

## 使用卷积神经网络和迁移学习进行恶意流量的检测

陈浩<sup>1</sup>

<sup>1</sup>(东南大学 网络空间与安全学院, 江苏 南京 210000)

**摘要:** 本文主要是复现论文《Malware traffic classification using convolutional neural network for representation learning》和《TransNet: Unseen Malware Variants Detection Using Deep Transfer Learning》. 论文一使用了一种新的方法来进行恶意流量的检测, 即直接将流量数据视为图像, 使用较为成熟的卷积神经网络来实现端到端的恶意流量检测; 论文二在论文一的基础上, 根据 2019 年发表在 NIPS 上有关迁移学习的论文《Transferable Normalization: Towards Improving Transferability of Deep Neural Networks》, 将 TransNorm 引入恶意流量检测中, 希望实现对未知恶意流量的检测. 本文在复现上述论文的同时, 对其中的一些实验进行了改进, 并讨论了论文中存在的一些问题.

**关键词:** 恶意流量检测; 卷积神经网络; 迁移学习; TransNorm;

**中图法分类号:** TP311

中文引用格式: 陈浩. 使用卷积神经网络和迁移学习进行恶意流量的检测. 软件学报, 2021.

英文引用格式: Chen H. Malware Traffic Classification Using Convolutional Neural Network and Transfer Learning. Ruan Jian Xue Bao/Journal of Software, 2021 (in Chinese).

## Malware Traffic Classification Using Convolutional Neural Network and Transfer Learning

CHEN Hao<sup>1</sup>

<sup>1</sup>(School of Cyberspace and Security, Southeast University, Nanjing 210000, China)

**Abstract:** This article mainly reproduces the papers "Malware traffic classification using convolutional neural network for representation learning" and "TransNet: Unseen Malware Variants Detection Using Deep Transfer Learning". The first paper uses a new method to detect malicious traffic, that is, directly treats the traffic data set as an image, and uses a more mature convolutional neural network to achieve end-to-end malicious traffic detection; the second paper is based on the first paper. According to the paper "Transferable Normalization: Towards Improving Transferability of Deep Neural Networks" published on NIPS in 2019, TransNorm was introduced into malicious traffic detection, hoping to realize the detection of unknown malicious traffic. While reproducing the above papers, this article improves some of the experiments and discusses some problems in the paper.

**Key words:** malware traffic classification; convolutional neural network; transfer learning; TransNorm;

1 相关背景介绍

首先对论文一使用卷积神经网络进行恶意流量检测的相关背景和所使用的基本方法进行简单的介绍.

恶意流量检测任务就是将恶意流量从流量数据集中识别出来,即对流量进行分类.流量分类是将网络流量与生成应用程序关联起来的任务,它是网络管理特别是网络安全领域中至关重要的任务.在网络安全领域,流量分类实际上代表了对网络资源恶意使用的异常检测等活动的第一步[1].

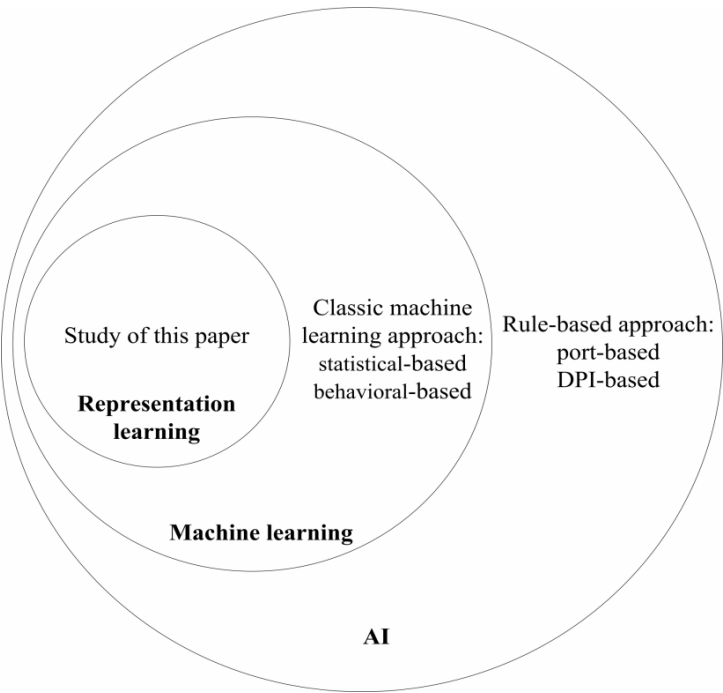


Fig.1 AI-based traffic classification method

图 1 基于 AI 的流量分类方法

常用的流量分类主要有四种方法<sup>[1]</sup>:基于端口的、深度数据包检查(DPI)、基于统计的、基于行为的.从人工智能的角度来看<sup>[2]</sup>,基于端口和基于 DPI 的方法是基于规则的方法,它通过匹配预定义的硬编码规则来执行流量分类.基于统计和行为的方法是典型的机器学习方法,它通过使用一组选择性特征从经验数据中提取模式来对流量进行分类.虽然经典的机器学习方法解决了许多基于规则的方法无法解决的问题,例如加密的流量分类和高计算成本,但是它面临着设计适当特性的新挑战<sup>[3]</sup>.

表征学习是近年来快速发展的一种快速发展的机器学习方法,它从原始数据中自动学习特征,在一定程度上解决了手工设计特征的问题<sup>[4]</sup>.特别是在图像分类和语音识别等多个领域,深度学习方法是表征学习的一种典型方法,取得了很好的效果<sup>[5][6]</sup>.

因此论文一尝试将表征学习应用于恶意软件流量分类领域,并证明其有效性.图 1 展示了人工智能视角下的流量分类分类.Fig.2 AI-based traffic classification workflow

图 2 显示了这些方法的不同工作流程,阴影框表示能够从数据中学习的组件<sup>[2]</sup>.

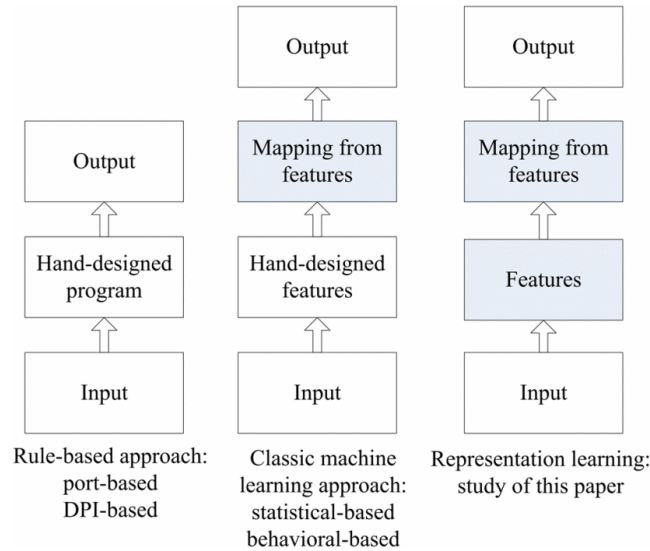


Fig.2 AI-based traffic classification workflow

图 2 基于 AI 的流量分类 workflow

论文一所使用的技术是卷积神经网络(CNN),它是目前最流行的表征学习方法之一.其没有从流量中提取特征,而是将原始的流量数据作为图像,然后使用 CNN 进行图像分类,这是 CNN 最常见的任务<sup>[7] [8]</sup>,以此实现恶意软件流量检测的目标.作者在论文中讲到,据他所知,这一有趣的尝试是第一个使用原始流量数据的表征学习方法在恶意软件流量分类领域的应用.

由于流量数据的连续性和图像数据的离散性的不同,论文一还研究了多种流量表示类型,并通过实验找到了最佳类型.为了证明论文中所提出的方法的可扩展性,其使用三种类型的分类器进行两种场景的实验,最终的平均精度为 99.41%.

## 2 数据处理及实验方法

### 2.1 数据集

寻找可用数据集并不是一件简单的事情,正如 Dainotti 等<sup>[11]</sup>所指出的那样,缺乏各种可共享的跟踪数据集作为测试数据是在流量分类上取得进展的最明显的障碍.许多关于恶意软件流量分类的研究使用了自收集的流量或安全公司的私人流量,从而破坏了其结果的可信度.由于经典的机器学习方法侧重于特征选择技术,许多当前的公开流量数据集都是流特征数据集,而不是原始的流量数据集<sup>[9][12]</sup>.这些数据集不能满足我们对原始流量的要求.在提供原始流量的数据集中,几乎没有包含足够的正常和恶意软件流量数据,比如<sup>[13]</sup>.

为了解决这些问题,论文一创建了一个数据集 USTC-TFC2016.数据集由两部分组成,如 Table 1 USTC-TFC2016 Data Set - Malicious Traffic

表 1 和 Table 2 USTC-TFC2016 Data Set - Benign Traffic

表 2 所示.第一部分为从 2011 年到 2015 年 CTU 研究人员从真实的网络环境中收集到的 10 种恶意软件流量<sup>[14]</sup>.对于一些太大的流量,只使用了一部分.对于一些规模过小的流量,我们将它们合并并在同一个应用程序中.第二部分包含了使用 IXIA BPS<sup>[15]</sup>采集的 10 种正常流量,这是一种专业的网络交通仿真设备.关于模拟方法的信息可以在他们的产品网站上找到.为了尽可能地反映更多的流量,十种流量包含八个类的通用应用程序.USTC-TFC2016 数据集的大小为 3.71GB,格式为 PCAP.

Table 1 USTC-TFC2016 Data Set - Malicious Traffic

表 1 USTC-TFC2016 数据集-恶意流量

Name	CTU num	Binary MD5	process
Cridex	108-1	25b8631afeea279ac00b2da70fffe18a	original
Geodo	119-2	306573e52008779a0801a25fafb18101	part
Htbot	110-1	e515267ba19417974a63b51e4f7dd9e9	original
Miuref	127-1	a41d395286deb113e17bd3f4b69ec182	original
Neris	42,43	bf08e6b02e00d2bc6dd493e93e69872f	merged
Nsis-ay	53	eaf85db9898d3c9101fd5fcfa4ac80e4	original
Shifu	142-1	b9bc3f1b2aace824482c10ffa422f78b	part
Tinba	150-1	e9718e38e35ca31c6bc0281cb4ecfae8	part
Virut	54	85f9a5247afbe51e64794193f1dd72eb	original
Zeus	116-2	8df6603d7cbc2fd5862b14377582d46	original

**Table 2** USTC-TFC2016 Data Set - Benign Traffic

表 2 USTC-TFC2016 数据集-正常流量

Name	Class	Name	Class
BitTorrent	P2P	Outlook	Email/WebMail
Facetime	Voice/Video	Skype	Chat/IM
FTP	Data Transfer	SMB	Data Transfer
Gmail	Email/WebMail	Weibo	Social NetWork
MySQL	Database	WorldOfWarcraft	Game

我们在作者的 github 仓库中找到其创建的数据集 USTC-TFC2016,并下载到本地.经过检查,和论文中列出的属性完全相同.Github 链接为 : [https://github.com/echowei/DeepTraffic/tree/master/1.malware\\_traffic\\_classification](https://github.com/echowei/DeepTraffic/tree/master/1.malware_traffic_classification).

## 2.2 网络流量表示

下面是论文中关于网络流量划分的叙述,其共提出了四种划分方法,在数据包的层级上划分出全部层(ALL)和 L7 层,在流量粒度上划分出 Session 和 Flow,因此四种类型的流量表示为: Flow+ All、Flow+ L7、Session+ All、Session+ L7.论文进行了相关的实验来探讨不同的流量划分方法对流量分类结果的影响,从中找到最佳的网络流量划分方法.

## 2.3 流量粒度

网络流量分割粒度包括:TCP 连接、流 (Flow)、会话 (Session)、服务和主机<sup>[11]</sup>.不同的分割粒度会导致不同的流量单元.我们的方法使用的是流和会话,这也是大多数研究人员使用的方法.流被定义为具有相同 5 元组的所有包,即源 IP、源端口、目的地 IP、目的端口和传输层协议.会话被定义为双向流,包括流量的两个方向.正式描述如下:

(1) 原始流量: 所有数据包都定义为一组  $P = \{p^1, p^2, \dots, p^{|p|}\}$ .每个数据包定义为  $p^i = (x^i, b^i, t^i)$ ,  $i = 1, 2, \dots, |p|$ ; 元素  $x^i$  代表五元组,元素  $b^i \in [0, \infty)$  代表数据包的字节大小; 元素  $t^i \in [0, \infty)$  代表流量的开始时间.

(2) 流: 一组原始流量  $P$  可以分为多个子集.子集中的所有分组按时间顺序排列,即  $\{p^i = (x^i, b^i, t^i), \dots, p^n = (x^n, b^n, t^n)\}$ ,  $t^1 < t^2 < \dots < t^n$ .子集定义为流  $f = (x, b, d_t, t)$ .元素  $x$  代表相同的五元组,即  $x_1 = x_2 = \dots = x_n$ .元素  $b$  代表流中所有数据包大小的和.元素  $d_t$  代表流的持续时间.最后一个元素是第一个数据包传输的开始时间.原始流量可以转化为流  $F = \{f_1, f_2, \dots, f_n\}$ .

(3) 会话: 会话包括两个方向的流,即源 IP 和目标 IP/端口是可互换的.不同的流量或会话可能有不同的大小,但是 CNN 的输入数据大小必须是一致的,所以只使用每个流量或会话的前  $n$  个字节(本文中的  $n=784$ ).我们可以对这个选择给出一个直观的解释.通常,流或会话的前面部分通常是连接数据和一些内容数据,应该最好地反映流或会话的固有特性.这一选择与<sup>[16][17]</sup>非常相似,它研究了典型的机器学习方法的恶意软件流量识别.此外,由于只使用了前数百个字节,所以这种方法比许多基于规则的方法更轻量级.

从数据包层级分析的角度来看,固有特性应该反映在 TCP / IP 模型的应用层,即 OSI 模型的第 7 层.例如,SMTP 协议代表电子邮件通信,HTTP 协议代表浏览器流量.基于这个假设,Wang<sup>[10]</sup>只选择了第 7 层,并将其称为 TCP 会话有效负载.另一方面,其他层的数据也应该包含一些流量特征信息.例如,传输层中的端口信息

可以使用标准端口号识别大多数应用程序.有时,标记信息可以识别网络攻击,例如 SYN 攻击和 RST 攻击.因此,使用了两种类型的包层:所有层(全部)和第 7 层(L7).应该注意的是,会话或流中的 IP 和 MAC 信息可能会损坏特征提取过程.为了消除这种负面影响,我们必须通过随机化(通常被称为流量匿名化或净化处理<sup>[20]</sup>)来移除这些信息.

综上所述,有四种类型的流量表示:Flow+ All、Flow + L7、Session + All、Session + L7.他们的性能测试使用了 2.1 节介绍的两种类型的流量数据集,最后确定了最佳的表示类型.

## 2.4 数据预处理

在讨论完网络流量的划分方法后,便开始进行流量数据的处理工作.主要就是将下载得到的 Pcap 文件格式的原始流量划分成不同表示方法的 IDX 文件.在作者的 GitHub 仓库里,作者一同提供了数据处理的工具箱,但是在使用这个工具箱进行数据处理时遇到了一些问题,在对代码进行修复之后,成功按照论文里提出的方法完成了数据流量的处理.下面是详细的流量处理流程:

数据预处理是将原始流量数据(pcap 格式)转换为 CNN 输入数据(IDX 格式)的过程.它包括四个步骤:流量分割、流量清除、图像生成和 IDX 转换. Fig.3 Data preprocessing process

图 3 显示了数据预处理的全过程.

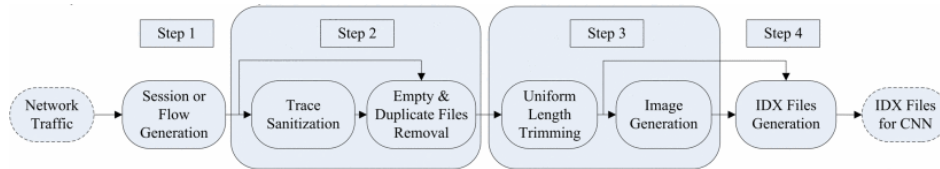


Fig.3 Data preprocessing process

图 3 数据预处理流程

步骤 1(流量分割).这一步分割了连续的原始流量,以增加离散的流量单元.输入数据格式为 Pcap.如果表示类型是 Flow + All 或 Session + All,则输出数据格式为 Pcap.如果表示类型为 Flow + L7 或会话 + L7,则输出数据格式为 Bin.

步骤 2(流量清洗).该步骤首先执行流量匿名化/清洗,在数据链路层和 IP 层分别随机化 MAC 地址和 IP 地址.这是可选的,例如,当所有流量来自同一网络时,MAC 和 IP 可能不再是区分信息,在这种情况下,我们不需要执行它.流量清洗的第二个作用是清除文件.有些包没有应用程序层,所以结果 bin 文件是空的.有些包在具有相同内容时生成相同的文件,而重复的数据在训练 CNN 时可能导致偏差.那些空的和重复的文件需要删除.此步骤中的数据格式没有变化.

步骤 3(图像生成).这一步首先将所有文件整理成均匀长度.如果文件大小大于 784 字节,则将其裁剪为 784 个字节.如果文件大小小于 784 字节,则在末尾添加 0x00 以补充到 784 字节.使用相同大小的结果文件被转换为灰色图像.原始文件的每个字节表示一个像素,例如,0x00 是黑色的,0xff 是白色的.实际上,这种转换是可选的,我们可以直接将文件转换为 IDX 文件.这种转换只是为了向研究人员展示视觉图像,所以他们可以以视觉的方式进行分析.

步骤 4(IDX 转换).此步骤将图像转换为 IDX 格式文件.IDX 文件包含一组图像的所有像素和统计信息.IDX 格式是机器学习领域常见的文件格式<sup>[18][19]</sup>.

USTC-TFC2016 流量数据集使用 USTC-TK2016 工具包进行处理,共生成 752,040 条记录.表三显示了结果.因为会话包括双向流,会话的数量通常少于流.

Dataset	Representation	Count Range	Count Total
Malware	Flow+ALL	6000~17178	134563
	Flow+L7	4569~13968	76716
	Session+ALL	6000~8629	71008
	Session+L7	4569~7592	63120
	<b>Total</b>	——	<b>406633</b>
Benign	Flow+ALL	10051~17008	138145
	Flow+L7	8908~16391	126665
	Session+ALL	5134~9634	71692
	Session+L7	4952~9576	70131
	<b>Total</b>	——	<b>345407</b>
<b>Total</b>	——	——	<b>752040</b>

Fig.4 USTC-TFC2016 data set processing results

图 4 USTC-TFC2016 数据集处理结果

2.5 数据可视化分析

由于论文一是将流量数据集视为图像,利用卷积神经网络的方式进行处理.在进行流量分类前我们可以通过直接观察的方式来查看图像形式的流量数据集的特点.

在此部分分析了数据预处理过程步骤 3 后生成的图像.每个灰色图像的大小为 784(28\*28)字节.Session +all 的可视化结果如 Fig.5

图 5 和 Fig.6 Similarity of similar traffic

图 6 所示.其他三种表示形式的结果通常与它们相似.

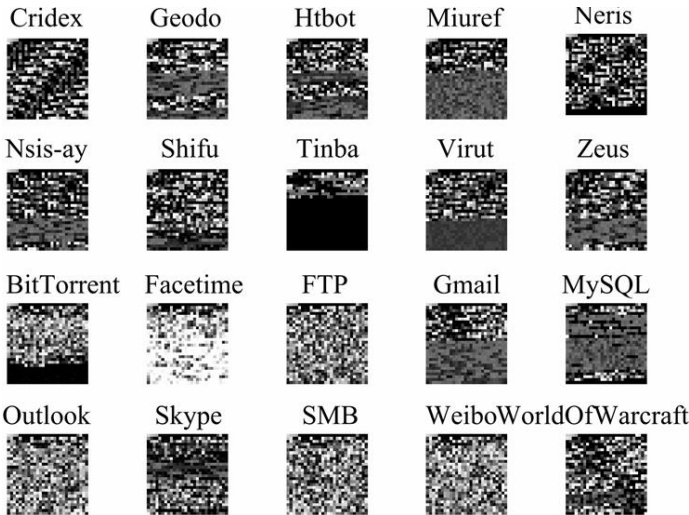


Fig.5 Visualized results of various types of traffic

图 5 各类流量的可视化结果

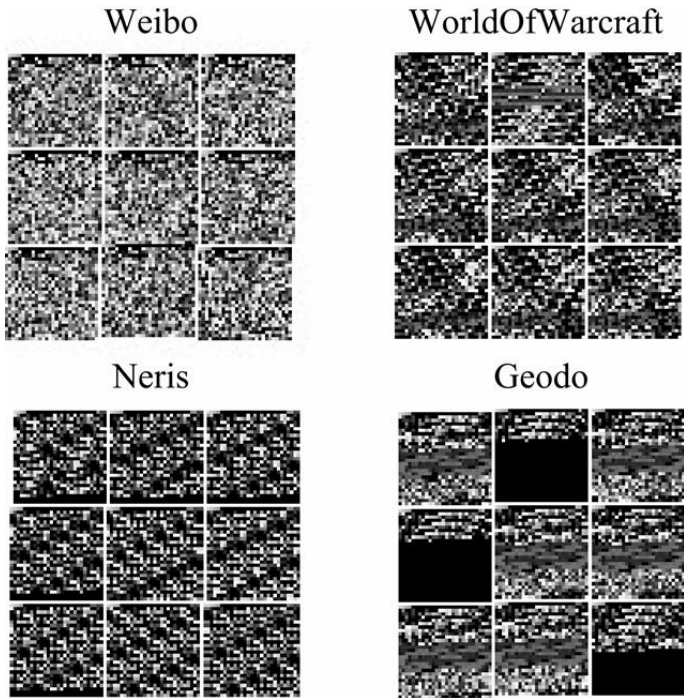


Fig.6 Similarity of similar traffic

图 6 同类流量的相似性

Fig.5 Visualized results of various types of traffic

图 5 显示了所有类的可视化结果.很明显,他们很容易区分.只有很少的图像是非常相似的,例如 FTP 和 SMB.Fig.6 Similarity of similar traffic

图 6 显示了同一流量类中的一致性.在一个类中随机选取 9 个图像,随机选取 4 个类.同样明显的是,微博、魔兽和 Neris 的图像纹理非常相似.只有 Geodo 图像可以被划分为两个子类,但即使在各自的子类中图像的纹理仍然非常相似.其他 16 个类的一致性大体相似.通过可视化分析,我们得出结论:不同的流量类别具有明显的歧视程度,每一类流量都具有高度的一致性,因此我们有理由认为我们的方法可以取得良好的效果.

## 2.6 CNN 架构

在完成数据集的处理之后,需要开始构建 CNN 卷积神经网络用于流量分类.整体 CNN 架构是比较常见的架构,下面的具体的网络架构:

预处理数据集的大小和每个图像的大小都非常类似于经典的 MNIST 数据集<sup>[19]</sup>,因此使用了类似于 LeNet-5<sup>[20]</sup>的 CNN 体系结构,但是在每个卷积层中使用了更多的通道.

CNN 首先从 IDX 文件中读取  $28 \times 28 \times 1$  的流量图像.这些图像的像素从  $[0, 255]$  被归一化到  $[0, 1]$ .第一个卷积层 C1 执行一个卷积运算,有 32 个大小为  $5 \times 5$  的内核.C1 层的结果为 32 个特征图,大小为  $28 \times 28$ .在 C1 层之后,在 P1 层中有一个  $2 \times 2$  的 max 池操作,结果是 32 个特征图,大小为  $14 \times 14$ .第二个卷积层 C2 的内核大小也是  $5 \times 5$ ,但有 64 个通道.结果是 64 个尺寸  $14 \times 14$  的特征图.在第二个  $2 \times 2$  的 max 池层 P2 之后,生成了 64 个大小为  $7 \times 7$  的 feature map.最后两个层是完全连接的,输出大小分别为 1024 和 10,并使用 softmax 函数用来输出每个类的概率.此外,dropout 是用来缓解过度拟合的.后续的实验探究都使用了这种 CNN 架构.

## 2.7 可拓展性研究

为了探究论文所提出方法的通用性,作者设计了两种应用场景,场景 A 为 2 分类、10 分类、场景 B 为 20 分类,接着实验探究每种场景下所设计网络结构的性能.

为了验证本文提出方法的可拓展性,设计两种应用场景,包括了三种类型的 CNN 分类器,分别是: 2 分类分类器,10 分类分类器,20 分类分类器.Fig.7 Three different types of experimental scenarios

图 7 显示了两个场景的实验设置.

场景 A 包含 2 分类分类器和 10 分类分类器.在实际应用的流量分类中,最常见的要求是恶意软件的流量识别,这也是 NIDS 的主要目标.如果必要的话,可以进行更精细的流量分类来识别每一类恶意软件和正常的流量.在这种情况下,首先执行一个二进制分类来识别恶意软件或正常,然后分别执行两个 10 级分类来识别每一类流量.场景 B 包含 10 分类和 20 分类分类器.在某些应用中,我们需要对所有类型的流量进行一次分类,这就要求对分类器进行比较高的性能.使用所有类型的流量,进行 20 分类实验.

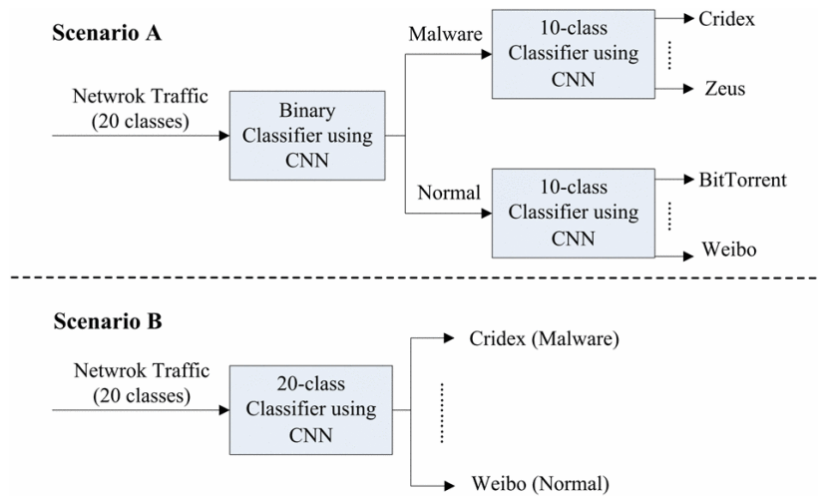


Fig.7 Three different types of experimental scenarios

图 7 三类不同的实验场景

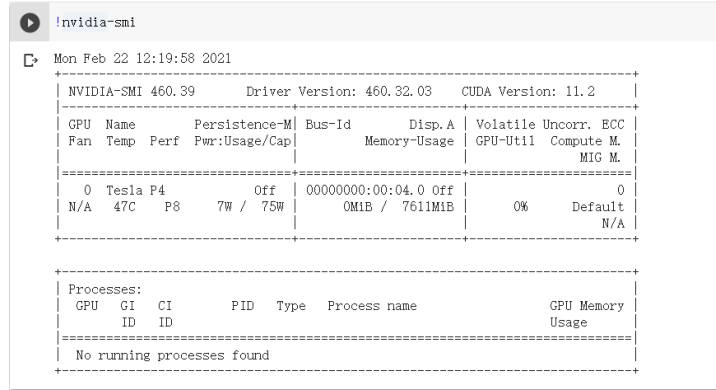


### 3、评估

#### 3.1 实验设置

论文一使用了 TensorFlow 用作实验软件框架,我们开始也使用了 TensorFlow,不过在复现论文二的时候,需要修改底层代码,Pytorch 修改起来更加方便一些.为了统一,我们在复现论文一和论文二全都使用 Pytorch 作为软件框架.虽然所设计的 CNN 框架并不算很复杂,但是依然需要使用 GPU 进行加速,为此我们的整个实验在 Google 的 Colab 上面进行.Fig.8 Details of the GPU used in the experiment

图 8 显示了详细的 GPU 信息,其他 CNN 架构的实验参数设计都与论文中设计的相同,不过在复现过程中我们也尝试了参数的改变对实验结果的影响.



```
Mon Feb 22 12:19:58 2021

+-----+
| NVIDIA-SMI 460.39          | Driver Version: 460.32.03 | CUDA Version: 11.2 |
+-----+-----+
| GPU  Name      Persistence-M | Bus-Id        Disp.A    | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|=====-=+=====+=====+=====+=====+=====+=====+
| 0   Tesla P4     Off          | 00000000:00:04:0 Off    |      0%      Default  |
| N/A   47C    P8      7W / 75W |  0MiB / 16111MiB |             MIG M.   |
+-----+-----+

+-----+
| Processes:                 |
| GPU  GI  CI       PID    Type   Process name          GPU Memory |
| ID   ID  ID               |              | Usage      |
+-----+-----+
| No running processes found |
+-----+
```

Fig.8 Details of the GPU used in the experiment

图 8 实验使用的 GPU 的详细信息

TensorFlow<sup>[21]</sup>被用作实验软件框架,运行在 Ubuntu 14.04 64 位操作系统上.服务器是戴尔 R720,有 16 核 CPU 和 16GB 内存.一个 Nvidia Tesla K40m GPU 被用作加速器.有十分之一的数据被随机选择作为测试数据,其余的是训练数据.Batch size 是 50,损失函数是交叉熵函数.在 TensorFlow 中构建的梯度下降优化器被用作优化器.学习率为 0.001,一共训练约 40 代.

#### 3.2 评价指标

使用了四个评价指标:准确性(A)、精确性(P)、召回率(R)、f1 值(f1).准确性用于评价分类器的整体性能.精确性 (P)、召回率 (R) 和 f1 值被用来评估分类器对于每种流量类型的分类性能.

$$A = \frac{TP + TN}{TP + FP + FN + TN}, P = \frac{TP}{TP + FP} \quad (1)$$

$$R = \frac{TP}{TP + FN}, F_1 = \frac{2PR}{P + R} \quad (2)$$

TP 是正确分类 X 的实例数,TN 是正确分类非 X 的实例数,FP 是错误分类 X 的实例数,而 FN 是错误分类非 X 的实例数.

#### 3.3 实验结果和分析

为了探究论文中设计检测方法的性能,同时探究流量数据不同表示方法对结果的影响,作者设计了一系列的实验,实验结果如下.

本文进行了八项实验以确定最佳的流量表示类型.Fig.9 Classification accuracy of malicious traffic and normal traffic

图 9 显示了恶意软件流量和正常流量数据集四种表示类型的 10 分类准确性.从结果中可以发现,使用所有层表示的流量的准确性总是高于 L7 层.除了在正常流量中 L7 的 Session 和 Flow 的准确性是相等的,其他三个比较都表明,使用 Session 表示的流量的准确性总是高于使用 Flow.

Fig.9 Classification accuracy of malicious traffic and normal traffic

图 9 显示了恶意流量 Nsis -ay 的精确率、召回率和 f1 值,它是 20 种流量类型之一.从结果可以看出和 Fig.9



### Classification accuracy of malicious traffic and normal traffic

图 9 相似的结论.在所有的 12 个比较中,除了使用所有层和 Session 表示的召回率比使用所有层和 Flow 表示的稍微低一点(0.24%),其他 11 个比较均显示了如下模式:所有层表示的流量的精确率、召回率、F1 值都比使用 L7 层的高,而且使用 Session 的精度、召回率、F1 值都比使用 Flow 要高.不仅如此,同样的模式也可以在其他 19 类流量结果中找到.总之,使用所有层表示的都比使用 L7 表示好,而且使用 Session 表示的比使用 Flow 表示的更好.

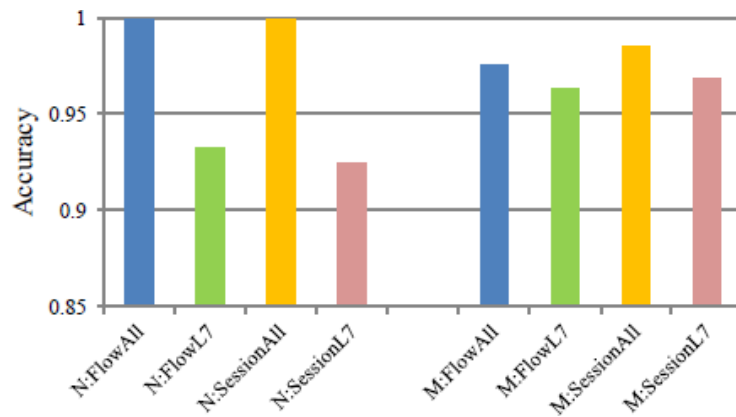


Fig.9 Classification accuracy of malicious traffic and normal traffic

图 9 恶意流量和正常流量分类准确率

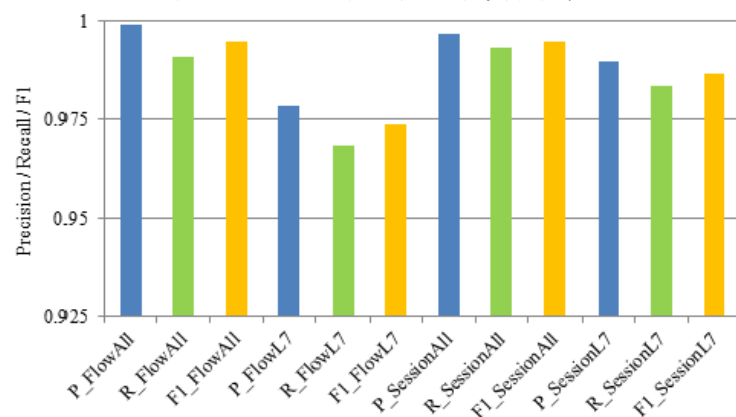


Fig.10 Nsis-ay traffic classification of P, R, and F1

图 10 Nsis-ay 流量分类的 P,R,及 F1

对于这种结果,可以给出一个直观的解释.因为会话包含双向流,而不是单向流,因此包含更多的交互信息.因为所有层表示的流包含了比使用 L7 层更多的层,特别是包含端口和标记信息,所以它可以代表更多的关键信息,这证明了我们在 2.2 节的假设.需要注意的是,许多类流量的 L7 层预处理结果记录要比所有层的预处理结果记录少得多,例如 CTU 107-1 的 L7 层记录只有 8,但所有层记录都是 16386.当生成相同数量的训练数据时,使用 L7 层表示需要使用比所有层表示更多的流量数据.因此,我们可以看到,使用所有层进行表示比使用 L7 层进行表示更加灵活.

### 3.4 复现论文一的实验结果

我们对论文一的实验进行了复现,在设置相同参数,训练 40 代时,部分实验并没有达到最佳的性能,以实验 N:FlowL7 为例,训练 40 代的实验结果如 Fig.11 N: FlowL7 experimental training 40epoch results 图 11 所示.

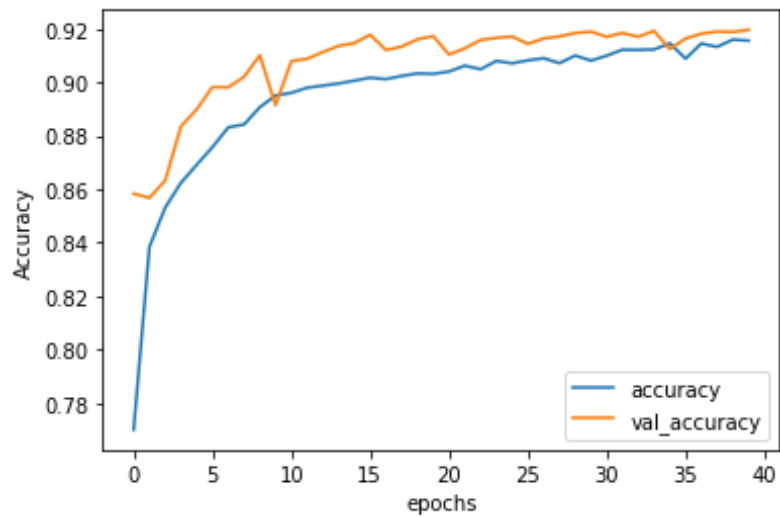


Fig.11 N: FlowL7 experimental training 40epoch results

图 11 N:FlowL7 实验训练 40epoch 结果

为此我们将实验的 epochs 参数调整为 80, 这样能保证分类器得到充分的训练. Fig. 12 Classification accuracy of malicious traffic and normal traffic

图 12 和 Fig. 13 Nsis-ay traffic classification of P, R, and F1

图 13 我们对论文一中实验的复习结果, 基本和论文中提到的一致.

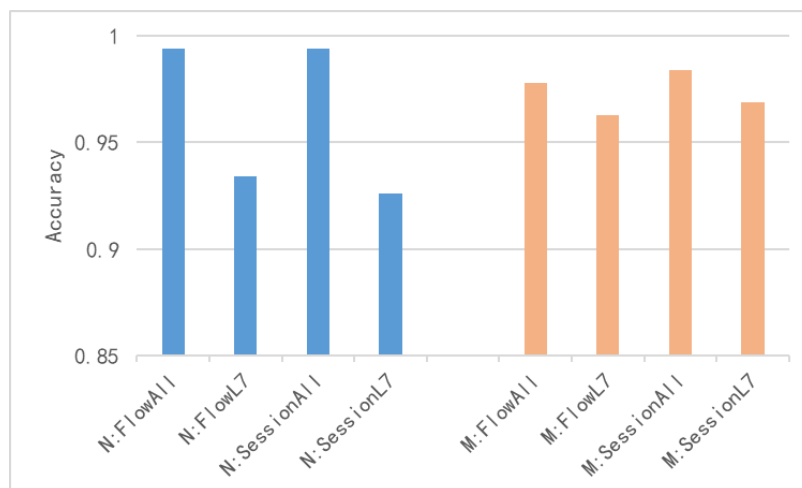


Fig.12 Classification accuracy of malicious traffic and normal traffic

图 12 恶意流量和正常流量分类准确率

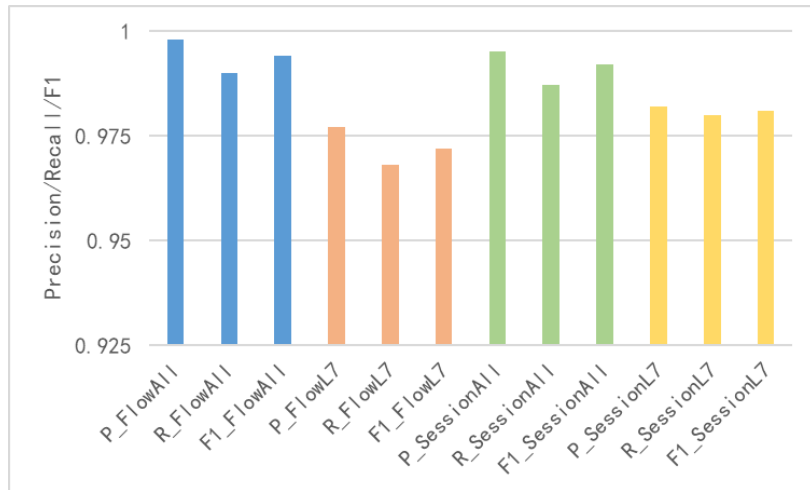


Fig.13 Nsis-ay traffic classification of P, R, and F1

图 13 Nsis-ay 流量分类的 P,R,及 F1

### 3.5 可拓展性实验结果和分析

通过上述的实验确定了使用 FlowAll 进行表示流量最好,接着使用 FlowAll 表示的流量用于确定不同类型分类器的详细性能.Fig.14 The accuracy of different classifiers

图 14、Fig.15 P, R, F1 of each flow in the 20-classification

图 15、Fig.16 P, R, F1 of each flow in the 10-classification

图 16、显示了分类器的具体性能,从图上可以看出论文所提出的分类器性能准确率指标都很高,满足实际的需求.我们同样使用数据复现了相关的实验,结果和论文中显示的基本完全相同.

Fig.14 The accuracy of different classifiers

图 14 显示了使用会话+所有层流量表示的两种场景中三种类型的分类器的总体准确性.Fig.15 P, R, F1 of each flow in the 20-classification

图 15 和 Fig.16 P, R, F1 of each flow in the 10-classification

图 16 显示了每一类流量的精度、召回率和 F1 值.由于 2 分类分类器的精度为 100%,所以不需要显示二进制分类器的精度、召回率和 F1 值.

Metric	Average	Binary Classifier	10-class Classifier		20-class Classifier
			Normal	Malware	
Accuracy	99.41	100	99.94	98.52	99.17

Fig.14 The accuracy of different classifiers

图 14 不同分类器的准确率

DATA	PR	RC	F1	DATA	PR	RC	F1
Cridex	100	100	100	BitTtr	100	100	100
Geodo	100	99.9	99.9	Facetime	100	100	100
Htbot	99.8	100	99.9	FTP	100	100	100
Miuref	100	100	100	Gmail	98.4	99.3	98.8
Neris	96.3	92.9	94.6	MySQL	100	100	100
Nsis-ay	99.8	99.0	99.4	Outlook	99.2	98.1	98.7
Shifu	99.9	99.9	99.9	Skype	99.8	100	99.9
Tinba	100	100	100	SMB	100	100	100
Virut	90.7	95.6	93.1	Weibo	100	100	100
Zeus	100	100	100	Wow	100	99.9	99.9

Fig.15 P, R, F1 of each flow in the 20-classification

图 15 20-分类中各流量的 P,R,F1

DATA	PR	RC	F1	DATA	PR	RC	F1
Cridex	100	100	100	BitTrr	100	100	100
Geodo	100	100	100	Facetime	100	100	100
Htbot	99.8	99.8	99.8	FTP	100	100	100
Miuref	100	100	100	Gmail	99.9	99.7	99.8
Neris	97.1	91.1	94.0	MySQL	100	100	100
Nsis-ay	99.7	99.3	99.5	Outlook	99.6	99.9	99.7
Shifu	99.9	99.8	99.8	Skype	100	100	100
Tinba	99.9	100	99.9	SMB	100	100	100
Virut	89.0	96.5	92.6	Weibo	100	100	100
Zeus	100	100	100	Wow	100	100	100

Fig.16 P, R, F1 of each flow in the 10-classification

图 16 10-分类中各流量的 P,R,F1

Fig.14 The accuracy of different classifiers

图 14 显示四种分类器的准确度都很高,即使是最低准确度也达到了 98.52%。Fig.15 P, R, F1 of each flow in the 20-classification

图 15 显示 Neris 和 Virut 流量的 P、R 和 F1 值稍低(90%~96%),但其他 18 类流量均达到非常高的 P、R 和 F1 值(高于 99%)。Fig.16 P, R, F1 of each flow in the 10-classification

图 16 显示了相似的模式,Neris 和 Virut 流量的指标值稍低一些(89%~97%),但是其他 18 个级别的流量都达到了很高的指标(高于 99%)。为什么 Neris 和 Virut 会出现一些较低的指标,这可能是它们的特殊特征,需要进一步研究。综上所述,这三种分类器的平均准确度已经达到了 99.41%,满足了实际使用的准确度要求。

#### 4、局限性和未来的工作

论文一作者在最后讨论分析了其工作的一些局限性和未来将要进行的工作,在完成论文一的复现之后,考虑到实际应用的需求,最重要的是需求增加分类器对未知恶意流量的检测能力。为此我们复现了论文二《TransNet: Unseen Malware Variants Detection Using Deep Transfer Learning》,探讨如何检测未知的恶意流量。在接下来的部分我们将详细探讨具体的实现方法和细节。

关于我们的工作有三个限制和相关的未来工作。首先,本文的主要目的是利用表示学习方法来证明恶意软件流量分类的有效性,因此我们没有研究 CNN 参数的调优。在实际应用中,流量大小和类数肯定不是固定的,所以在机器学习术语中,我们的方法的泛化能力需要进一步验证。其次,传统的流量分类方法采用经典的机器学习方法,使用了许多时间序列特征,证明了它们的高效率<sup>[22]</sup>。我们的方法实际上只使用了流量的空间特性,并且完全忽略了时间特性。我们将研究如何在未来的工作中加入这些时间特征,例如使用递归神经网络(RNN)。最后,本文的工作只对已知的恶意软件流量进行分类。识别未知恶意软件流量的能力对于 NIDS<sup>[23]</sup>来说也是非常重要的,如何在未来的工作中进一步研究如何增加未知的恶意软件流量的能力。

#### 5、未知恶意流量的检测

##### 5.1 背景

Internet 上恶意软件的数量和种类不断增加,给互连的网络社区提出了严峻的挑战。看不见的恶意软件变体的出现导致训练和测试数据集中特征和标签的分布不同。对于广泛使用的基于机器学习的检测方法,数据集偏移问题将使训练后的模型在面对新数据时失效。但是,要重新学习描述新数据的功能还是收集大量带标签的样本以重新训练分类器,这是一项艰巨而繁琐的工作。

迁移学习是一种新颖的机器学习技术,旨在培养有效的学习器,他们可以利用源域中丰富的标记数据知识,并将其转移到缺少标记数据的目标域中,以减少跨域数据的分布变化。

##### 5.2 问题描述

所需要解决的问题是,当训练集数据中包含的恶意流量类型与测试集数据中包含恶意流量类型不同的时候,如何使训练出的学习器依然能够在测试集数据上表现出不错的性能。

然而对于训练集数据中完全不包含测试集中的恶意流量类型时,问题很难得到解决,只能从降低过拟合的角度进行入手.为了能够训练出在训练集和测试集上都表现良好的学习器,不再要求训练集中完全不包含测试集中任恶意流量类型,而是允许训练集中包含一部分测试集的数据,但是只能使用很少的一部分.

由于数据集 USTC-TFC2016 中一共包含 10 种类型的恶意流量,因此我们可以将 10 种恶意流量分别划分到源域和目标域数据集中,论文二采用的方式是使用 10 种恶意流量中的 9 种作为源域数据,剩下的一种作为目标域数据集.

### 5.3 解决方法

为了使得训练出的学习器能够在目标域数据集上表现出不错的性能,论文二引入了论文《Transferable Normalization: Towards Improving Transferability of Deep Neural Networks》中提出的 TransNorm 思想,设计了 Transnet 网络.TransNorm 是 BatchNorm 思想的一种改进,具体的实现方式相对较为复杂,就不在这里进行介绍了,可以参考论文《Transferable Normalization: Towards Improving Transferability of Deep Neural Networks》中的详细介绍.

### 5.4 复现结果

我们根据论文二中的思想设计了实验,也将 TransNorm 引入未知恶意流量的检测中.一开始我们也和论文二中一样将 10 种恶意流量中的 9 种用作源域数据集,剩下的一种用作目标域.Zeus 恶意流量作为目标域数据时的实验结果如图 Fig.17 Experimental results of Zeus malicious traffic as the target domain

图 17 所示.

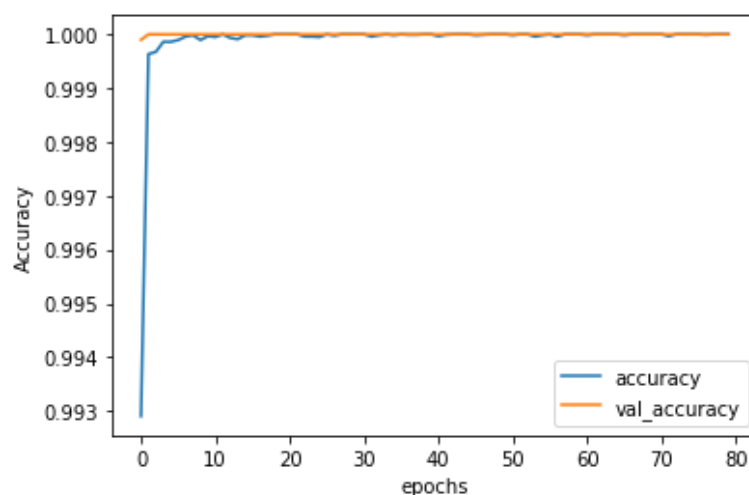


Fig.17 Experimental results of Zeus malicious traffic as the target domain

图 17 Zeus 恶意流量为目标域的实验结果

Fig.17 Experimental results of Zeus malicious traffic as the target domain

图 17 显示当使用 Zeus 作为目标域,另外 9 种恶意流量作为源域时,并且在训练集数据中完全不包含 Zeus 恶意流量的数据,依然达到了非常高的准确率,接近 100%.然而根据论文二中实验结果,其只有 90%左右的准确率,甚至部分实验只有 70%左右的准确率,即使使用 TransNet 网络迁移学习,依然只有 95%左右的准确率.这和我们复现的结果不一样,无法确定是什么原因造成的,可能论文二的作者在做实验时部分参数没有设置好.

除此之外论文二使用了 AlexNet,VGG19,ResNet-50 作为基准实验(baseline),我个人觉得是有部分问题的,因为比如 ResNet 网络主要是针对大型卷积神经网络,当网络层数比较多时,为了缓解梯度消失和梯度爆炸所提出来的,并不适用与本文的问题.

为了能继续复现实验,我们修改了源域和目标域的划分方式,此次我们使用 5 种恶意流量作为源域数据,另外 5 种作为目标域数据集,实验的结果如图 Fig.18 5 kinds of malicious traffic as the experimental results of the target domain

图 18.

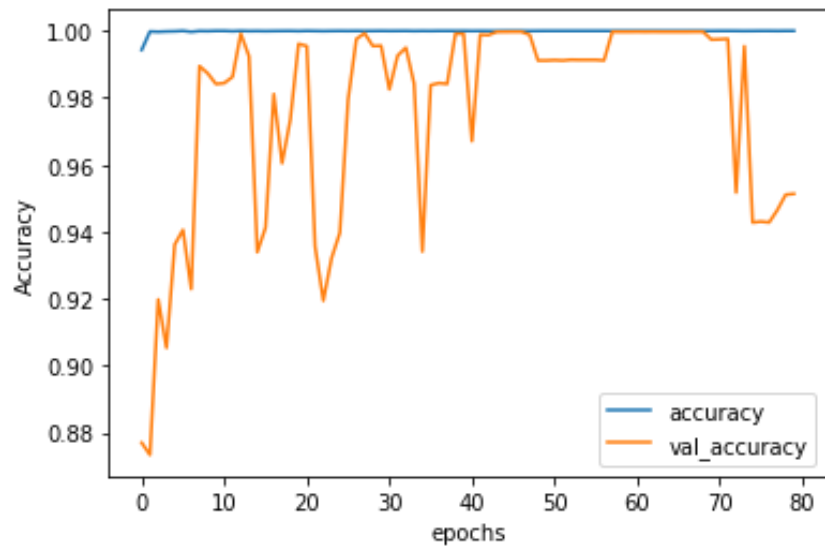


Fig.18 5 kinds of malicious traffic as the experimental results of the target domain

图 18 5 种恶意流量作为目标域的实验结果

根据 Fig.18 5 kinds of malicious traffic as the experimental results of the target domain

图 18 的实验结果,我们可以看出当使用 5 种恶意流量作为源域数据,另外 5 种恶意流量作为目标域数据时,训练集很快达到了 100%的准确率,然而测试集的数据却波动剧烈,难以收敛,这是由于测试集中包含了 5 种训练集中不存在的恶意流量.

为了使训练得到的分类器在目标域上能够有更好的表现,我们将 TransNorm 引入卷积神经网络框架中,即替换原来卷积神经网络中的 BatchNorm 层,实验的结果如 Fig.19 Experimental results after the introduction of TransNorm

图 19,可以看出相比于未使用 TransNorm 的分类器,引入 TransNorm 后的分类器的准确率大幅提高,达到了 99.99%的准确率.

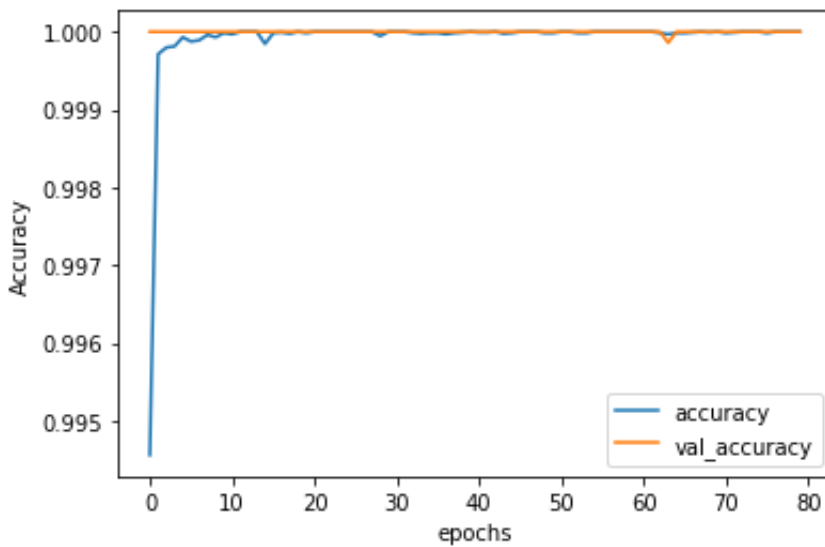


Fig.19 Experimental results after the introduction of TransNorm

图 19 引入 TransNorm 后的实验结果

## 6、结论

通过对论文一《Malware traffic classification using convolutional neural network for representation learning》和论文二《TransNet: Unseen Malware Variants Detection Using Deep Transfer Learning》的复现,可以发现使用卷积神经网络的方法进行恶意流量的检测能够达到非常高的性能,足以满足实际中的需求,还具有较高的拓展性;且使用卷积神经网络的方法不需要预先设计流量特性,原始流量数据是流量分类器的输入数据,分类器可以自动学习特征.同时也可以发现使用 TransNorm 在未知恶意流量的检测上能取得非常好的效果,将在实际的应用中发挥重要的作用,恶意流量检测的问题将得到很好的解决.

### Reference:

- [1] E. Biersack, C. Callegari and M. Matijasevic, Data traffic monitoring and analysis, Berlin:Springer, 2013.
- [2] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning", Book in preparation for MIT Press, 2016.
- [3] Y. Dhote and S. Agrawal, "A Survey on Feature Selection Techniques for Internet Traffic Classification" in Computational Intelligence and Communication Networks, Jabalpur, pp. 1375-1380, 2015.
- [4] Y. Bengio, A. Courville and P. Vincent, "Representation learning: A review and new perspectives", IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, pp. 1798-1828, Aug. 2013.
- [5] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", Nature, vol. 521, pp. 436-444, May 2015.
- [6] Z. Qingqing, L. Yong, W. Zhichao, P. Jielin and Y. Yonghong, "The Application of Convolutional Neural Network in Speech Recognition", Microcomputer Applications, vol. 3, pp. 39-42, June. 2014.
- [7] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy and B. Shuai, "Recent Advances in Convolutional Neural Networks", arXiv preprint arXiv:1512.07108, 2015.
- [8] A. Javaid, Q. Niyaz, W. Sun and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System", Proc.9th EAI International Conference on Bio-inspired Information and Communications Technologies, 2016.
- [9] M. Tavallaei, E. Bagheri, W. Lu and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", Proc. 2009 IEEE Int. Conf. Comput. Intell. Security Defense Appl., pp. 53-58.
- [10] Z. Wang, "The Applications of Deep Learning on Traffic Identification", [online] Available: <https://goo.gl/WouIM6>.
- [11] A. Dainotti, A. Pescapé and K. Claffy, "Issues and future directions in traffic classification", Network IEEE, vol. 26, no. 1, pp. 35-40, 2012.
- [12] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection", Wireless Communications and Networking Conference (WCNC) 2013 IEEE, pp. 4487-4492, 2013.
- [13] F. Haddadi and A. Nur Zincir-Heywood, "Data confirmation for botnet traffic analysis", Proc. 7th Int. Symp. FPS to be published.
- [14] "The Stratosphere IPS Project Dataset", 2016, [online] Available: <https://stratosphereips.org/category/dataset.html>.
- [15] Ixia Breakpoint Overview and Specifications, 2016, [online] Available: <https://www.ixiacom.com/products/breakingpoint>.
- [16] Z. B. Celik, R.J. Walls, P. McDaniel and A. Swami, "Malware traffic detection using tamper resistant features", Military Communications Conference MILCOM 2015 – 2015 IEEE, pp. 330-335, 2015.
- [17] W. Li, "Efficient Application Identification and the Temporal and Spatial Stability of Classification Schema", Computer Networks, vol. 53, pp. 790-809, Apr. 2009.
- [18] D. Koukis, S. Antonatos, D. Antoniadis, E. P. Markatos and P. Trimintzios, "A Generic Anonymization Framework for Network Traffic", IEEE International Conference on Communications, pp. 2302-2309, 2006.
- [19] Behaviour and Example, 2016, [online] Available: [http://www.fon.hum.uva.nl/praat/manual/IDX\\_file\\_format.html](http://www.fon.hum.uva.nl/praat/manual/IDX_file_format.html).
- [20] Y. A. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker et al., "Learning algorithms for classification: A comparison on handwritten digit recognition", Neural Networks, pp. 261-276, 1995.
- [21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., "Tensor-Flow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", arXiv preprint arXiv:1603.04467, 2016.
- [22] A. W. Moore, D. Zuev and M. Crogan, Department of Computer Science Queen Mary University of London, pp. 05-13, September



2005.

- [23]MH. Bhuyan, DK. Bhattacharyya and JK. Kalita, "Network Anomaly Detection: Methods Systems and Tools", IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303-336, 2014.