第1章 目 录

一. 作品概述

基于多维度特征加强的恶意代码检测系统

作品简介

随着大数据时代的到来，以恶意软件和网页后门为主的攻击对个人计算机和工业生产造成了严重威胁，它们不仅能窃取敏感信息，还能够破坏系统的正常运行，甚至控制整个网络设备。但是随着恶意代码的种类和数量不断增多，以及对抗检测的技术不断发展，以传统的基于签名和机器学习的检测方法因无法高效地提取恶意代码特征，已经难以识别新型的恶意代码。因此，本作品采用深度学习技术，基于多维度特征加强检测的模型方法，进行恶意软件与 Webshell（网站后门）检测，本作品所做的工作如下:

1.针对 Webshell（网页后门）检测，提出一种基于多维度特征的检测方法。在内容特征上，使用字符级单位转化为灰度图像的方法，将特征图输入卷积神经网络进行训练捕捉内容层面的特征；在语义特征上，使用 BERT 模型获取代码的 CLS（Classification）特殊标记头用以全段语义特征表示；在代码结构方面，获取函数调用流程树，通过图卷积网络训练学习结构关系。最后，本作品将三种不同特征维度的模型聚合以加强后门代码的检测能力，通过与三种模型单独检测的对比分析，证明了本作品中提出的多维度特征加强检测方法的有效性。

2.针对恶意软件（木马、蠕虫、挖矿软件等）检测，提出一种基于 API 调用序列的检测方法。将软件的执行时所需 API 函数、动态链接库、节数据处理为文本序列。对于传统词嵌入方法无法提取 API 序列语义关系的问题，本作品使用预训练的 Roberta 模型进行动态词嵌入。对于软件执行时重复调用 API 函数导致序列过长的问题，提出基于聚焦文本词向量权重的序列压缩算法，有效减少数据噪声、聚焦重要特征。为了解决单一模型检测效果差的问题，提出一种多维度特征加强的 TextCNN-Liner-Att-BiLSTM 模型，本模型使用 Liner 获取序列中浅层特征，使用 Text-CNN 获取序列中短期依赖的深层次特征，使用 Att-BiLSTM 获取序列中长期依赖的深层次特征，并将提取的深浅层次特征合并输入全连接层进行分类检测。最终本模型通过实验与对比，在模型检测的准确率、精确率、召回率等多个方面上都证明了本作品的方法的优势所在。

关键词：恶意代码检测；多维度特征加强；API 调用序列；词嵌入；权重序列压缩

二. Webshell 检测模型结构设计

本模型结构流程从代码的三个特征层面依次介绍设计结构的设计流程，依次从内容特征提取层、语义特征提取层、结构特征提取层和最后的置信度融合层进行详细介绍。具体模型设计的全局流程图如下：

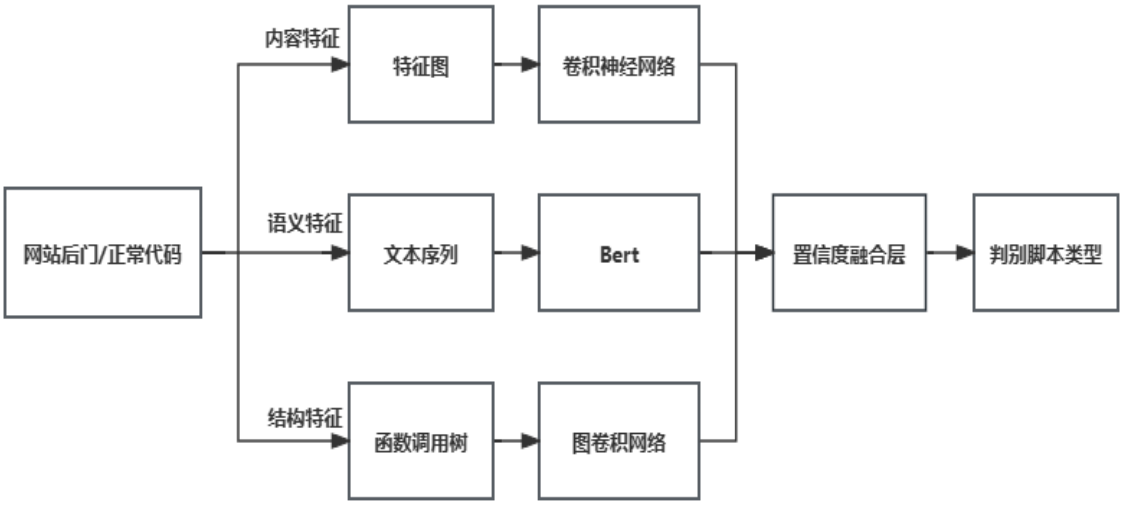


图 1　Webshell 检测模型

1.1 Webshell 检测模型结构设计

1.1.1 内容特征捕捉层

本层模型结构设计旨在提取代码的内容特征，以下为本层的详细设计过程：

1. 字符到像素点的转换：

假设我们有一个代码文件，其中包含n个字符，每个字符的 ASCII 码值为$char_i$，其中 $i = 1,2,\ldots,n$ 。我们需要将每个字符的 ASCII 码值转换为一个灰度像素点。由于 ASCII 码值的范围是 0-255，我们首先将其除以 256 进行归一化，以获得一个范围在 0-1 之间的值。

$$pixel_i = \frac{char_i}{256}$$

2. 构造灰度图像：

将上述转换得到的像素值用来构造一个灰度图像。如果代码文件的长度不足以填满一个 128x128 的图像，可以通过填充 (padding) 来补足。设 G 为构造的灰度图像，$G(x,y)$

表示图像中位置(x, y)的像素值。

3. 调整图像大小（Resize）：

将构造的灰度图像调整到所需的输入尺寸，例如，调整到适合卷积神经网络输入的尺寸。使用 Resize 函数进行尺寸调整：

$$G' = resize(G, size_{desired})$$

其中$size_{desired}$是卷积神经网络所需的输入尺寸。

4. 输入到卷积神经网络（CNN）：

将调整大小后的图像 `G'` 输入到卷积神经网络中进行特征提取和分类。令 CNN 表示卷积神经网络模型，`F` 表示网络的输出，通常是一系列类别的概率分布。

$$F = CNN(G')$$

其中 $F(c)$表示属于类别 c的概率。

综上所述，整个处理流程可以用以下公式表示：

$$F(c) = CNN\left(resize\left(256chari, sizedesired\right)\right)$$

对于每一个字符$char_i$，按照上述步骤生成灰度图像，调整图像大小，并输入到卷积神经网络中进行处理。最终，网络输出每个类别的概率分布，可以据此内容特征层面进行恶意后门脚本的检测和分类。

1.1.2 语义特征捕捉层

基于 BERT 模型进行语义层面的特征捕捉，并利用 CLS 标记进行恶意后门的分类检测的流程可以公式化如下：

1. 输入代码文件到 BERT 模型：

假设我们有一个代码文件，其内容可以表示为一系列的单词或标记（tokens）序列，记为$T = [t_1, t_2, \ldots, t_m]$，其中 m是标记的数量。

2. 转换为 BERT 的输入格式：

将代码文件的内容通过 BERT 的分词器（tokenizer）转换为 BERT 能理解的输入格式，包括输入序列T和相应的注意力掩码（attention mask） $Mask$。

3. 通过 BERT 模型提取特征：

使用 BERT 模型处理输入序列T，以获取每个标记的隐藏表示。BERT 模型的输出是一个固定大小的隐藏层表示，其中 CLS 标记（位于序列的开始）的表示通常用于捕捉整个输入序列的全局特征。

$$H = BERT(T, M)$$

其中H 是 BERT 模型输出的隐藏层表示，H_{CLS}是 CLS 标记对应的特征向量。

4. 特征向量通过全连接层：

将 CLS 标记的特征向量H_{CLS}输入到三个连续的全连接层（或称为线性层）以进行分类。记这三个全连接层的权重分别为W_1, W_2, W_3，偏置项分别为b_1, b_2, b_3，并且 F_1, F_2, F_3 分别代表每一层的输出。

$$F_1 = \sigma(W_1 H_{CLS} + b_1)$$
$$F_2 = \sigma(W_2 F_1 + b_2)$$
$$F_3 = W_3 F_2 + b_3$$

其中 σ表示激活函数，本模型使用 ReLU 激活函数。

5. 输出置信度进行分类：

最后一层全连接层F_3 的输出被用作分类器的输入，通过softmax函数获得每个类别的置信度分布。

$$P(c) = softmax(F_3)$$

其中$P(c)$表示代码属于类别c）的置信度。

整个流程的公式化表达可以概括为：

$$P(c) = softmax\left(W_3\sigma\left(W_2\sigma\left(W_1 H_{\{CLS\}} + b_1\right)\right) + b_3\right)$$

这个表达式描述了从代码文件输入到最终的分类置信度输出的完整过程。在实际应用中，这个过程通常涉及到训练阶段，其中通过反向传播算法优化权重（W_1, W_2, W_3）和偏置（b_1, b_2, b_3）以最小化分类错误。

1.1.3 结构特征捕捉层

基于代码结构层面的特征捕捉，使用 Tree-sitter 库和 Word2Vec 以及图卷积网络（GCN）进行恶意后门的分类检测的流程可以公式化如下：

1. 构建抽象语法树（AST）：

使用 Tree-sitter 库解析代码文件，得到函数节点间调用的抽象语法树（AST）。

2. 函数节点命名的词嵌入：

将 AST 中的每个函数节点名通过 Word2Vec 模型转换为词嵌入向量。记函数节点名为 f_i，其中（$i = 1, 2, \ldots, n$）表示函数节点的索引，n 是 AST 中函数节点的总数。Word2Vec

模型将每个节点名 f_i 映射到一个固定维度的向量空间中。

$$e_i = Word2Vec(f_i)$$

其中 e_i 是函数节点 f_i 的词嵌入向量。

3. 构建图结构数据:

根据 AST 中的函数调用关系构建图结构数据 G，其中节点集合 V 表示所有的函数节点，边集合 E 表示函数之间的调用关系。

4. 图卷积网络（GCN）处理:

将图结构数据 G 和节点的词嵌入向量 e_i 输入到图卷积网络中。记 K 为图卷积网络的层数，$H^{\{(k)\}}$ 为第 k 层的节点特征表示，$W^{\{(k)\}}$ 为第 k 层的权重矩阵。

其中 $H^{\{0\}}$ 是初始的词嵌入向量集合，GCN_k 表示第 k 层的图卷积操作。

5. 全连接层分类:

图卷积网络的最后输出通过全连接层进行分类预测。

6. 输出分类预测:

最后一层全连接层的输出 Z_2 通过$sigmoid$函数转换为分类概率分布。

$$P(c) = sigmoid(Z_2)$$

其中 ($P(c)$) 表示代码属于类别 c 的概率。

整个从结构层面进行特征提取的结构流程描述了从代码文件输入到最终的分类概率输出的完整过程。在训练阶段，通过优化图卷积网络和全连接层的权重和偏置项，可以最小化分类错误，提高模型的预测准确性。

1.1.4 **置信度融合层**

通过将上述三个不同维度特征方向的模型进行聚合:

1. 阈值判断:

对每个模型的置信度输出应用阈值函数 ，阈值设定为 0.5，输出结果为 1（表示检测为 Webshell）或 0（表示未检测为 Webshell）。

$$R_{\{BERT\}} = \Theta\left(P_{\{BERT\}(c)}\right)$$
$$R_{\{CNN\}} = \Theta\left(P_{\{CNN\}(c)}\right)$$
$$R_{\{GCN\}} = \Theta\left(P_{\{GCN\}(c)}\right)$$

其中，阈值函数$\Theta(x)$定义为:

$$\Theta(x) = \begin{cases} 1, & \text{if } x \geq 0.5 \\ 0, & \text{if } x < 0.5 \end{cases}$$

2. 串行判断逻辑:

根据阈值判断的结果 $R_{\{BERT\}}$，$R_{\{CNN\}}$，和 $R_{\{GCN\}}$ 应用串行判断逻辑 L 来得到最终的判定结果 O 。

$$[O = L(R_{\{BERT\}}, R_{\{CNN\}}, R_{\{GCN\}})]$$

串行判断逻辑 L 定义如下:

$$L(R_{BERT}, R_{CNN}, R_{GCN}) = \begin{cases} 1, & \text{if } R_{BERT} = 1 \text{ or } (R_{BERT} = 0 \text{ and } R_{CNN} = 1) \\ 0.5, & \text{if } R_{BERT} = 0, R_{CNN} = 0 \text{ and } R_{GCN} = 1 \\ 0, & \text{if } R_{BERT} = 0, R_{CNN} = 0 \text{ and } R_{GCN} = 0 \end{cases}$$

整个流程的公式化表达可以概括为:

$$O = L\left(\Theta(P_{\{BERT\}(c)}), \Theta(P_{\{CNN\}(c)}), \Theta(P_{\{GCN\}(c)})\right)$$

这个表达式描述了从代码文件输入到最终的分类判定结果的完整过程。在实际应用中，这个流程涉及到对三个模型的输出进行集成和决策，以提高整体的检测准确性和鲁棒性。

三. 恶意软件检测模型设计流程图

本作品提出的模型通过在 RoBERTa 层后进行序列的词嵌入，通过引入注意力机制计算序列中词向量的权重并依据权重大小所占比例进行高权重的词向量保留与低权重词向量的压缩，再将数据输入多维度混合模型，分别提取浅层（Liner）特征和深层特征中的局部依赖（TextCNN）和长期依赖（Att-Bi-LSTM），最后将特征合并输入全连接层分类预测。恶意软件检测模型流程图如下:

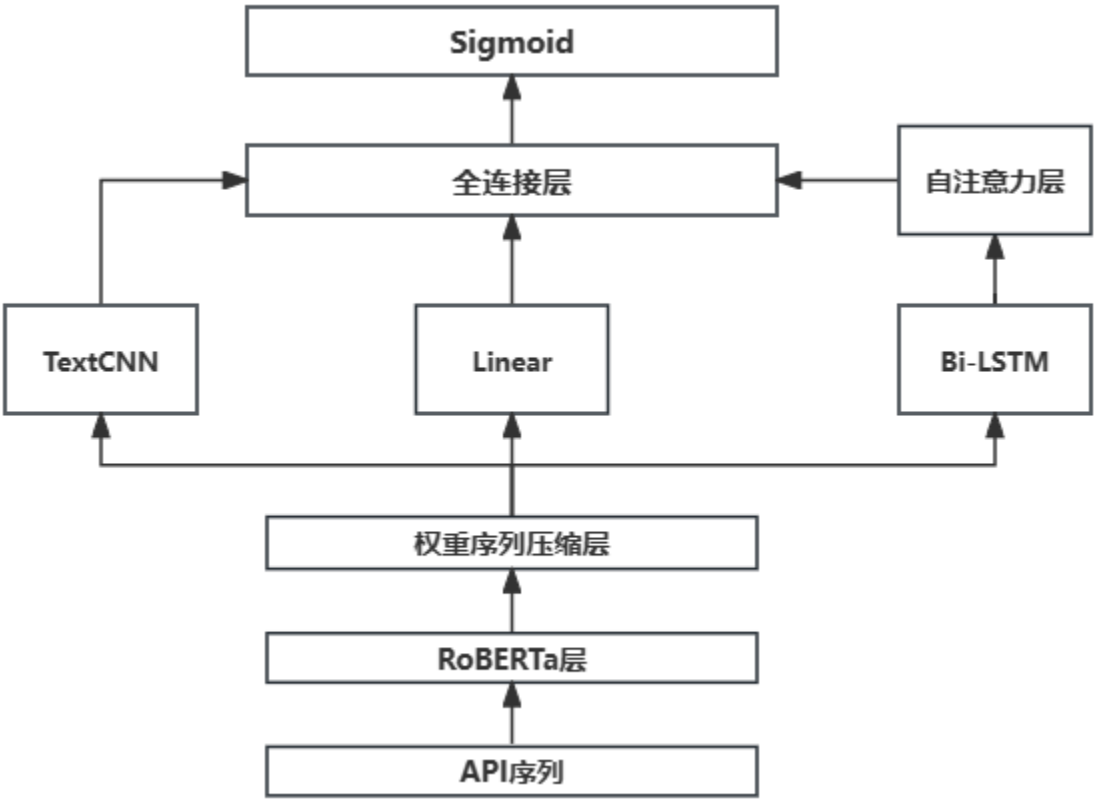


图 2 恶意软件检测模型流程图

2.1 恶意软件检测模型结构介绍

2.1.1 词嵌入层

本层需要将 API 序列转化为可以输入模型训练的词向量，因为是英文函数词的 API 调用序列检测任务。对于传统方法而言，首先要进行的就是分词操作，但是基于分词工具对 API 序列中出现的恶意 API 函数名处理效果差从而导致分词正确性低，从而影响分类正确性。而本作品中使用的 RoBERTa 模型是以字符为单位进行嵌入操作对比之下有着以下优势：强大的理解上下文能力，BERT 是一种基于 Transformer 的模型，它具有能够理解词在句子中的上下文语义的功能。这对于理解不同 API 调用序列中的细微差别非常重要；RoBERTa 是也是一种在大量文本数据上的预训练模型，能够捕捉语言的通用特征；RoBERTa 还有一大功能是微调，但是计算量大，基于现有的计算资源并没有进行此功能的使用。将文本数据 Texts=$\{T_1, T_2, \ldots, T_N\}$输入 RoBERTa 模型，设每一个文本长度为 n，经过处理得到一个该文本的词向量矩阵$\{e_1, e_2, \ldots, e_n\}$，其中矩阵大小为 n*w，其中 w 为每个字的维度大小，最后将 BERT 模型的输出矩阵 E 输入注意力层。

2.1.2 特征捕捉层

深层特征

TextCNN 模型

TextCNN 通过卷积层有效地捕捉局部特征（如词组或短语的模式），这对于理解文本结构和含义至关重要。不同大小的卷积核可以捕捉不同长度的依赖关系，增加了模型的灵活性和表现力。

卷积操作中的参数共享机制使得模型在处理大规模文本数据时更加高效，因为它减少了模型参数的数量，从而降低了过拟合的风险和计算复杂度。

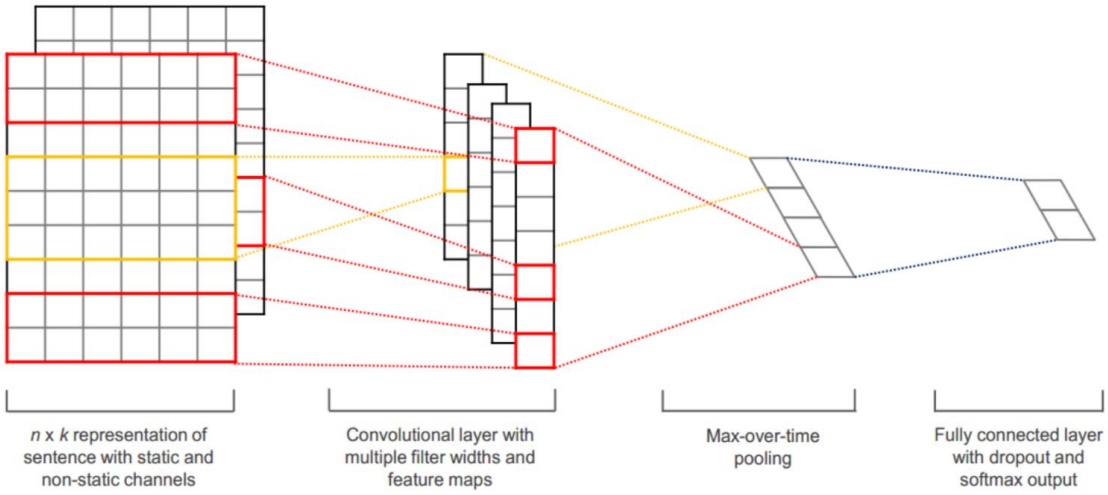


图 3 TextCNN 过程

输入张量：

$$X_{\text{cnn}}\text{:形状为}B \times L \times D$$

首先经N过个卷积层，每个卷积层的核大小为F_i（i从 1 到N）

对于每个过滤器，卷积操作可以表示为：

$$ci = f(E * Fi + bi)$$

f 是非线性激活函数 ReLU

对于每个卷积特征图池化操作可以表示为：

$$c^i = \max(c^i)$$

所有卷积层的输出被拼接在一起，形成最终的特征向量V_{cnn}。

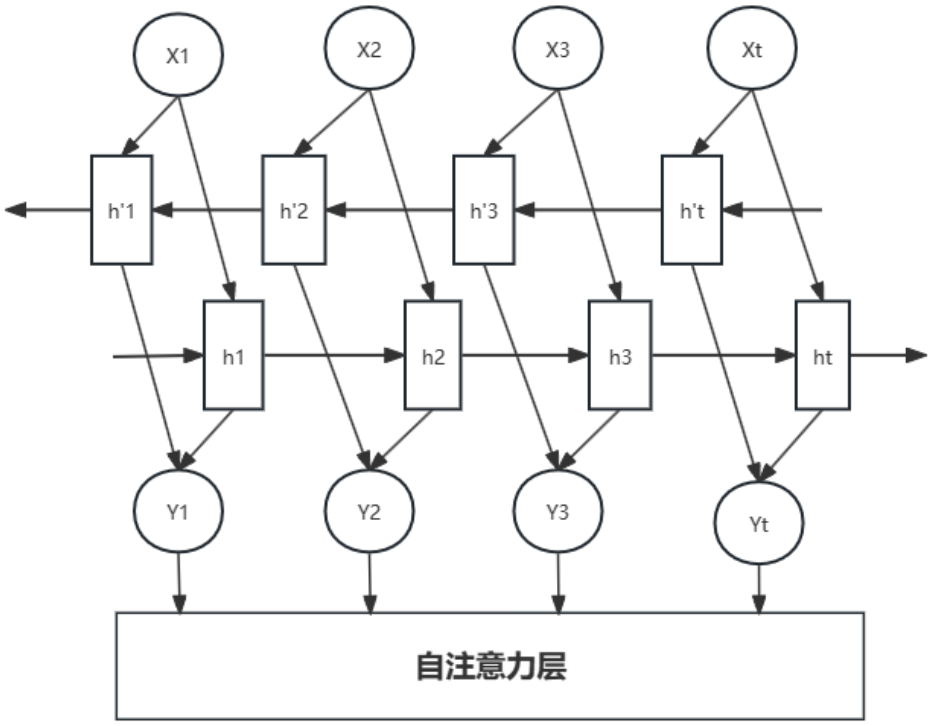
模型输出

V_{cnn}：TextCNN 模型的输出，形状为 $B \times (\sum_{i=1}^{N} F_i)$

Att-BiLSTM 模型

传统循环神经网络会存在的梯度消失或爆炸问题，因此本作品选用 Att-BiLSTM 模型来进行长期文本特征捕捉，它由两个 LSTM 组成，一个处理序列的正向（从开始到结束），另一个处理反向（从结束到开始）。这使得每个时间点的输出都包含了前文和后文的信息，从而提供了对整个输入序列更全面的理解。具有高效的长距离依赖捕捉能力 LSTM 设计用于解决传统 RNN 难以学习长距离依赖的问题。通过引入门控机制（遗忘门、输入门和输出门），LSTM 能够有效地保留长期信息，而 Bi-LSTM 进一步加强了这一能力，因为它从两个方向捕捉信息。

图 4 Att-BiLSTM 过程

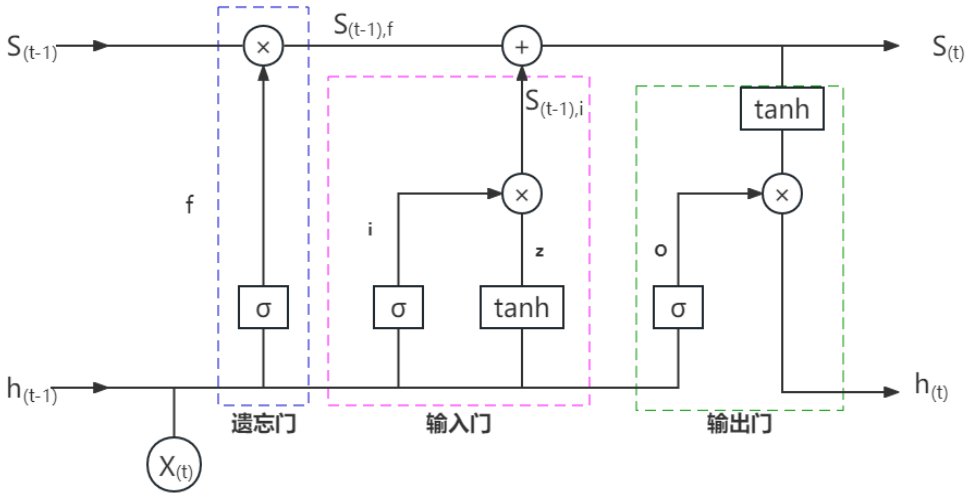


输入张量

X_{lstm}:形状为$B \times L \times D$，通过一个双向 LSTM 层处理，LSTM 的隐藏层维度为H。

其中一个 LSTM 的结构处理流程:

图 5 LSTM 结构



遗忘门(Forget Gate):

$$ft = \sigma\left(Wf \cdot \left[\hbar t - 1, xt\right] + bf\right)$$

输入门 (Input Gate):

$$it = \sigma\left(Wi \cdot \left[\hbar t - 1, xt\right] + bi\right)$$

$$c_t = \tanh\left(Wc \cdot \left[\hbar t - 1, xt\right] + bc\right)$$

细胞状态更新门 (Cell State Update):

$$c_t = ft \odot ct - 1 + it \odot c_t$$

输出门 (Output Gate):

$$ot = \sigma\left(Wo \cdot \left[\hbar t - 1, xt\right] + bo\right)$$

$$\hbar t = ot \odot \tanh(ct)$$

输出:

X_{lstm}:Bi-LSTM 模型的输出,形状为$B \times L \times 2H$,最后取最终的隐藏状态,形状为$B \times 2H$。

最后将 Att-Bi-LSTM 中双向的时间步信息通过计算自注意力权重,对每个时间步进行加权融合,突出不同时间步的重要程度,最后进行平均池化以方便后续输入到全连接层进行分类检测。

浅层特征:

Liner 线性层

将注意力层处理后的序列数据,首先在时间步上采用全局平均池化以匹配深层特征捉的维度大小,然后将维度大小为形状为$B \times L \times D$输入一个线性层中。

$$y = Wx + b$$

x是输入维度向量,维度为 D

W是权重矩阵维度为$D \times 256$

b是偏置向量,维度设置为 256

y是偏置向量,维度设置为 256

从而实现了降维来达到对文本中的浅层特征捕捉的效果。

四． 实验与分析

3.1 实验数据集

对于其中 Webshell（网页恶意后门）检测模型的训练数据集为 Github 中用户发布收集的名为“Webshell Detect DataSet”开源恶意后门和正常代码文件，总量约为 1.6w，二分类任务，正常和恶意代码文件数量各占一半。

对于恶意软件的检测模型的训练数据集为国外 MALWAREbazaar 网站中，已经有明确数据标注的各个家族的恶意代码种族例如 AgentTesla、PureLogStealer、Privateloader（商业窃密木马）、DCRAT，AsyncRAT，Netsupport（远控木马家族类）、Worms（蠕虫病毒）、Coinminer（挖矿家族）、MassLogger，Formbook，Vidar（窃密木马家族类）、Mirai（僵尸网络）、CobaltStrike（黑客工具）、Lesso（勒索病毒）这些家族类别恶意软件各 150 个。

正常软件数据集来源以 Github 中开源发布与开放数据集，以及开源代码自行编译生成的 exe 文件。

3.2 实验环境和参数设置

本实验在 Pycharm 中进行代码编写，编程语言选择 Python 3.10.0 版本，深度学习框架选择 PyTorch，CPU 版本为 8 核 I7-11870H，GPU 版本为 RTX 3050TI laptop，共 4 GB 显存。

3.2.1 Webshell 检测模型参数

卷积神经网络：特征图尺寸为 128x128，包含五个卷积层，每个卷积核大小为 3x3。使用 2x2 的最大池化核，步长为 2，以及 LeakyReLU 激活函数。输出层使用 Sigmoid 激活函数，模型训练了 50 轮，使用 Adam 优化器，学习率为 0.0005。

BERT 模型:使用`bert-base-uncased`模型和`BertTokenizer`进行分词。特征提取基于 BERT 最后一个隐藏层的[CLS]标记。分类器模块包括两个全连接层（fc1： 768->128， fc2: 128->64），输出维度为 1，同样采用 LeakyReLU 激活函数和 Sigmoid 输出层，训练设置与卷积神经网络相同。

GCN 模型：节点词嵌入使用 Word2Vec，输入维度为 Word2Vec 词嵌入后的 size 大小，输出维度为 48，中间隐层维度为 64，同样采用采用 LeakyReLU 激活函数和 Sigmoid 输出层，训练设置为 110 轮。

3.2.2 恶意软件检测模型参数

模型使用 RoBERTa 进行词嵌入，词向量的维度设为 768。其中模型流程中，TextCNN 模型中的卷积核的高度分别为 2、3 和 4，卷积核宽度与词向量维度相同，卷积核的数量为 100。 Att-Bi-LSTM 层中的隐藏单元个数为 128，dropout 的参数大小设置为 0.5。每次训练的批次 batch_size 大小为 32，学习率大小为 0.0005，epoch 数为 30。

3.3 模型指标分析

3.3.1 Webshell 检测模型指标分析

在 Webshell 检测任务中，我们采用了三种不同维度特征的模型：基于字符级图像的卷积神经网络（CNN），基于文本序列语义信息的 BERT 模型，以及捕捉代码结构特征的图卷积网络（GCN）。以下是每种模型单独使用时的性能指标，以及它们融合后的改进。

单独模型性能：

1. CNN 模型
   - 准确率：95.47%
   - 召回率：95.34%
   - F1 分数：95.39%

CNN 模型通过将代码转换为图像数据并使用卷积层捕捉局部模式，表现出较强的特征提取能力，尤其适合识别基于模式的攻击载荷。

2. BERT 模型
   - 准确率：98.34%
   - 召回率：98.04%
   - F1 分数：97.97%

BERT 模型通过其深层次的语义理解能力，能够很好地理解代码的上下文含义，从而在检测那些依赖于语义理解的 Webshell 时表现出色。

3. GCN 模型

对于正常文件识别准确率：

- 准确率：99.61%

GCN 模型通过分析代码的抽象语法树（AST）来捕捉函数调用和控制流的结构信息，并将其放置最后一层，作为检测的最后一关，能够对那些结构复杂、依赖于特定代码结构的 Webshell 发挥出特殊的检测效能。

4.融合模型性能

对于验证的恶意代码识别准确率：

- 准确率：99.55%

对于验证的正常代码识别准确率：

- 准确率：98.91%

融合模型结合了 CNN 的局部特征捕捉能力、BERT 的深层语义理解和 GCN 的结构信息分析。通过综合这些模型的优势，我们能够创建一个更为全面的检测系统，它能够有效地检测出各种类型的 Webshell，包括那些单一模型难以识别的变种。

通过对比分析，我们可以看到融合模型尤其在对恶意后门识别的准确率上都优于单独的模型，对于召回率指标也明显高于单独模型也大幅减少了误报的可能。这证明了融合不同特征层面的检测方法是提高 Webshell 检测准确率的有效手段，以应对不断演变的 Webshell 攻击手段。

3.3.2 恶意软件检测模型指标分析

本作品模型提出是在不断进行模型准确率对比上进行改进与提升，故将本作品通过图表的形式与以下的模型进行对比。

(1) FastText

FastText 是一个高效的文本分类和词嵌入学习库，采用子词信息和层次 softmax，适用于大规模数据集，广泛应用于情感分析、主题分类等任务。

(2) TextCNN

TextCNN 是一种基于卷积神经网络（CNN）的文本分类模型。它通过将文本转换为词向量序列，然后使用多个不同大小的卷积核在序列上进行卷积操作，捕捉局部特征。TextCNN 能有效处理文本数据的空间结构，适用于情感分析、主题分类等任务，表现出良好的分类效果。

(3) Att-BiLSTM

Att-BiLSTM 是一种结合了双向长短期记忆网络(BiLSTM)和注意力机制(Attention)的文本分类模型。BiLSTM 能够捕捉文本中的前后文信息，而注意力机制能够自动识别文本中最重要的部分，为分类提供更加精准的特征表示。

(4) BERT

BERT 是一种基于 Transformer 的预训练语言模型，通过双向自注意力机制捕获文本的深层语义特征。在文本分类任务中，BERT 可以通过微调预训练模型来适应具体任务，从而实现高效的特征提取和分类。

模型	准确率
FastText	84.97%
TextCNN	89.64%
Att-BiLSTM	90.47%
BERT	92.71%
TextCNN-Liner-Bi-LSTM	94.31%

图 6 模型分析

本作品提出的 TextCNN-Liner-Bi-LSTM 模型在测试数据集上分别获得了 94.31%准确率是所有进行实验模型中准确率最高的模型方法。对于前 4 个模型中，准确率最高的模型 BERT 模型中取得的 92.71%准确率，TextCNN-Liner-Bi-LSTM 模型在测试数据集上的准确率提升了 1.60%，从而证明了 TextCNN-Liner-Bi-LSTM 混合模型先通过捕捉浅层特征，再分别捕获深层文本特征下多粒度的局部特征和全局语义特征，从而充分利用了 API 序列中的多维度的特征，从而有效提升了模型的准确率。

五. 结论与展望

在本作品中,提出了两个基于多维度特征加强的恶意代码检测模型,分别针对 Webshell 网站后门和包括木马、蠕虫、挖矿软件在内的恶意软件。这些模型的设计理念是结合不同维度的特征来提升检测的准确性和鲁棒性。

1.Webshell 检测模型: 该模型通过内容特征的图像化处理、语义特征的 BERT 模型分析, 以及结构特征的图卷积网络学习, 实现了对 Webshell 后门的高效检测。特别是, 我们采用了一种置信度融合策略, 通过阈值判断和串行逻辑判断, 进一步提高了检测的准确率。

2.恶意软件检测模型: 针对恶意软件的检测, 我们引入了基于 API 调用序列的检测方法, 并使用 RoBERTa 模型进行动态词嵌入, 以捕捉 API 序列中的语义关系。此外, 我们提出了一种序列压缩算法, 以解决因重复 API 调用导致的序列过长问题。最终, 我们设计了一个多维度特征加强的 TextCNN-Liner-Att-BiLSTM 模型, 该模型通过结合浅层特征和不同粒度的深层特征, 显著提升了检测性能。

本作品的主要在于提出了一种结合多种特征的检测框架, 该框架能够有效应对当前恶意代码的多样性和复杂性。此外, 我们还提供了一种新的视角, 即通过融合不同维度的特征来增强检测系统的性能。

尽管本作品取得了积极的成果, 但仍有进一步改进的空间。未来的工作可以包括但不限于: 探索更多的特征融合技术, 优化模型结构以适应特定类型的恶意代码检测, 以及在更大规模的数据集上验证模型的泛化能力。