax}(z)_i \coloneqq \frac{\exp z_i}{\sum_{j=1}^{n} \exp z_j}, \tag{4-3}$$

其中 $z \in \mathbb{R}^n$ 是未归一化的输入向量，输出也是一个 $n$ 维向量。

从 $[x_{i,1}, \cdots, x_{i,N}]$ 到 $s_i$ 的变换称为注意力编码器: $s_i = \text{AttEncoder}([x_{i,1}, \cdots, x_{i,N}])$。图 4.3为注意力编码器的整体结构，其中输入为 $[x_{i,1}, \cdots, x_{i,N}]$，输出为 $s_i$。图中 MatMul 表示矩阵乘法。使用相同的注意力编码器与相同的 RNN 层 $\text{RNN}^{(1)}(\cdot)$，每个段都被转换为新的表示。我们将数据包的所有段的表示集合表示为 $S = [s_1, \cdots, s_L] \in \mathbb{R}^{f \times L}$。请注意，$S$ 是一系列的向量 $[x_{i,1}, \cdots, x_{i,N}]$，就像一个单一的嵌入段。因此，我们采用另一个不同的 $\text{RNN}^{(2)}(\cdot)$ 的注意力编码器来从 $S$ 得到整个包的注意力输出。最后，得到整个数据包的表示向量 $d = \text{AttEncoder}([s_1, \cdots, s_L]) \in \mathbb{R}^f$。由于我们已经有了数据包的表示 $d$，我们可以转向分类问题。对于 $C$ 分类，$d$ 可以将其转化为一个 $C$-维向量 $\tilde{y} = \text{softmax}(Wd + b)$，其中 $\tilde{y} \in \mathbb{R}^C$。因此，我们模型的整个过程可以表述为:

$$\tilde{y} = \text{EBSNN}(A; \theta), \tag{4-4}$$

其中 $\theta$ 为 EBSNN 的模型参数。

考虑到训练损失，我们在 EBSNN 中采用了焦点损失[61]，因为它在不平衡多分类问题上表现出强大的优势。焦距损失的关键思想是，那些难以训练的实例对整个神经网络来说是重要的。对于 $C$-类分类的焦点损失可以简单计算如下:

$$\text{FL}(y, \tilde{y}; \alpha, \beta) \coloneqq -\alpha(1 - \tilde{y}_l)^\gamma \log(\tilde{y}_l), \tag{4-5}$$

其中 $y \in \mathbb{R}^C$ 是数据包的真实标签、$\alpha$ 是类权重和聚焦参数的独热向量。$l$ 是真实标签，使得 $y_l = 1$ 和其他 $y_i$ $(i \neq l)$ 都是零。$\tilde{y} \in \mathbb{R}^C$ 是我们提出的模型预测的标签

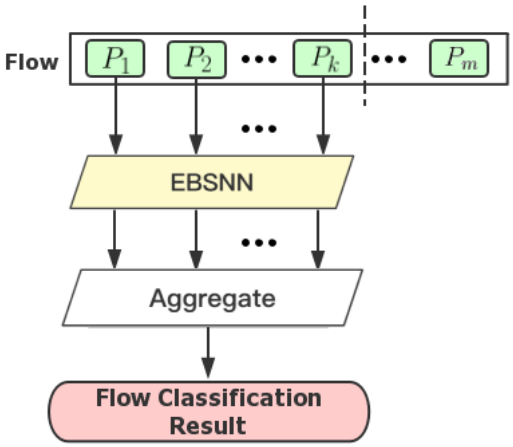


图 4.4　网络流级别分类扩展的工作流

的概率分布。

综上所述，EBSNN 的迷你批量输入和参数为 $\mathcal{X}$ 和 $\theta$。在优化算法方面，我们采用 Adam 优化器[60]。

### 4.2.3　延伸到网络流级分类

BSNN 的初始目标是识别单个数据包，并在数据包级进行分类。在 EBSNN 中，我们对 BSNN[58] 进行了扩展，通过引入聚合策略 $\text{agg}(\cdot): \mathbb{R}^{k\times C} \rightarrow \mathbb{R}^{C}$ 使其适合于流级分类。有研究[14,62] 表明，流的前 $k$ 个数据包涉及流的重要信息。因此，流中第一个具有非空的应用级有效载荷的数据包可以代表流。与 BSNN 相比，EBSNN 提供了流层面的支持，可以用来对整个流进行分类。

正如我们在上一小节中提到的，EBSNN 学习每个数据包的注意力编码表示 d，并利用 softmax 函数计算数据包标签的概率分布 $\hat{y}$。图 4.4展示了扩展的整体工作流程和架构。一个流由一个数据包序列组成，包括一些零长度的数据包（如 ACK 和 SYN）。让来自数据集的流为 $x = [P_1, P_2, \cdots, P_k, \cdots, P_m]$，其中 $P_i$ 是流中第 $i$-个具有非空的应用级有效载荷的数据包，它涉及头和有效载荷。由于流中的前几个数据包在网络流分类中起着至关重要的作用，我们只对流中具有非空应用级有效载荷的前 k 个数据包应用 EBSNN。之后，可以得到这些数据包的 softmax 输出。最后，我们对这些输出进行聚合，得到流的标签。详细来说，这些聚合策略包括 softmax 输出和数据包结果的投票之和等。

值得一提的是，流量级分类方法[14,62] 主要是基于特定的侧通道特征，如长度和方向。这些侧信道特征主要是从原始头（如 TCP 头）中提取的。除了常用的长度和方向特征外，数据包头还提供了更多的特征，如 TCP 头中的选项字段和 TLS 握手元数据。与这些方法相比，原始的数据包头被输入到 EBSNN 中，这样涉及到

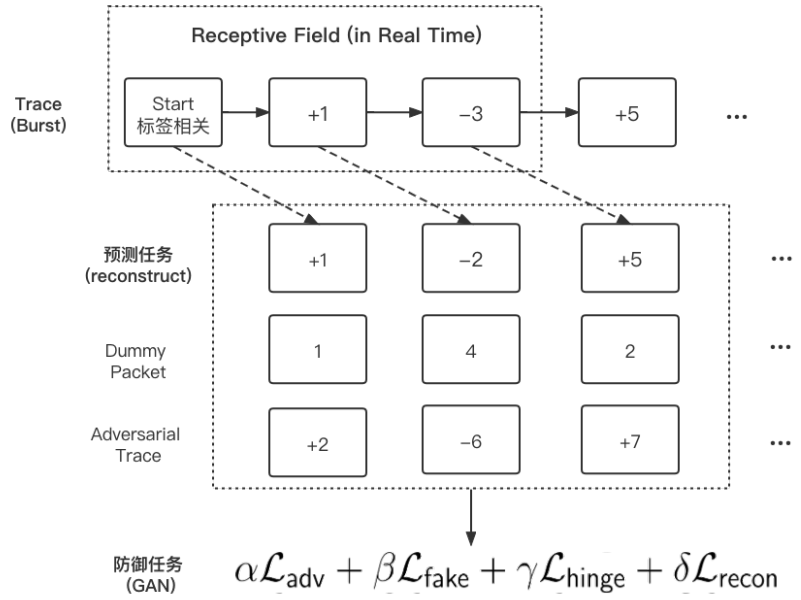


图 4.5　网络流量保护工作原理

更多的侧信道特征（如 TCP 头中的选项字段的特征），我们的方法会自动学习最合适的侧信道特征。因此，EBSNN 集中了头中的侧信道特征和有效载荷中的内容数据这两个方面更全面的信息。

## 4.3　网络流量保护

在本研究中，我们使用基于注意力机制的序列模型和生成式对抗网络（GAN）来自动的在网络流中插入噪声数据包，以使得攻击者无法通过网络流量分类方法进行攻击。总体的工作原理如图 4.5所示。模型主要分为生成器和判别器两个部分，生成器用于给定当前已发送出去的部分网络流量数据包的信息，计算得到下一次发送数据包的时候需要添加的噪声数据包的数量。而判别器则是攻击者使用的一个网络流量分类模型，可以是白盒（攻击者使用的模型的具体架构和参数我们都知道）也可以是黑盒（我们无法获得攻击者使用的模型的具体信息）。训练则是按照 GAN 训练的方式来进行：生成器和判别器轮流训练，通过精心设计的目标函数使得两个模块互相竞争，共同增强。

给定当前已经发送的包长序列 $\tilde{\boldsymbol{x}} = [x_1, \cdots, x_n]$，例如 $[+600, -1500, -120, +738, +289, -200, -400, -1400]$，正数代表正方向（实现规定好，例如客户端到服务端），负数代表负方向。因为像洋葱网络之类的工具产生的流量的每个数据包的长度都一样，不失一般性地，我们首先根据每个包的方向将其映射到 $\{-1, +1\}^n$ 空间。例如 $[+1, -1, -1, +1, +1, -1, -1, -1]$。然后我们将其

转化为突发序列 $\boldsymbol{x} \in \mathbb{Z}^n$，突发只指相同方向连续的数据包的累计个数（或大小）。因为在实际网络设备中，为了提高效率，数据包的发送都会存入到一个队列中，当队列满才会发送这些数据包。所以我们将其转化为突发序列有助于更加高效和符合实际地进行网络流量保护。上述的例子转化为突发序列为 $[+1, -2, +2, -3]$。攻击者可以训练出一个网络流量分类模型 $c = \text{argmax}\mathbb{F}_{\text{attack}}(\boldsymbol{x})$ 来对网络流量进行分析，并得到网络流量对应的网站、应用等类别标签 $c$，进而获取用户的隐私。

为了进行网络流量保护，我们将会在每一个突发中加入一些噪声数据包（对抗扰动），使得攻击者的攻击模型无法正常工作。而这些对抗扰动的数量由防御模型计算得到，即噪声数据包序列 $\delta = \mathbb{F}_{\text{defense}}(\boldsymbol{x}) \in \mathbb{Z}^n$。因为数据包突发序列是有方向的，而我们添加的噪声数据包也需要严格按照方向，故需要限制 $\forall \delta_i, \text{sign}(\delta_i) = \text{sign}(x_i)$。保护后的网络流量的突发序列可以表示为 $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta}$。网络流量保护的目标主要有三个：

1. 添加噪声数据包后的网络流量使得攻击者的网络流量分析模型得到错误的标签结果，$\mathbb{F}_{\text{attack}}(\hat{\boldsymbol{x}}) \neq c, c = \mathbb{F}_{\text{attack}}(\boldsymbol{x})$。
2. 因为添加噪声数据包会带来带宽的额外负担以及产生延迟，所以我们需要尽可能在使得攻击者模型出错的前提下噪声数据包的数量足够少：$\min \sum_{i=1}^{n} |\delta_i|$。
3. 当保护模型计算第 $i$ 个突发需要添加的噪声数据包的时候，只能知道已发送或者已接收的历史数据包 $[x_1, \cdots, x_i]$ 的信息，而无法知晓未来还没有发送或接收的数据包 $[x_{i+1}, x_{i+2}, \cdots]$ 的信息。也就是说，$\delta_i = \mathbb{F}_{\text{defense}}([x_1, \cdots, x_i])$。

为了达到上述目的，我们结合序列模型、生成式对抗网络、对抗样本设计了一组新颖的目标函数，如图 4.5所示。同时，为了增强我们的保护模型对流量序列特征的上下文语义的感知能力，我们的保护模型的输出除了噪声数据包的数量之外，还需要预测下一个（未来的）突发的大小 $\delta_i, x_{i+1}^{(\text{pred})} = \mathbb{F}_{\text{defense}}([x_1, \cdots, x_i])$，理想情况下 $x_{i+1}^{(\text{pred})} = x_{i+1}$，目标函数为：

$$\mathcal{L} = \alpha \mathcal{L}_{\text{adv}} + \beta \mathcal{L}_{\text{fake}} + \gamma \mathcal{L}_{\text{hinge}} + \sigma \mathcal{L}_{\text{recon}} \tag{4-6}$$

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{\hat{\boldsymbol{x}} \sim T'} \text{ReLU}\left( \mathbb{F}_{\text{attack}}(\hat{\boldsymbol{x}})_c - \max_{t \neq c} \mathbb{F}_{\text{attack}}(\hat{\boldsymbol{x}})_i \right) \tag{4-7}$$

$$\mathcal{L}_{\text{fake}} = \mathbb{E}_{\hat{\boldsymbol{x}} \sim T'} \log(\mathbb{D}(\hat{\boldsymbol{x}})) \tag{4-8}$$

$$\mathcal{L}_{\text{hinge}} = \|\boldsymbol{\delta}\|_1 \tag{4-9}$$

$$\mathcal{L}_{\text{recon}} = \frac{1}{n} \sum_{i=1}^{n} |x_i^{(\text{pred})} - x_i| \tag{4-10}$$

其中 $\mathbb{D}$ 是判别器，是一个二分类器，用于判断 $\hat{\boldsymbol{x}}$ 为生成器添加噪声的网络流量（输

出为 $\hat{\boldsymbol{x}}$ 被判别为生成器添加噪声后的网络流量的概率)。

$$\mathcal{L}_{\mathrm{D}} = \mathbb{E}_{\boldsymbol{x}\sim T} \log(\mathbb{D}(\boldsymbol{x})) + \mathbb{E}_{\hat{\boldsymbol{x}}\sim T'} \log(\mathbb{D}(1-\hat{\boldsymbol{x}})) \tag{4-11}$$

保护模型的网络结构定义为第 4.2.2小节说述的注意力编码器 AttEncoder(·)。

# 第 5 章　现有工作基础

- **课程情况**：已完成培养计划里除专业实践、文献综述与选题报告外的所有课程。
- 已完成文献阅读与调研。
- **网络流量分类**：
  - 实现提出的方法的代码。
  - 收集网站识别任务的数据集，处理应用识别任务的数据集。
  - 在两个数据集上进行充分的实验，对超参数和模型架构进行实验。
- **网络流量保护**：
  - 实验提出的方法的大部分代码。
  - 初步实验结果。
- **已接收论文成果**：
  - （清华 A 类期刊学生一作）X. Xiao, W. Xiao, R. Li, X. Luo, H. -T. Zheng and S. -T. Xia, ”EBSNN: Extended Byte Segment Neural Network for Network Traffic Classification,” in IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2021.3101311.

## 5.1　EBSNN 网络流量分类实验验证

为了评估 EBSNN 的性能，我们在本节提供了大型新的真实世界数据集上的详细实验结果。首先，我们对数据集和指标进行了解释。然后，我们分析了我们模型的参数和网络结构的影响。此外，在数据包层面对应用识别任务进行了深入的实验，并详细介绍了与其他方法的比较。网站与应用不同，网站是互联网内容的另一种表现形式，其中包含了更多的噪声流量，因此，网站识别任务在数据包层面进行了彻底的实验，并与其他方法进行了详细的比较。因此，我们进行了网站识别任务，以验证 EBSNN 在该任务上对更多噪声数据的鲁棒性。此外，我们还进行了大量的流量层面的应用识别实验。所有的实验都是在 128GB 内存、Intel Xeon Silver 4210 CPU、NVIDIA 2080Ti GPU 的服务器上进行的。

表 5.1　用应用识别任务的数据集 **D1** 的详情

| ID | 应用 | # 数据包数量 | ID | 应用 | # 数据包数量 |
|---|---|---|---|---|---|
| 1 | Vimeo | 35267 | 16 | iTunes | 18066 |
| 2 | Spotify | 10034 | 17 | Twitter | 9314 |
| 3 | VoipBuster | 360403 | 18 | JD | 19487 |
| 4 | Sina UC | 155951 | 19 | SOHU | 39014 |
| 5 | Netease Cloud Music | 84873 | 20 | Youtube | 40856 |
| 6 | Sina Weibo | 49796 | 21 | Youku | 99909 |
| 7 | Baidu | 36364 | 22 | Netflix | 55189 |
| 8 | Tudou | 89720 | 23 | AIM Chat | 17260 |
| 9 | Amazon | 14530 | 24 | Kugou | 170167 |
| 10 | Thunder | 64325 | 25 | Skype | 632501 |
| 11 | Gmail | 11004 | 26 | Facebook | 245434 |
| 12 | PPLive | 181113 | 27 | Google | 6767 |
| 13 | QQ | 14045 | 28 | MS-SQL | 20882 |
| 14 | Taobao | 31136 | 29 | MS-Exchange | 1720 |
| 15 | Yahoo Mail | 38948 | - | Average | 88071 |

表 5.2　用于网站识别任务的数据集 **D2** 的详情

| ID | 网站 | # 数据包数量 | ID | 网站 | # 数据包数量 |
|---|---|---|---|---|---|
| 1 | JD | 150, 157 | 11 | Instagram | 123, 930 |
| 2 | Netease Cloud Music | 152, 988 | 12 | iQIYI | 193, 280 |
| 3 | TED | 262, 622 | 13 | QQ Mail | 97, 661 |
| 4 | Amazon | 145, 377 | 14 | Reddit | 230, 603 |
| 5 | Baidu | 137, 859 | 15 | Taobao | 227, 034 |
| 6 | Bing | 161, 689 | 16 | Tieba | 125, 235 |
| 7 | Douban | 138, 904 | 17 | Twitter | 229, 780 |
| 8 | Facebook | 249, 179 | 18 | Sina Weibo | 323, 014 |
| 9 | Google | 69, 251 | 19 | Youku | 137, 877 |
| 10 | IMDb | 253, 404 | 20 | Youtube | 315, 101 |

### 5.1.1　数据集

在实验中，使用 D1 和 D2 两个数据集，分别用于应用识别和网站识别两个不同的任务。D1 是一个大型的真实世界的应用识别数据集，由 29 种流行的应用组成。我们的数据集中出现了很多有代表性的应用，包括电子邮件、音乐、视频、社交网络、搜索和购物等。在实际的网络环境中，通过追踪相应的应用进程来捕获特定协议的数据报。具体来说，流量采集器首先获取目标应用的套接字信息，然后对数据包进行过滤，将数据包在真实的网络环境中进行相应的追踪。然后，根据套接字信息对数据报进行过滤和保存。通过这种方式，可以无噪音地收集网络流量。需要注意的是，我们的数据集包括加密和未加密的网络流量数据，这反映了真实的网络环境。每个应用中数据包数量的详细信息可以在表 5.1中找到。从统计信息中可以看出，数据集中各个类的数据包数量差异较大，即类不平衡。这样的类不平衡现象在实际网络场景中经常出现。

络场景中经常出现。另一方面，D2 是另一个用于网站识别的大型真实世界数据集，包含 20 种热门网站。我们在 2020 年通过在隔离环境中运行定制的轻量级浏览器和网络流量转储工具（即 WireShark）来收集 D2。隔离环境是用 Docker 和网络命名空间构建的，其中只运行浏览器和 WireShark。为了尽可能多地收集纯网络流量，自定义浏览器禁用了所有嘈杂的网络流量（如搜索推荐、浏览器更新检查和账户同步）。对于一些必须登录账号的网站，如 QQ 邮箱、微博等，我们手动登录个人账号来获取真实的网络流量。与应用识别相比，网站识别是一项噪音较多的任务，很多网站共享存储在同一个 CDN 的静态文件，表现出相似的行为。具体来说，D2 由网站组成。JD、网易云音乐、TED、亚马逊、百度、必应、豆瓣、Facebook、谷歌、IMDb、Instagram、爱奇艺、QQ 邮箱、Reddit、淘宝、铁霸、Twitter、新浪微博、优酷、Youtube。这些网站几乎涵盖了日常生活的方方面面。大部分流量采用 https 作为传输协议。因此，https 中的数据都是加密的，无法通过目的 IP 和端口进行区分。统计信息如表 5.2所示。

### 5.1.2　网络包级别应用识别任务

在本节中，我们重点关注多类应用识别任务。我们在这部分提供了 EBSNN 的详细结果，并与现有的方法[27,29] 进行了比较。Securitas 是传统的机器学习方法，它在 2016 年被提出，并成为流量分类领域的最佳解决方案之一。另一方面，DeepPacket 是典型的基于深度学习的方法。这两种方法和 EBSNN 都是在数据包层面识别网络流量，即可以识别一个单一数据包的应用。Securitas 是一种有效载荷级的流量分类方法。它使用 LDA 主题模型来提取特定应用的特征，然后再将这些特征分别反馈

| ID | Application | Securitas-C4.5 | | | Securitas-SVM | | | Securitas-Bayes | | | DP-SAE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *R.* | *P.* | $F_1$. | *R.* | *P.* | $F_1$. | *R.* | *P.* | $F_1$. | *R.* | *P.* | $F_1$. |
| 1 | Vimeo | 0.9801 | 0.9875 | 0.9838 | 0.9167 | 0.9625 | 0.9390 | 0.9106 | 0.9675 | 0.9382 | 0.9133 | 0.9562 | 0.9343 |
| 2 | Spotify | 0.9556 | 0.9675 | 0.9615 | 0.8990 | 0.8675 | 0.8830 | 0.8990 | 0.8900 | 0.8945 | 0.7682 | 0.7643 | 0.7662 |
| 3 | VoipBuster | 0.7813 | 0.9825 | 0.8704 | 0.5783 | 0.9975 | 0.7321 | 0.7007 | 0.9950 | 0.8223 | 0.9908 | 0.9931 | 0.9920 |
| 4 | Sina UC | 0.9807 | 0.8875 | 0.9318 | 0.9398 | 0.8200 | 0.8758 | 0.9368 | 0.8525 | 0.8927 | 0.9747 | 0.9945 | 0.9845 |
| 5 | Netease Cloud Music | 0.9490 | 0.9300 | 0.9394 | 0.9968 | 0.7750 | 0.8720 | 0.9742 | 0.8500 | 0.9079 | 0.7252 | 0.9361 | 0.8173 |
| 6 | Sina Weibo | 0.7073 | 0.9425 | 0.8081 | 0.6365 | 0.9675 | 0.7679 | 0.6812 | 0.9350 | 0.7882 | 0.6595 | 0.8009 | 0.7233 |
| 7 | Baidu | 0.6913 | 0.9350 | 0.7949 | 0.6770 | 0.9325 | 0.7844 | 0.6893 | 0.8875 | 0.7760 | 0.6166 | 0.5921 | 0.6041 |
| 8 | Tudou | 0.7255 | 0.9450 | 0.8208 | 0.9427 | 0.5350 | 0.6826 | 0.8960 | 0.5600 | 0.6892 | 0.8518 | 0.8060 | 0.8282 |
| 9 | Amazon | 0.6893 | 0.9375 | 0.7945 | 0.6437 | 0.9350 | 0.7625 | 0.6692 | 0.8800 | 0.7603 | 0.6489 | 0.1889 | 0.2926 |
| 10 | Thunder | 0.7451 | 0.9575 | 0.8381 | 0.8379 | 0.6075 | 0.7043 | 0.7922 | 0.6575 | 0.7186 | 0.7880 | 0.7755 | 0.7817 |
| 11 | Gmail | 0.9281 | 0.6450 | 0.7611 | 0.8621 | 0.5000 | 0.6329 | 0.8258 | 0.5450 | 0.6566 | 0.5475 | 0.1808 | 0.2719 |
| 12 | PPLive | 0.9529 | 0.9100 | 0.9309 | 0.9969 | 0.8000 | 0.8877 | 0.9971 | 0.8625 | 0.9249 | 0.9619 | 0.9354 | 0.9485 |
| 13 | QQ | 0.9016 | 0.6875 | 0.7801 | 0.8773 | 0.5900 | 0.7055 | 0.8796 | 0.6025 | 0.7151 | 0.7227 | 0.3535 | 0.4748 |
| 14 | Taobao | 0.7034 | 0.9250 | 0.7991 | 0.6443 | 0.9600 | 0.7711 | 0.6888 | 0.9350 | 0.7932 | 0.7014 | 0.7132 | 0.7072 |
| 15 | YahooMail | 0.7490 | 0.9400 | 0.8337 | 0.6736 | 0.9650 | 0.7934 | 0.6878 | 0.9475 | 0.7971 | 0.8619 | 0.9241 | 0.8920 |
| 16 | iTunes | 0.7058 | 0.9475 | 0.8090 | 0.6594 | 0.9775 | 0.7875 | 0.6797 | 0.9550 | 0.7942 | 0.6395 | 0.3501 | 0.4525 |
| 17 | Twitter | 0.9705 | 0.9875 | 0.9789 | 0.9459 | 0.9175 | 0.9315 | 0.9397 | 0.9350 | 0.9373 | 0.6448 | 0.8486 | 0.7328 |
| 18 | JD | 0.6861 | 0.9125 | 0.7833 | 0.6562 | 0.9400 | 0.7729 | 0.6904 | 0.8975 | 0.7804 | 0.5623 | 0.1737 | 0.2654 |
| 19 | Sohu | 0.7050 | 0.9200 | 0.7983 | 0.8922 | 0.4550 | 0.6026 | 0.8683 | 0.5275 | 0.6563 | 0.8483 | 0.8531 | 0.8507 |
| 20 | Youtube | 0.9777 | 0.9850 | 0.9813 | 0.9442 | 0.9300 | 0.9370 | 0.9398 | 0.9375 | 0.9387 | 0.9416 | 0.9293 | 0.9354 |
| 21 | Youku | 0.9079 | 0.7150 | 0.8000 | 0.9261 | 0.5950 | 0.7245 | 0.8958 | 0.6450 | 0.7500 | 0.8315 | 0.8916 | 0.8605 |
| 22 | Netflix | 0.9725 | 0.9725 | 0.9725 | 0.9530 | 0.8625 | 0.9055 | 0.9251 | 0.9575 | 0.9410 | 0.9667 | 0.9776 | 0.9721 |
| 23 | Aimchat | 0.7446 | 0.9550 | 0.8368 | 0.9696 | 0.5575 | 0.7079 | 0.9405 | 0.5925 | 0.7270 | 0.7118 | 0.5379 | 0.6128 |
| 24 | Kugou | 0.9860 | 0.8775 | 0.9286 | **1.0000** | 0.8225 | 0.9026 | 0.9941 | 0.8425 | 0.9120 | 0.9926 | 0.9602 | 0.9761 |
| 25 | Skype | 0.7538 | 0.9800 | 0.8522 | 0.9157 | 0.1900 | 0.3147 | 0.7180 | 0.9675 | 0.8243 | 0.9534 | 0.9743 | 0.9637 |
| 26 | Facebook | 0.9975 | 0.9875 | 0.9925 | 0.9921 | 0.9375 | 0.9640 | 0.9874 | 0.9800 | 0.9837 | 0.9762 | 0.9911 | 0.9836 |
| 27 | Google | 0.9850 | 0.9825 | 0.9837 | 0.9475 | 0.9475 | 0.9475 | 0.9478 | 0.9525 | 0.9501 | 0.6290 | 0.3659 | 0.4626 |
| 28 | MS-SQL | 0.9900 | 0.9950 | 0.9925 | 0.9924 | 0.9775 | 0.9849 | 0.9949 | 0.9775 | 0.9861 | 0.9594 | 0.9447 | 0.9520 |
| 29 | MS-Exchange | 0.9813 | 0.7645 | 0.8595 | 0.9881 | 0.7238 | 0.8356 | 0.9806 | 0.7355 | 0.8405 | 0.8342 | 0.4826 | 0.6114 |
| - | Average | 0.8553 | 0.9159 | 0.8765 | 0.8588 | 0.7948 | 0.7970 | 0.8528 | 0.8369 | 0.8309 | 0.8008 | 0.7309 | 0.7466 |
| - | *Accuracy* | | 0.8674 | | | 0.8082 | | | 0.8325 | | | 0.9169 | |

| ID | Application | DP-CNN | | | EBSNN-GRU | | | EBSNN-LSTM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *R.* | *P.* | $F_1$. | *R.* | *P.* | $F_1$. | *R.* | *P.* | $F_1$. |
| 1 | Vimeo | 0.9743 | 0.9820 | 0.9781 | 0.9959 | 0.9967 | 0.9963 | **0.9984** | **0.9991** | **0.9988** |
| 2 | Spotify | 0.9511 | 0.9108 | 0.9305 | 0.9776 | 0.9771 | 0.9773 | **0.9935** | **0.9915** | **0.9925** |
| 3 | VoipBuster | 0.9994 | 0.9974 | 0.9984 | 0.9981 | **0.9998** | 0.9989 | **0.9991** | **0.9998** | **0.9995** |
| 4 | SinaUC | 0.9998 | 0.9992 | 0.9995 | 0.9999 | 0.9999 | 0.9999 | **1.0000** | **1.0000** | **1.0000** |
| 5 | Netease Cloud Music | 0.9785 | 0.9928 | 0.9856 | 0.9985 | 0.9994 | 0.9990 | **0.9999** | **0.9997** | **0.9998** |
| 6 | Sina Weibo | 0.9519 | 0.9796 | 0.9656 | 0.9994 | 0.9969 | 0.9981 | **1.0000** | **0.9998** | **0.9999** |
| 7 | Baidu | 0.9227 | 0.9284 | 0.9255 | 0.9865 | **0.9993** | 0.9929 | **0.9999** | 0.9985 | **0.9992** |
| 8 | Tudou | 0.9880 | 0.9899 | 0.9890 | 0.9985 | 0.9992 | 0.9989 | **0.9996** | **0.9998** | **0.9997** |
| 9 | Amazon | 0.9250 | 0.8737 | 0.8986 | 0.9993 | **1.0000** | 0.9997 | **1.0000** | **1.0000** | **1.0000** |
| 10 | Thunder | 0.9737 | 0.9955 | 0.9845 | 0.9990 | 0.9973 | 0.9981 | **0.9994** | **0.9997** | **0.9995** |
| 11 | Gmail | 0.8653 | 0.9109 | 0.8876 | 0.9886 | 0.9523 | 0.9701 | **0.9900** | **0.9945** | **0.9923** |
| 12 | PPLive | 0.9932 | 0.9829 | 0.9880 | **0.9999** | 0.9972 | 0.9985 | 0.9998 | **0.9999** | **0.9999** |
| 13 | QQ | 0.9535 | 0.9203 | 0.9366 | 0.9897 | 0.9918 | 0.9907 | **0.9982** | **0.9975** | **0.9979** |
| 14 | Taobao | 0.9486 | 0.9107 | 0.9293 | 0.9992 | 0.9994 | 0.9993 | **0.9997** | **0.9997** | **0.9997** |
| 15 | YahooMail | 0.9958 | 0.9854 | 0.9906 | 0.9994 | 0.9992 | 0.9993 | **1.0000** | **1.0000** | **1.0000** |
| 16 | iTunes | 0.9529 | 0.9125 | 0.9323 | 0.9914 | 0.9975 | 0.9944 | **0.9994** | **1.0000** | **0.9997** |
| 17 | Twitter | 0.9726 | 0.9898 | 0.9811 | **0.9995** | 0.9989 | 0.9992 | **0.9995** | **1.0000** | **0.9997** |
| 18 | JD | 0.8763 | 0.9251 | 0.9000 | 0.9985 | 0.9990 | 0.9987 | **0.9992** | **0.9997** | **0.9995** |
| 19 | Sohu | 0.9795 | 0.9759 | 0.9777 | 0.9974 | 0.9911 | 0.9942 | **0.9999** | **0.9986** | **0.9992** |
| 20 | Youtube | 0.9806 | 0.9794 | 0.9800 | 0.9930 | 0.9969 | 0.9950 | **0.9980** | **0.9978** | **0.9979** |
| 21 | Youku | 0.9863 | 0.9873 | 0.9868 | 0.9990 | 0.9982 | 0.9986 | **0.9995** | **0.9999** | **0.9997** |
| 22 | Netflix | 0.9958 | 0.9922 | 0.9940 | 0.9989 | 0.9982 | 0.9985 | **0.9994** | **0.9991** | **0.9992** |
| 23 | Aimchat | 0.9206 | 0.9299 | 0.9252 | 0.9606 | **0.9967** | 0.9783 | **0.9945** | 0.9962 | **0.9954** |
| 24 | Kugou | 0.9982 | 0.9942 | 0.9962 | 0.9996 | 0.9996 | 0.9996 | 0.9999 | **0.9999** | **0.9999** |
| 25 | Skype | 0.9965 | 0.9983 | 0.9974 | 0.9997 | 0.9985 | 0.9991 | **0.9998** | **0.9993** | **0.9996** |
| 26 | Facebook | 0.9994 | 0.9996 | 0.9995 | 0.9998 | 0.9999 | 0.9999 | **0.9999** | **1.0000** | **1.0000** |
| 27 | Google | 0.9856 | 0.9608 | 0.9731 | **0.9978** | **1.0000** | **0.9989** | **0.9978** | **1.0000** | **0.9989** |
| 28 | MS-SQL | 0.9947 | 0.9966 | 0.9957 | 0.9990 | 0.9995 | 0.9993 | **1.0000** | **1.0000** | **1.0000** |
| 29 | MS-Exchange | 0.9788 | 0.9390 | 0.9585 | **0.9971** | 0.9971 | 0.9971 | **0.9971** | **1.0000** | **0.9985** |
| - | Average | 0.9668 | 0.9635 | 0.9650 | 0.9952 | 0.9957 | 0.9954 | **0.9987** | **0.9990** | **0.9988** |
| - | *Accuracy* | | 0.9887 | | | 0.9990 | | | **0.9996** | |

给传统的监督机器学习方法，如 SVM、C4.5 决策树和贝叶斯网络。Securitas 是传统机器学习的经典流量分类方法，它致力于寻找合适的特征来构建机器学习分类器。DeepPacket 利用数据包中的有效载荷内容作为深度学习模型的输入，包括多层感知器、一维 CNN 和堆栈式自动编码器。作为一个利用深度学习从数据包中自动学习有用表示的框架，DeepPacket 开启了在网络流量的分类任务中利用深度学习模型的趋势。

实验结果如上表所示，其中最好的是粗体字。指标 R., P. 和 $F_1$ 分别代表 Recall, Precision, 和 $F_1$ score。Securitas-SVM、Securitas-C4.5 和 Securitas-Bayes 分别指采用 SVM、C4.5 决策树和 Bayes 网络的 Securitas 分类方法。DP-SAE 和 DP-CNN 分别代表堆叠的自回归器和一维 CNN 版本的 DeepPacket。

从上表中可以看出，在 Recall, Precision, 和 $F_1$ score 中，EBSNN-LSTM 几乎实现了所有应用的最佳结果。而其他最好的结果则由 EBSNN-GRU 获得。此外，EBSNN-LSTM 和 EBSNN-GRU 分别取得了第一和第二高的 Accuracy。其中，最高的是 0.9988。总的来说，EBSNN 在所有的指标上都得到了最好的结果（即 Recall, Precision, $F_1$ score, 和 Accuracy），并且优于最先进的方法。在加密流量分类的情况下，深度 LSTM 和 GRU 网络学习和享受非线性变换，这与加密算法中应用的非线性变换类似，特别是块密码算法。我们的方法采用了 LSTM 和 GRU，因此取得了最佳的性能。本实验中 EBSNN-GRU 的得分略低于 EBSNN-LSTM。具体来说，与 EBSNN-LSTM 相比，EBSNN-GRU 只是在百度和 AIM 聊天上得到了更高的分数。同时，EBSNN-GRU 仅在 PPLive 方面获得了更好的分数。但是，在同时考虑到两者的情况下，EBSNN-LSTM 在所有应用中都优于包括 EBSNN-GRU 在内的其他方法。另一方面，EBSNN-GRU 对于 Google 的结果与 EBSNN-LSTM 相同。总的来说，结果表明 LSTM 比 GRU 能更好地捕捉特征模式。其中一个原因是 LSTM 比 GRU 拥有更多的门，它将遗忘门和输入门合并成一个更新门。从不同的应用来看，所有的方法在新浪 UC 和 MS-SQL 上的得分都高于其他应用。

# 第 6 章　预期研究成果

- 完成系统设计及实验测试。
- 总结创新点及实验结果，预期高水平安全方向或网络方向论文一篇，专利一篇。
- 撰写硕士学位论文。

# 第 7 章　研究工作计划

| 时间 | 任务安排 |
| --- | --- |
| 2021.04 ~ 2021.09 | 文献调研 |
| 2021.09 ~ 2022.03 | 复现相关文献，模型算法的初步设计 |
| 2022.03 ~ 2023.01 | 改进模型算法，撰写学术论文，学术论文投稿 |
| 2023.01 ~ 2023.05 | 总结工作成果，硕士学位论文撰写、修改、定稿 |
| 2023.05 ~ 2023.06 | 硕士论文答辩 |

# 参考文献


[1] Saltaformaggio B, Choi H, Johnson K, et al. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic[C/OL]// 10th USENIX Workshop on Offensive Technologies (WOOT 16). Austin, TX: USENIX Association, 2016. https://www.usenix.org/conference/woot16/workshop-program/presentation/saltaformaggio.

[2] Fu Y, Xiong H, Lu X, et al. Service usage classification with encrypted internet traffic in mobile messaging apps[J/OL]. IEEE Transactions on Mobile Computing, 2016, 15(11): 2851-2864. DOI: 10.1109/TMC.2016.2516020.

[3] Taylor V F, Spolaor R, Conti M, et al. Robust smartphone app identification via encrypted network traffic analysis[J/OL]. IEEE Transactions on Information Forensics and Security, 2018, 13(1): 63-78. https://doi.org/10.1109/TIFS.2017.2737970.

[4] Wang P, Ye F, Chen X, et al. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway[J/OL]. IEEE Access, 2018, 6: 55380-55391. https://doi.org/10.1109/ACCESS.2018.2872430.

[5] Xiao X, Li R, Zheng H T, et al. Novel dynamic multiple classification system for network traffic[J/OL]. Information Sciences, 2019, 479: 526 - 541. https://doi.org/10.1016/j.ins.2018.10.039.

[6] Hubballi N, Swarnkar M. *bitcoding* : Network traffic classification through encoded bit level signatures[J/OL]. IEEE/ACM Transactions on Networking, 2018, 26(5): 2334-2346. https://doi.org/10.1109/TNET.2018.2868816.

[7] CNNIC 中国互联网络信息中心. 第 48 次中国互联网络发展状况统计报告 [Z]. 2021.

[8] 北京智游网安科技有限公司（爱加密），移动互联网系统与应用安全国家工程实验室. 全国移动 App 风险监测评估报告》（2021 年 3 季度版）[Z]. 2021.

[9] 中关村在线. 调查显示：大量移动 App 易受恶意软件攻击 [EB/OL]. 2019. https://safe.zol.com.cn/720/7200414.html.

[10] 爱加密. 全国移动 App 安全性研究报告：约 702019. https://www.freebuf.com/articles/paper/202843.html.

[11] 网络安全那点事.《2021 年移动安全报告》恶意软件攻击遍布全球 [EB/OL]. 2021. https://baijiahao.baidu.com/s?id=1697081313109424120&wfr=spider&for=pc.

[12] Luo X, Zhang J, Perdisci R, et al. On the secrecy of spread-spectrum flow watermarks[C]// Gritzalis D, Preneel B, Theoharidou M. Proc. ESORICS. 2010.

[13] Roughan M, Sen S, Spatscheck O, et al. Class-of-service mapping for qos: A statistical signature-based approach to ip traffic classification[C/OL]// Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC '04). New York, NY, USA, 2004: 135-148. https://doi.org/10.1145/1028788.1028805.

[14] Bernaille L, Teixeira R, Salamatian K. Early application identification[C/OL]// Proceedings of

the 2006 ACM CoNEXT Conference (CoNEXT ’ 06). New York, NY, USA, 2006. https: //doi.org/10.1145/1368436.1368445.

[15] Zhang J, Chen C, Xiang Y, et al. An effective network traffic classification method with unknown flow detection[J/OL]. IEEE Transactions on Network and Service Management, 2013, 10(2): 133-147. https://doi.org/10.1109/TNSM.2013.022713.120250.

[16] Sacramento L, Medeiros I, Bota J, et al. Flowhacker: Detecting unknown network attacks in big traffic data using network flows[C/OL]// 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). New York, NY, USA, 2018: 567-572. DOI: 10.1109/TrustCom/BigDataSE.2018.00086.

[17] Chen T, Zhang Y, Li Z, et al. Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum[C]// CCS. 2019.

[18] Huang Y, Wang H, Wu L, et al. Understanding (mis)behavior on the eosio blockchain[C]// ACM SIGMETRICS. 2020.

[19] Shen M, Zhang J, Zhu L, et al. Encrypted traffic classification of decentralized applications on ethereum using feature fusion[C/OL]// Proceedings of the International Symposium on Quality of Service (IWQoS ’ 19). New York, NY, USA, 2019. https://doi.org/10.1145/3326285.3329053.

[20] Chen T, Li Z, Zhu Y, et al. Understanding ethereum via graph analysis[J]. ACM Transactions on Internet Technology (TOIT), 2020.

[21] Kampeas J, Cohen A, Gurewitz O. Traffic classification based on zero-length packets[J/OL]. IEEE Transactions on Network and Service Management, 2018, 15(3): 1049-1062. https://doi.org/10.1109/TNSM.2018.2825881.

[22] Anderson B, McGrew D. Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity[C/OL]// Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’ 17). New York, NY, USA, 2017: 1723-1732. https://doi.org/10.1145/3097983.3098163.

[23] Liu C, Cao Z, Xiong G, et al. Mampf: Encrypted traffic classification based on multi-attribute markov probability fingerprints[C/OL]// 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). Banff, AB, Canada, 2018: 1-10. https://doi.org/10.1109/IWQoS.2018.8624124.

[24] Liu C, He L, Xiong G, et al. Fs-net: A flow sequence network for encrypted traffic classification[C/OL]// IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. Paris, France, France, 2019: 1171-1179. https://doi.org/10.1109/INFOCOM.2019.8737507.

[25] Deri L, Martinelli M, Bujlow T, et al. ndpi: Open-source high-speed deep packet inspection[C/OL]// 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). Nicosia, Cyprus, 2014: 617-622. https://doi.org/10.1109/IWCMC.2014.6906427.

[26] Haffner P, Sen S, Spatscheck O, et al. Acas: Automated construction of application signatures[C/OL]// Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data

(MineNet ’ 05). New York, NY, USA, 2005: 197-202. https://doi.org/10.1145/1080173.1080183.

[27] Yun X, Wang Y, Zhang Y, et al. A semantics-aware approach to the automated network protocol identification[J/OL]. IEEE/ACM Transactions on Networking, 2016, 24(1): 583-595. https://doi.org/10.1109/TNET.2014.2381230.

[28] Luo C, Tan Z, Min G, et al. A novel web attack detection system for internet of things via ensemble classification[J/OL]. IEEE Transactions on Industrial Informatics, 2020: 1-1. DOI: 10.1109/TII.2020.3038761.

[29] Lotfollahi M, Jafari Siavoshani M, Shirali Hossein Zade R, et al. Deep packet: a novel approach for encrypted traffic classification using deep learning[J/OL]. Soft Computing, 2019, 24: 1999-2012. https://doi.org/10.1007/s00500-019-04030-2.

[30] Aceto G, Ciuonzo D, Montieri A, et al. Mimetic: Mobile encrypted traffic classification using multimodal deep learning[J/OL]. Computer Networks, 2019, 165: 106944. https://www.sciencedirect.com/science/article/pii/S1389128619304669. DOI: https://doi.org/10.1016/j.comnet.2019.106944.

[31] Aceto G, Ciuonzo D, Montieri A, et al. Multi-classification approaches for classifying mobile app traffic[J/OL]. Journal of Network and Computer Applications, 2018, 103: 131-145. https://www.sciencedirect.com/science/article/pii/S1084804517303740. DOI: https://doi.org/10.1016/j.jnca.2017.11.007.

[32] Cherubin G, Hayes J, Juarez M. Website fingerprinting defenses at the application layer[J/OL]. Proceedings on Privacy Enhancing Technologies, 2017, 2017(2): 186-203. https://doi.org/10.1515/popets-2017-0023. DOI: doi:10.1515/popets-2017-0023.

[33] De la Cadena W, Mitseva A, Hiller J, et al. Trafficsliver: Fighting website fingerprinting attacks with traffic splitting[C/OL]// CCS ’20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing Machinery, 2020: 1971–1985. https://doi.org/10.1145/3372297.3423351.

[34] Sirinam P, Imani M, Juarez M, et al. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning[C/OL]// Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS ’18). New York, NY, USA: Association for Computing Machinery, 2018: 1928–1943. https://doi.org/10.1145/3243734.3243768.

[35] Bhat S, Lu D, Kwon A, et al. Var-cnn: A data-efficient website fingerprinting attack based on deep learning[J/OL]. Proceedings on Privacy Enhancing Technologies, 2019, 2019(4): 292-310. https://doi.org/10.2478/popets-2019-0070. DOI: doi:10.2478/popets-2019-0070.

[36] Cai X, Nithyanand R, Johnson R. Cs-buflo: A congestion sensitive website fingerprinting defense[C/OL]// WPES ’14: Proceedings of the 13th Workshop on Privacy in the Electronic Society. New York, NY, USA: Association for Computing Machinery, 2014: 121–130. https://doi.org/10.1145/2665943.2665949.

[37] Cai X, Nithyanand R, Wang T, et al. A systematic approach to developing and evaluating website fingerprinting defenses[C/OL]// CCS ’14: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: Association for Computing

Machinery, 2014: 227–238. https://doi.org/10.1145/2660267.2660362.

[38] Juarez M, Imani M, Perry M, et al. Toward an efficient website fingerprinting defense[C]// Askoxylakis I, Ioannidis S, Katsikas S, et al. Computer Security – ESORICS 2016. Cham: Springer International Publishing, 2016: 27-46.

[39] Gong J, Wang T. Zero-delay lightweight defenses against website fingerprinting[C/OL]// 29th USENIX Security Symposium (USENIX Security 20). USENIX Association, 2020: 717-734. https://www.usenix.org/conference/usenixsecurity20/presentation/gong.

[40] Nithyanand R, Cai X, Johnson R. Glove: A bespoke website fingerprinting defense[C/OL]// WPES '14: Proceedings of the 13th Workshop on Privacy in the Electronic Society. New York, NY, USA: Association for Computing Machinery, 2014: 131–134. https://doi.org/10.1145/2665943.2665950.

[41] Wang T, Goldberg I. Walkie-talkie: An efficient defense against passive website fingerprinting attacks[C/OL]// 26th USENIX Security Symposium (USENIX Security 17). Vancouver, BC: USENIX Association, 2017: 1375-1390. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao.

[42] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[Z]. 2015.

[43] Carlini N, Wagner D. Towards evaluating the robustness of neural networks[C/OL]// 2017 IEEE Symposium on Security and Privacy (SP). Los Alamitos, CA, USA: IEEE Computer Society, 2017: 39-57. https://doi.ieeecomputersociety.org/10.1109/SP.2017.49.

[44] Chen P Y, Sharma Y, Zhang H, et al. Ead: Elastic-net attacks to deep neural networks via adversarial examples[J/OL]. Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1). https://ojs.aaai.org/index.php/AAAI/article/view/11302.

[45] Ebrahimi J, Rao A, Lowd D, et al. Hotflip: White-box adversarial examples for text classification[C/OL]// ACL (2). 2018: 31-36. https://aclanthology.info/papers/P18-2006/p18-2006.

[46] Zhang W E, Sheng Q Z, Alhazmi A, et al. Adversarial attacks on deep-learning models in natural language processing: A survey[J/OL]. ACM Trans. Intell. Syst. Technol., 2020, 11(3). https://doi.org/10.1145/3374217.

[47] Grosse K, Papernot N, Manoharan P, et al. Adversarial examples for malware detection[C]// Foley S N, Gollmann D, Snekkenes E. Computer Security – ESORICS 2017. Cham: Springer International Publishing, 2017: 62-79.

[48] Suciu O, Coull S E, Johns J. Exploring adversarial examples in malware detection[C/OL]// 2019 IEEE Security and Privacy Workshops (SPW). 2019: 8-14. DOI: 10.1109/SPW.2019.00015.

[49] Hou C, Gou G, Shi J, et al. Wf-gan: Fighting back against website fingerprinting attack using adversarial learning[C/OL]// 2020 IEEE Symposium on Computers and Communications (ISCC). 2020: 1-7. DOI: 10.1109/ISCC50000.2020.9219593.

[50] Rahman M S, Imani M, Mathews N, et al. Mockingbird: Defending against deep-learning-based website fingerprinting attacks with adversarial traces[J/OL]. IEEE Transactions on Information Forensics and Security, 2021, 16: 1594-1609. DOI: 10.1109/TIFS.2020.3039691.

[51] Nasr M, Bahramali A, Houmansadr A. Defeating dnn-based traffic analysis systems in real-time

with blind adversarial perturbations[C/OL]// 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 2021: 2705-2722. https://www.usenix.org/conference/usenixsecurity21/presentation/nasr.

[52] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J/OL]. Nature, 1986, 323(6088): 533-536. https://doi.org/10.1038/323533a0.

[53] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[54] Cho K, van Merrienboer B, Çaglar Gülçehre, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation[C/OL]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar, 2014: 1724-1734. http://aclweb.org/anthology/D/D14/D14-1179.pdf.

[55] Lin R, Liu S, Yang M, et al. Hierarchical recurrent neural network for document modeling[C/OL]// Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Lisbon, Portugal, 2015: 899-907. https://www.aclweb.org/anthology/D15-1106. DOI: 10.18653/v1/D15-1106.

[56] Lai S, Xu L, Liu K, et al. Recurrent convolutional neural networks for text classification[C]// Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15). Austin, Texas, 2015: 2267–-2273.

[57] Yang Z, Yang D, Dyer C, et al. Hierarchical attention networks for document classification[C/OL]// Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL). San Diego, California, 2016: 1480-1489. https://www.aclweb.org/anthology/N16-1174. DOI: 10.18653/v1/N16-1174.

[58] Li R, Xiao X, Ni S, et al. Byte segment neural network for network traffic classification[C/OL]// 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). Banff, AB, Canada, 2018: 1-10. https://doi.org/10.1109/IWQoS.2018.8624128.

[59] Moore A, Zuev D, Crogan M, et al. Research report (queen mary and westfield college (university of london) department of computer science)): Discriminators for use in flow-based classification[M]. Queen Mary and Westfield College, Department of Computer Science, 2005.

[60] Kingma D P, Ba J. Adam: A method for stochastic optimization[C/OL]// Proceedings of the 3rd International Conference on Learning Representations (ICLR). San Diego, CA, USA, 2015. http://arxiv.org/abs/1412.6980.

[61] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[J/OL]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(2): 318-327. DOI: 10.1109/TPAMI.2018.2858826.

[62] Chen Y, Zang T, Zhang Y, et al. Rethinking encrypted traffic classification: A multi-attribute associated fingerprint approach[C/OL]// 2019 IEEE 27th International Conference on Network Protocols (ICNP). 2019: 1-11. https://doi.org/10.1109/ICNP.2019.8888043.