基于在线学习的分布式流量实时分析系统

1. 概述

目前，各种类型的流量充满了网络空间，其中包含了正常上网业务，具有CVE 编号的网络攻击、计算机病毒（邮件病毒、木马、蠕虫、勒索软件）等等。这些流量中既包含未加密的流量，也包含经过加密的流量，如广泛使用的安全传输层(TLS)协议。 如何开发出一个网络安全系统，能够有效抵对 TLS 加密与非加密的流量数据进行检测、分析与分类，并实时识别出网络上的恶意攻击行为成为现阶段面临的一个挑战。目前在工业界，已经已经有了基于网络端口映射的流量分类识别方法和基于有效载荷分析的流量分类识别方法等，但这些方法面临两个都面临了准确性和可靠性低的问题，且这两种方法是无法在加密流量上使用的，因此探究其他的流量检测方法显得格外重要。同时，CVE、恶意加密流量等网络攻击具有攻击量大、高并发的特点，对于检测系统更是提出了实时化与分布式部署的要求。

我们提出将分布式网络系统与深度学习技术、虚拟化技术相结合，同时引入在线学习机制做到实时推断与模型动态更新。该系统具有以下特点：利用 CyberFlood 产生 TLS 加密与非加密的、不同种类的业务流量和恶意流量；利用 Hive 分布式数据库存储原始流量数据；通过 Spark 和 Flink 流量分别对流量进行并行批处理和流式处理；使用 CNN+LSTM 的时空深度学习网络对不同类型的流量进行分类；利用 redis 缓存加速实时流量的特征读取；利用 Docker 虚拟化容器对深度学习模型进行部署。该系统同时具有高隔离性、高容错性、高准确性和高实时性特点，可以解决传统安全系统所面临的的挑战。

目前我们的系统版本为 V2.0，可以识别出包含加密与非加密的业务流量、恶意软件流量和网络攻击流量这三种流量类型，并进行动态流量数据的可视化呈现。

2. 目标

2.1.对 CyberFlood 产生流量数据进行抓取、分析与分类。

2.2.能够实时、准确识别出网络上的 TLS 加密与非加密的正常业务流量、网络攻击流量，恶意软件流量，模型上线后能够不断进行动态调整。

2.3.实现一个可交互的、具有高并发能力、高容错性的分布式网络流量监测系统，并且能对流量数据可视化。

3. 系统架构

系统架构大致分为以下几个部分：

3.1 流量生成模块：CyberFlood 生成包括 TLS 加密与非加密的业务流量与恶意流量。

3.2 流量抓取模块：Python 抓取流量并解析。

3.2 批处理模块:将解析后流量以 JSON 格式的存储在 Hive 分布式数据库中，然后由 Spark 对流量特征进行批处理。

3.3 流式处理模块：将解析后的流量由 Kafka 收集送往 Flink 对流量特征进行流式处理。

3.4 Web 后端模块：批处理和流式处理的结果都送往后端服务器的 MySQL 数据库以 CSV 格式存储。

3.5 模型训练模块：使用带标签的流量数据对时空神经网络(CNN+LSTM)模型进行训练和交叉验证，然后对模型进行准确的评估，通过不断地调试调优改进模型。

3.6 模型线上推断与在线学习模块：Java 后端将未知种类的流量特征从 MySQL 数据库中导出，送往 Redis 进行缓存，然后将 Redis 中的流量特征通过 gRPC 请求发往 Docker 容器，然后以 JSON 格式返回流量的分类结果。在推断的过程中，模型权重会动态调整并更新。

3.7 流量可视化模块：通过前端可视化呈现流量的分析结果。

整体系统架构如图 1 所示。

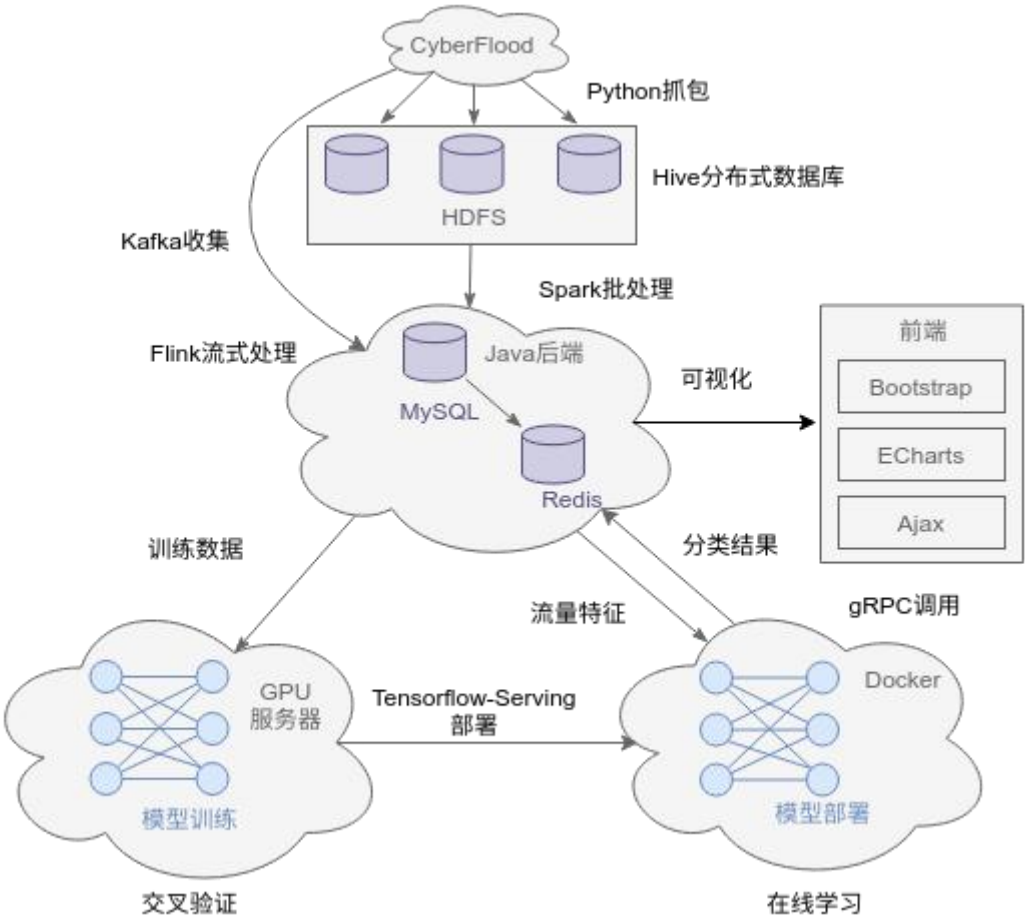

图 1 系统架构

4. 关键技术

4.1 数据采集与特征选取

数据集通过组委会提供的 CyberFlood 工具，生成 TLS 加密与非加密的正常业务流量与网络攻击行为流量，再使用 Python 对指定的网络端口抓取流量， 数据采集流程如图 2 所示。

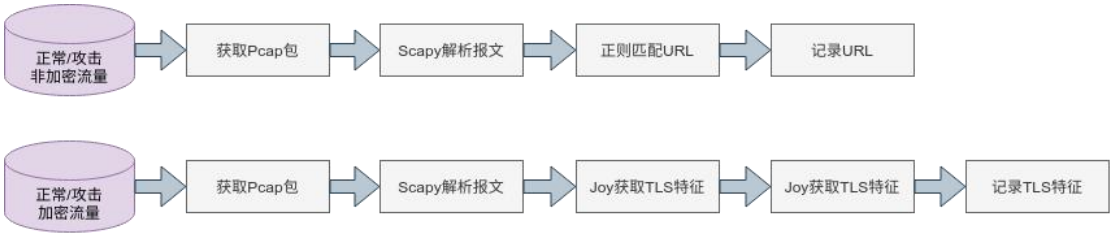

图 2 数据采集流程

其中对于非加密流量我们为了使用模型精度达到最大化，我们直接匹配其 URL 字段用于模型训练。

而对于 TLS 加密流量，检测加密流量中的恶意攻击行为通常应先解密相关的加密流量（如 SSL 和 TLS），再检测相应恶意攻击行为，但通过 CyberFlood 所生成的加密流量并未给出相应证书签名，故本系统考虑在不解密加密流量的情况下，通过获取数据流中的元数据特征并将其导入至机器学习模型中进行识别。

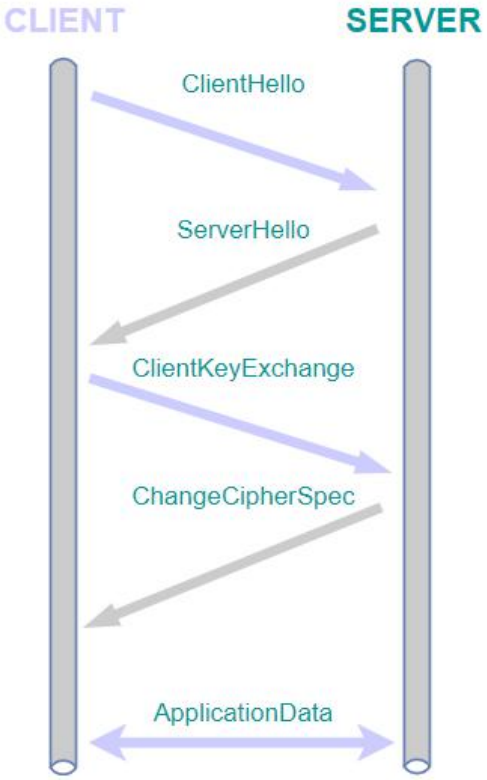

图 3 TLS 握手协议示意图

接下来介绍该操作的原理。由于加密流量的 TLS 数据流中的元数据包含了无法隐藏或者加密的指纹信息，TLS 数据流可被用于模糊明文数据，但同时必须生成一组“可观测的复杂参数”，后者可用于训练数据模型。当一个 TLS 数据流开始后，会先执行一次握手。客户端会向服务端发送一个 ClientHello 的消息，该消息包括一组参数（包括加密算法、版本等信息）。像 ClientHello 的这种 TLS 元数据主要在加密数据传输前进行交互，没有被加密。这样，数据模型就可以通过分析元数据来检测恶意攻击行为，而不需要对加密数据进行解密操作。

如图 3 所示，一个完整的 TLS 会话过程一定包含以下类型的消息：ClientHello、ServerHello、ServerHelloDone、ClinentKeyExchange、

ChangeCipherSpec。

根据 TLS 流在交互之初需要同远程服务器进行握手是不加密的特性，可以观测到的未加密 TLS 元数据包括 ClientHello 和 clientKeyExchange。从这些包的信息中，我们可以推断出客户端使用的 TLS 库等信息。通过比较获取到的特征信息，可以观察到业务流量与业务流量存在较大差别，因此将 TLS 流作为本系统特征来源，利用思科研究人员开发的基于 libpcap 的通用工具，用于分析并提取捕获到的数据流（恶意流量和正常流量）的数据特征，包含 clientHello, serverHello, certificate 和 clien-tKeyExchange 等信息基于进行特征提取。

接下来介绍我们对特征分析方法进行介绍。客户端方面，我们首先观察两个 TLS 特征：Offered Ciphersuites 和 Advertised TLS Extensions。对于前者，恶意流量更喜欢在 clientHello 中提供 0x0004(TLS_RSA_WITH_RC4_128_MD5)套件，而业务流量则更多提供 0x002f(TLS_RSA_WITH_AES_128_CBC_SHA)套件；对于后者，大多数 TLS 流量提供 0x000d(signature_algorithms)，但是业务流量会使用以下很少在恶意流量中见到的参数，如表 1 所示。

Encode	Feature
0x0005	status request
0x3374	next protocol negotiation
0xFF01	renegotiation info

表 1 TLS 特征以及对应编码

观察业务与恶意流量客户端公钥的区别。业务流量往往选择 256-bit 的椭圆曲线密码公钥，而恶意流量往往选择 2048-bit 的 RSA 密码公钥。

服务端方面，我们能够从 serverHello 流中得到服务端选择的 Offered Ciphersuites 和 Advertised TLS Extensions 信息。业务流量的选择比较多元化，而恶意流量往往会选择较为过时的技术。在 certificate 流中，我们能够得到服务端的证书链。无论是恶意流量还是业务流量，其证书的数量都是差不多的，但若我们观察长度为 1 的证书链，就能够发现，其中的 70%都来自恶意流量自签名，0.1%来自业务流量自签名。

由上述分析可知，加密恶意流量与加密业务流量具有非常明显的 TLS 特征差异，这些差异主要体现在客户端 TLS 扩展、客户端加密套件、客户端公钥长度、服务端 TLS 扩展、服务端证书签名长度以及服务端证书签名算法等。如表 2 所示。

TLS_Feature	Illustration
Client_Extensions	A set of length values followed by a set of extension type values is used to describe the TLS extension usage observed in the message of the TLS flow

Client_Key_length	The maximum length of the top n TLS public keys observed in the message of the data stream
Server_Extensions	A set of length values followed by a set of extension type values is used to describe the TLS extension usage observed in the message of the TLS flow
Server_Signature_Key_Size	Length of client signature
Server_Signature_Key_Algo	Algorithm used for client signature
TLS_Record_Time	The sequence of TLS arrival intervals of the top n records in a TLS stream
TLS_Record_Length	The sequence of length values of the top n records in TLS stream

表 2 TLS 特征选取结果

具体特征选取操作，我们采用思科开发的基于 libpcap，可以从实时网络流量中提取数据或直接捕捉到数据包文件的分析工具 Joy，其工作机制跟 IPFIX 或 Netflow 有些类似，在捕捉到数据之后，Joy 将以 JSON 格式呈现出这些数据。除此之外，Joy 还包含了分析工具，用户可以直接使用这些分析工具来对数据文件进行分析。

利用该工具可以快捷的捕获实时流量并从中提取 TLS 类数据特征，并将分析结果通过 Json 文件存储在特定目录下。针对机器学习模型所需的特定 TLS 特征，可以利用 Joy 工具的 Sleuth 分析功能，该功能通过类 SQL 语法选取指定特征来进一步处理 Json 文件，过滤出符合特定条件的流，从流中选择数据元素并打印出结果。

4.2 流量存储

经过特征提取后的业务类型或攻击类型流量都以 CSV 文件格式保存在 MySQL 数据库中。加密流量的记录包括 ID 字段、Client_Extensions、Client_Key_length、Server_Extensions 等。流量存储如表 3 所示。

ID	Clint-Extensions	Client_Key_length	Server_Extensions	Server_Signature_Key_Size	Server_Signature_Key_Algo	TLS_Record_Time	TLS_Record_Length	Time
1	0x23	128	0x10	1280	ECDHE_RSA_WITH_AES_128_CBC_	233	128	2020-7-29 16:44:00

					SHA256			
2	0x0d	2048	0x11	512	ECDH_RSA_WITH_AES_128_CBC_SHA256	166	128	2020-7-29 19:50:00
3	0x05	256	0x3374	512	DHE_RSA_WITH_AES_128_CBC_SHA256	230	128	2020-7-30 20:13:00
4	0x10	64	0x0d	512	RSA_WITH_AES_128_CBC_SHA	450	64	2020-7-30 13:05:00

表 3 TLS 加密流量数据存储字段

非加密流量记录的包括 ID 字段，URL 字段与时间戳字段。 流量数据存储如表 4 所示。

ID	URL	Time
1	GET /rtv.png?plf=3	2020-7-29 16:44:00
2	GET /default.aspx?wa=wsignin1.0	2020-7-29 19:50:00
3	GET /nagiosql/config/content.css	2020-7-30 20:13:00
4	GET /Interop.MessengerAPI.dll	2020-7-30 13:05:00

表 4 非加密流量数据存储字段

后期通过模型推断，完成分类后的流量则存储如表 5 所示，可通过后段代码读取该表数据从而完成前端动态流量可视化呈现。其中 0 标签表示为业务流量，1 标签表示为网络攻击流量，2 标签表示为恶意软件流量。

ID	Label	Time
1	0	2020-7-29 16:44:00
2	0	2020-7-29 19:50:00
3	1	2020-7-30 20:13:00

表 5 流量预测结果存储

4.3 数据预处理

该部分由数据获得、数据读取、特征提取三部分组成。先使用 Python 的 PyMySQL 库将数据从 MySQL 中提取出来，以 CSV 表格数据形式存储，然后使用 Python 的 Pandas 库读取 CSV 格式的数据表格，以 DataFrame 形式存于内存。

对于加密流量，我们采用利用 Python 的 Pandas 库进一步进行处理，如对离散型特征进行 one-hot 编码来对于每一个特征的多个可能值进行处理转变为多个二元特征，此外，对特征进行归一化、数值化等。

对于非加密流量的 URL 特征，我们利用文本处理中常用的技术“单词向量空间模型”(word vector space model)由字符型的 URL 构建数值特征。

数据预处理流程如图 4 所示。

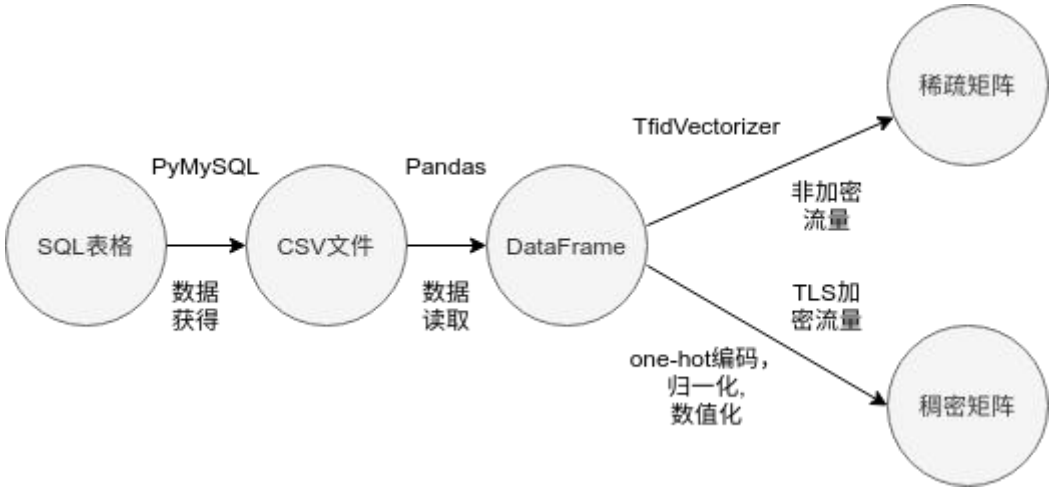

图 4 数据预处理流程

（1）数据获得

使用 Python 的 PyMySQL 库将数据从 MySQL 中提取出来，以 CSV 表格数据形式存储。

（2）数据读取

然后使用 Python 的 Pandas 库读取 CSV 格式的数据表格，将m条n维的流量记录以 DataFrame 格式存于m行n列的矩阵X中。

（3）特征提取

对于加密流量，由于 TLS 特征多为数值形式，我们直接利用 Python 的 Pandas 库进一步进行处理，对部分离散型特征进行 one-hot 编码来对于每一个特征的多个可能值进行处理转变为多个二元特征。此外，对数值特征进行归一化、数值化等操作。

对于非加密流量，由于流量特征为 URL 文本格式，读取后得到的是一个文本数据，设使用正则表达式解析预处理和分词之后的流量数据为单词-文本矩阵，m条流量的 URL 数据对应m条文本，每条文本的向量空间为单词向量空间(word vector space)，即对每一条流量 URL 文本，用一个向量表示该文

本的“语义”，向量的每一维对一个一个单词，其数值为该单词在该文本中出现的权值。

形式化的定义如下，给定一个含有m条文本的集合$D = \{d_1, d_2, \ldots, d_n\}$，以及在所有文本中出现的$n$个单词的集合$W = \{w_1, w_2, \ldots, w_n\}$。将单词在文本中出现的数据用一个单词－文本矩阵(word-document matrix)表示，记作X

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

这是一个$m \times n$矩阵，元素X_{ij}表示单词w_j在文本d_i中出现的频数或权值。由于单词的种类很多，而每个文本中出现单词的种类通常较少，所以单词－文本矩阵是一个稀疏矩阵。

权值我们用单词频率－逆文本频率（term frequency-inverse document frequency,TF-INDF）表示，其定义是

$$TFIDF_{ij} = \frac{tf_{ij}}{tf_{i\cdot}} log \frac{df}{df_j}, i = i, 2, \ldots, m; j = i, 2, \ldots, n$$

式中tf_{ij}是单词w_j在文本d_i中出现的频数，$tf_{i\cdot}$是文本d_i中出现的所有单词的频数之和，df_j是含有单词w_j的文本数，df是文本集合D的全部文本数。也就是说一个单词在一个 URL 中出现的频数越高，这个单词在这个 URL 中的重要度就越高；一个单词在整个 URL 集合中出现的 URL 条数越少，这个单词就越能表示其所在的 URL 的特点，重要度就越高；一个单词在一条 URL 中的 TF-IDF 是两种重要度的积，表示综合重要度。

单词向量空间模型直接使用单词－文本矩阵的信息。单词－文本矩阵的第i行向量x_i表示 URL d_i

$$x_i = [x_{i1}, x_{i2}, \ldots, x_{in}], i = 1,2, \ldots, m$$

其中x_{ij}是单词w_j在 URL d_i中的权值，$i = 1,2, \ldots, m$，权值越大，该单词在该 URL 中的重要程度就越高。这时矩阵X也可以写作$X = \begin{bmatrix} x1 \\ \vdots \\ xm \end{bmatrix}$。

4.4 模型训练

我们将流量信息建模为行向量作为特征取值，列向量为不同流量的矩阵。我们使用 CNN+LSTM 时空神经网络模型。CNN 神经网络学习流量的空间信息，LSTM 神经网络学习流量在时间方向的信息。该模型可以完成对 TLS 加密与非加密的流量的分类并对时序流量进行动态预测。模型总体架构如图 5 所示。

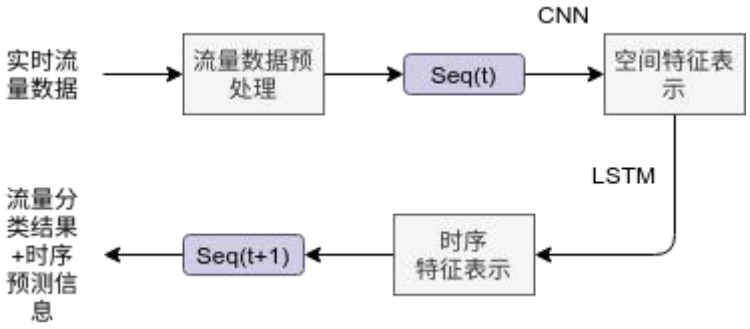

图 5 模型整体架构

我们在将文本用 TF-IDF 表示后可以获得$m \times n$矩阵，（m 为为流量条数，n 为当前单词表示所在的向量空间维度）。我们需要将该每一条流量的向量表示R^n嵌入到一个低维空间$R^p\ (n << p)$，需要建立一个R^n到R^p的线性映射。对每一个$n \times p$矩阵A都定义一个从R^n到R^p的线性映射：$x \to xA$，完成了词嵌入。

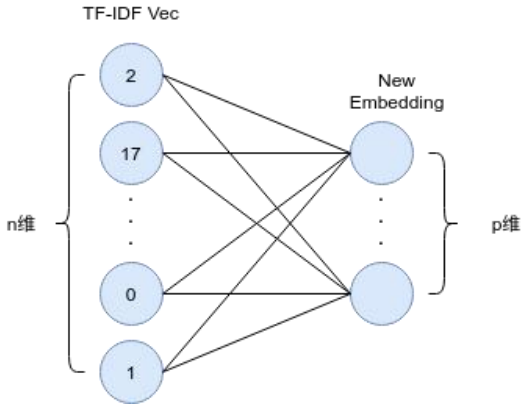

图 6 对单个向量 embedding 的示意图

完成词嵌入后，我们定义k为卷积核的大小，向量$M \in R^{k \times p}$是卷积操作的卷积核，对于我们每一个流量j，我们有窗口张量W_j做为连续的k个流量向量组成的张量，表示为：

$$W_j = \begin{bmatrix} x_j \\ \vdots \\ x_{j+k-1} \end{bmatrix}$$

然后我们使M卷积核对窗口张量W（k-grams）的每一个位置j进行卷积，产生特征图 $c_j \in R^{m-k+1}$,在特征图中每一个窗口张量W_j所对应的元素按照如下规则计算：

$$c_j = f(W_j \circ M + b)$$

在这里$\circ$是两个矩阵按元素逐个相乘运算，$b \in R$ 是一个偏置，f是一个非线性变换函数，它可以选择 sigmoid,tanh 等，这里我们选择 ReLU 做为非线性变换函数。我们的 CNN-LSTM 模型使用多重卷积核以产生多重特征图。对于大小相同的q个卷积核，我们会产生q个特征图。对每一个W_j，我们可以将其重新以不同特征表示，我们可以将W表示为：

$$W = [c_1\ c_2\ \ldots c_q], i = 1,2,\ldots,q$$

在这里，c_i是由第i个卷积核产生的特征图，$W \in R^{m-k+1}$ 的每一行W_j是一个由q个卷积核在位置j产生的新的特征表示，接着会被输入 LSTM 模型，如图 5 所示。

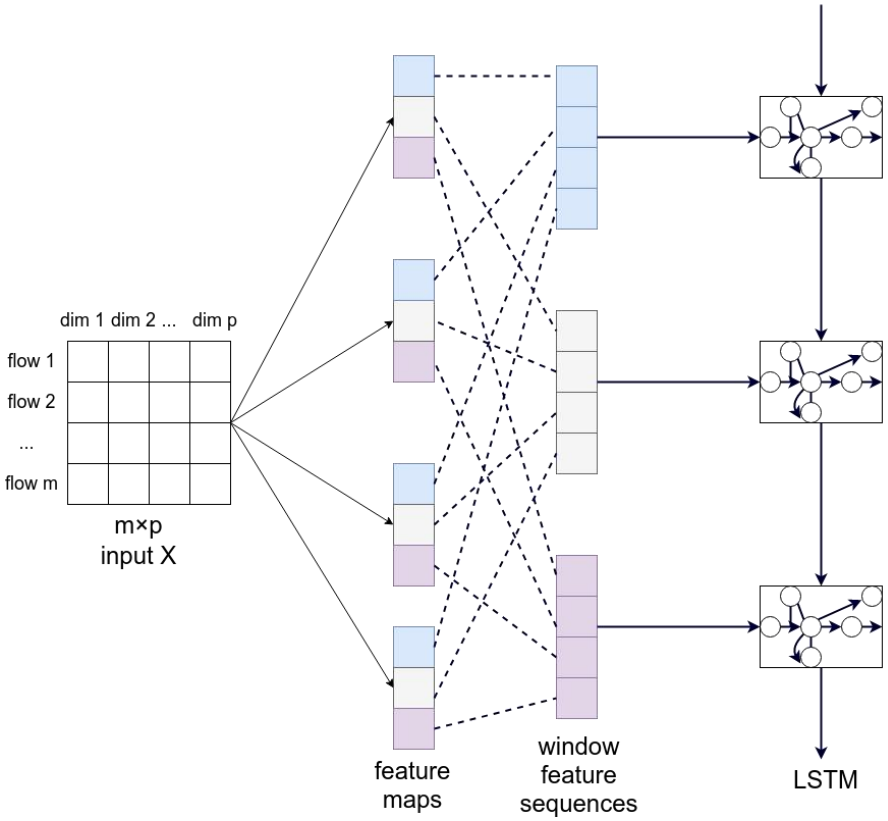

图 7 CNN 特征提取示意图

一般来说在完成卷积操作后，卷积神经网络中会有最大池化或k最大池化作用于多个特征图以选择最重要的 1 个或k个特征。然而，LSTM 由于输入序列而具有特殊性，由于不连续地选择特征，池化操作会破坏序列的组织结构。因为我们的 LSTM 是以 CNN 为基础的，因此我们在卷积操作后不会进行池化操作。

接下来是 RNN，它能够传播历史时序信息，在处理序列数据时，他同时会考虑到当前输入x_t以及上一步隐藏层的输出h_{t-1}。然而 RNN 无法学习长期依赖，因为两个时间步之间的差距很大。

我们这里采用标准 LSTM 架构。在每一步中，模块的输出由一系列做为旧的隐藏层的h_{t-1}和当前时间输入的x_t共同控制。模型还包括遗忘门f_t，输入门i_t，输出门o_t。这些门共同决定如何更新当前的记忆单元c_t以及当前的隐藏状态h_t，我们使用d 表示 LSTM 的记忆维度，在当前架构下的所有向量共享相同的维度，LSTM 的函数定义如下：

$$i_t = \sigma(W_i \cdot [h_{t-1},\ x_t] + b_i)$$
$$f_t = \sigma(W_f \cdot [h_{t-1},\ x_t] + b_f)$$
$$q_t = tanh(W_q \cdot [h_{t-1},\ x_t] + b_q)$$
$$o_t = \sigma(W_o \cdot [h_{t-1},\ x_t] + b_o)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot q_t$$
$$h_t = o_t \odot tanh(c_t)$$

在这里，σ是一个有[0,1]输出的逻辑函数。tanh 代表双曲正切激活函数，输出在[-1,1]区间内。⊙代表逐元素相乘。我们将f_t视为控制旧单元信息的传送，i_t控制多少新的信息会被存储在当前记忆单元中，o_t在记忆单元c_t的基础上控制输出什么。LSTM 非常利于学习时间序列与学习长期依赖，因此我们选择在 CNN 之上增加 LSTM 学习这些在更高特征序列中的依赖。

我们将 LSTM 最后一步的最后一个隐藏状态输出视为流量的新的表示，我们添加一个 softmax 层，最终获得不同分类标签的预测概率值。我们以最小化交叉熵损失的形式训练整个模型。给定流量样本x_i以及它的真实分类标签$y_i \in \{1,2,..,k\}$。这里k是流量可能的种类数量。每个流量样本x_i对每个标签$j \in \{1,2,..,k\}$的预测概率值为$p_{ij} \in [0,1]$，故损失函数定义如下：

$$L(x_i, y_i) = \sum_{j=1}^{k} 1\{y_i = j\} log(p_{ij})$$

这里 $1\{condition\}$是一个示性变量，当$condition$满足时为 1，当$condition$不满足时为 0，我们使用随机梯度下降法（SGD）来学习模型参数。

我们采用两种手段 dropout 和 L2 权重正则化来防止模型过拟合。在我们的模型中，我们在将流量序列输入卷积层之前以及在将 LSTM 的输出传入 softmax 层之前，我们对流量向量使用 dropout。L2 正则化作用于 softmax 层的权值。

最终我们的恶意流量检测模型的训练流程如图 6 所示。

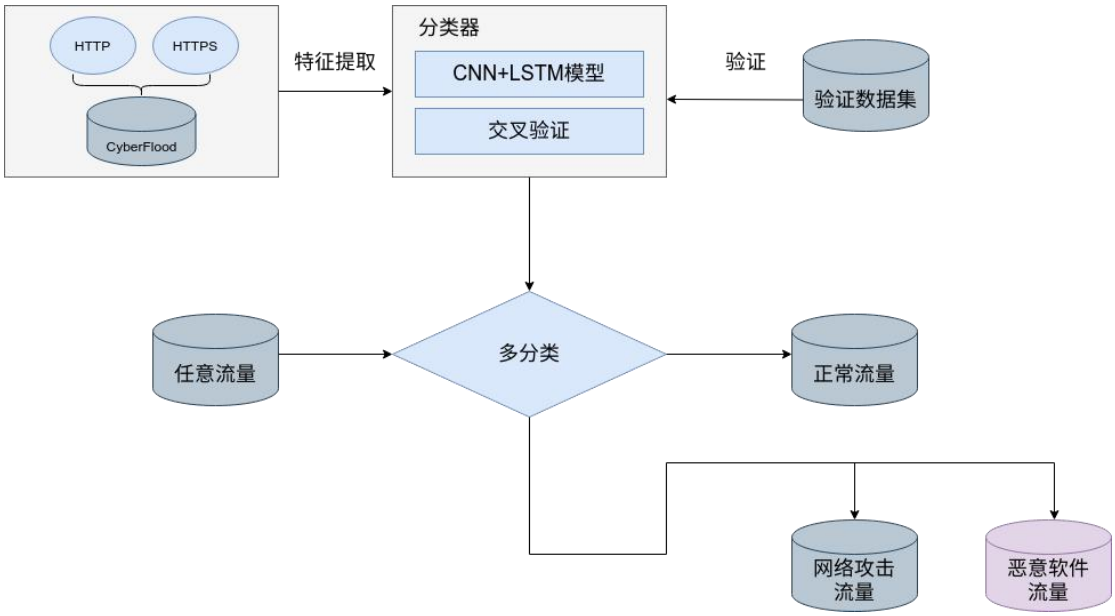

图 8 模型训练流程

为了避免测试的偶然性，本项目采用交叉验证方法，数据集分为验证集测试集，以验证集得到的正确率作为精度。

4.5 模型评价

我们评价分类器的标准包括整体准确率（OA）、查准率（P）、查全率(R)和综合评价(F1)。其中，查准率、查全率和综合评价代表分类器对每种类别的识别能力，整体准确率代表分类器整体的识别能力。

我们将类别为 K 定义为正例，类别为非 K 定义为负例。各种指标中相关参数如表 3 所示。

评价标准	参数意义
TP(true positives)	类别为 K 的流被判为类 K 的流个数
TN(true negatives)	类别非 K 被判为非 K 的流的个数
FP(false positives)	类别非 K 但判为 K 的流的个数
FN(false negatives)	类别为 K 但判为非 K 的流的个数

表 6 评价标准

整体准确率（OA）的表达式如式(1)所示：

$$OA=\frac{\sum_{i=1}^{n}(TP_i+TN_i)}{\sum_{i=1}^{n}(TP_i+FP_i+TN_i+FP_i)} \tag{1}$$

查准率（P）和查全率(R)的定义式如式(2)和式(3)所示：

$$P=\frac{TP}{TP+FP} \tag{2}$$

$$P=\frac{TP}{TP+FN} \tag{3}$$

综合评价 F1-Score 的定义如式(4)所示：

$$F_1=\frac{2PR}{P+R}=\frac{2TP}{2TP+FP+FN} \tag{4}$$

目前我们模型的流量分类预测整体准确率达到了 93.5%，查准率为 96.2%，查全率为 99.3%，综合评价得分为 96.0%。

4.6 数据可视化

（1）使用 Java 结合 Bootstrap 框架实现前端页面

（2）使用 Echarts 图标库来对数据提供直观、可交互数据可视化图表

5. 创新点与特色

5.1 数据并行处理。采用 Hive 分布式数据库对原始流量数据进行存储。同时使用 Spark 批处理机制与 Flink 流式处理机制对流量数据进行并行处理。

5.2 CNN+LSTM 时空神经网络使用。我们采用 CNN 对流量空间特征进行提取，LSTM 对流量时序特征进行提取，既能完成不同种类流量分类，也能

完成时序流量的动态预测功能。

5.3 在线学习与实时推断。我们引入虚拟化容器 Docker 对深度学习模型进行部署，通过 gRPC 远程过程调用完成推断操作，具有高隔离性与容错性。引入 redis 缓存流量特征确保实时性。同时，我们的模型具有在线学习特性，能够根据线上流量特性自行进行动态调整，确保模型的准确度。

6. 系统界面设计

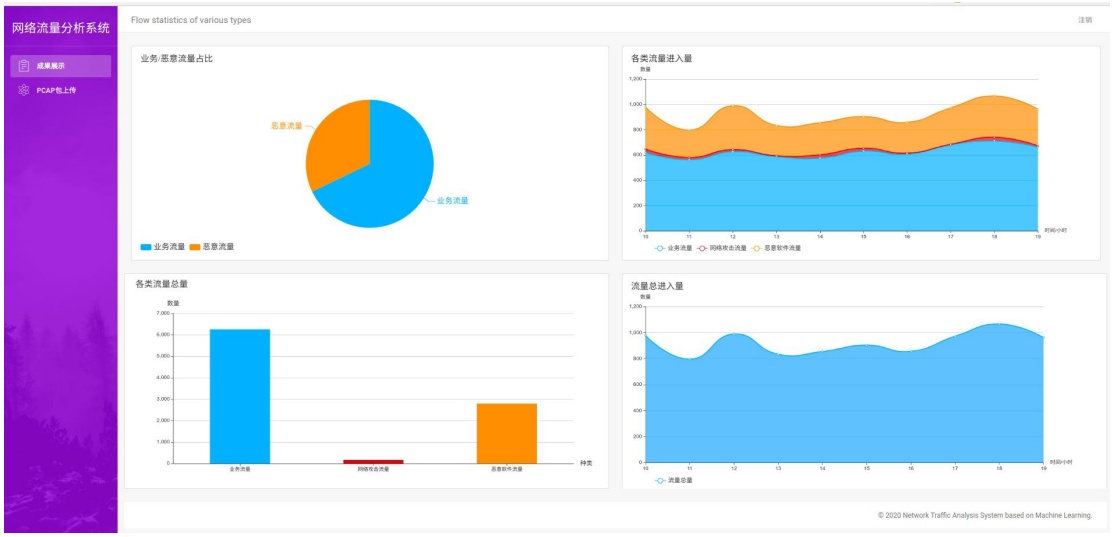

图 9 系统界面

6.1 图 1：以饼状图的形式对恶意流量和业务流量进行显示，展示恶意流量和业务流量的分布。

6.2 图 2：以折线图的形式展现各流量的时间序列分布图，可以直观的看出业务流量、网络攻击流量、恶意软件流量在不同时间的动态变化趋势。

6.3 图 3：以条形图的形式统计单日内业务流量、网络攻击流量、恶意流量的数量分布。

6.4 图 4：以条形图的形式展现所有流量总和随时间变化的趋势。