基于动态词向量和图卷积的 WebShell 检测方法

翟江涛[1,2], 王 涛[1,2], 周 桥[1,2], 董 燚[1,2], 王子豪[1,2]

[1]南京信息工程大学 电子信息与工程学院 南京 中国 210044
[2]复杂环境智能保障技术教育部重点实验室 南京 中国 210044

摘要 随着数字化转型的加速, Web 服务器已成为网络攻击的重灾区。WebShell 是一种通过 Web 脚本编写的恶意程序, 它允许攻击者远程控制服务器执行非法活动, 对网络安全构成了严重威胁。现有的 WebShell 检测方法由于难以处理 WebShell 代码的复杂性和多样性, 因此无法达到理想的检测效果。为了解决现有方法在处理代码的多义性、上下文理解及结构特征识别方面的不足, 本文提出了一种 WebShell 检测方法 JLBertGCN。该方法采用 Bert 模型生成动态词向量, 同时结合图卷积网络(Graph Convolutional Network, GCN), 将代码视为图结构进行特征提取。具体而言, Bert 模型能够从大规模未标注的文本中学习到通用的语言表示, 特别是在捕捉词汇的多义性和上下文关系方面表现出色。GCN 则通过捕捉文本中的不规则结构和语义信息, 进一步增强了模型的特征提取能力。此外, 本文还采用了双通道联合训练策略, 有效结合了 Bert 和 BertGCN 的输出, 提高模型检测性能。通过这种双通道联合训练策略, 模型能够充分利用 Bert 的预训练能力和 GCN 的图结构学习优势, 形成精确的文本表示。实验结果表明, JLBertGCN 方法在 WebShell 检测任务上的准确率和 F_1 分数分别达到了 99.26%和 99.05%, 显著优于现有方法。这表明, 本文提出的方法在处理复杂的代码语义和结构特征方面表现优越, 能够有效应对高度混淆和动态变化的 WebShell 攻击场景。JLBertGCN 在准确识别 WebShell 的同时, 具有较强的泛化能力和鲁棒性, 为提升网络安全提供了新的思路和技术手段。

关键词 WebShell; 深度学习; 动态词向量; 图卷积网络; 联合训练
中图法分类号 TP393.08 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2026.01.04

WebShell Detection Method Based on Dynamic Word Embeddings and Graph Convolution

ZHAI Jiangtao[1,2], WANG Tao[1,2], ZHOU Qiao[1,2], DONG Yi[1,2], WANG Zihao[1,2]

[1] School of Electronics & Information Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China
[2] Key Laboratory of Intelligent Support Technology for Complex Environments, Ministry of Education, Nanjing 210044, China

Abstract With the acceleration of digital transformation, Web servers have become a major disaster area for cyber attacks. WebShell is a malicious program written through Web scripts, which allows attackers to remotely control servers to perform illegal activities, posing a serious threat to network security. Existing WebShell detection methods cannot achieve ideal detection results because they are difficult to handle the complexity and diversity of WebShell codes. In order to address the shortcomings of existing methods in dealing with the ambiguity of code, context understanding and structural feature recognition, this paper proposes a WebShell detection method JLBertGCN. This method uses the Bert model to generate dynamic word vectors, and combines the graph convolutional network (GCN) to treat the code as a graph structure for feature extraction. Specifically, the Bert model can learn a general language representation from large-scale unlabeled text, especially in capturing the ambiguity and contextual relationship of vocabulary. GCN further enhances the feature extraction ability of the model by capturing the irregular structure and semantic information in the text. In addition, this paper also adopts a dual-channel joint training strategy to effectively combine the outputs of Bert and BertGCN to improve the model detection performance. Through this dual-channel joint training strategy, the model can fully utilize the pre-training ability of Bert and the graph structure learning advantages of GCN to form an accurate text representation. Experimental results show that the accuracy and F_1 score of the JLBertGCN method on the WebShell detection task reached 99.26% and 99.05% respectively, which is significantly better than the existing methods. This shows that the method proposed in this paper is superior in processing complex code semantics and structural features, and can effectively deal with highly obfuscated and dynamically changing WebShell attack scenarios. While accurately identifying WebShell, JLBertGCN has strong generalization ability and robustness, providing new ideas and technical means for improving network security.

Key words WebShell; deep learning; dynamic word embeddings; GCN; joint training

通讯作者: 翟江涛, 博士, 副教授, Email: jiangtaozhai@nuist.edu.cn。
本课题得到国家重点研发计划(No. 2021QY0700)和国家自然科学基金(No. U21B2003, No. 62072250)资助。
收稿日期: 2024-06-24; 修改日期: 2024-07-18; 定稿日期: 2025-11-11

1 引言

在信息时代, 网络安全已成为全球关注的焦点之一。特别是随着网络攻击技术的快速发展, 网络安全面临的挑战不断增加。WebShell 作为一种常见的网络攻击工具[1], 由于其隐蔽性强和破坏力大, 成为攻击者广泛利用的手段之一。WebShell 使攻击者能够通过网络远程控制受感染的服务器, 执行各种恶意行为, 如数据窃取、恶意代码执行等。因此, 开发有效的 WebShell 检测技术对于提高网络安全具有重要意义。

自从 2017 年《中华人民共和国网络安全法》实施以来, 虽然国内的网络环境有所改善, 但国际网络环境复杂, 跨国网络攻击事件层出不穷, WebShell 攻击的检测与防范问题日益突出。研究表明, 现有的 WebShell 检测方法虽然在某些方面取得了进展[2], 但仍然面临诸多挑战, 例如检测性能不高、适用性不强以及对新型和复杂 WebShell 的检测能力不足。

针对这些挑战, 本文提出了 JLBertGCN 检测方法, 这是一种结合了动态词向量和图卷积网络的 WebShell 检测模型。通过运用 Bert 模型生成动态词向量以增强对 WebShell 代码语义的理解, 并利用 GCN 在图结构上捕捉代码的结构特征, 该方法旨在有效区分恶意 WebShell 文件与正常脚本。此外, 该检测方法还采用了双通道联合训练策略, 将 Bert 和 BertGCN 的优势进行融合, 进一步提升了检测精度和效率。

2 相关工作

WebShell 检测是指在 Web 服务器上发现和识别恶意的 Web 后门脚本, 防止黑客通过 WebShell 执行远程命令或者窃取敏感数据。针对 WebShell 检测的研究已有很多年的历史, 研究者提出了多种检测方法和工具, 以下分别从基于动态分析和基于静态分析的方法两个方面介绍 WebShell 检测领域的相关研究现状。

动态检测方法通过监控 WebShell 攻击发生时系统行为和网络流量的变化来检测 WebShell。在动态检测方面, Yu 等[3]利用传输层相关行为特征和加密协议握手阶段的明文特征, 用堆叠自编码器检测 WebShell 攻击。Le 等[4]通过捕获和分析网络流量, 并与已知签名进行匹配, 实现 WebShell 检测和可追溯性。Sasikala 等[5]提出了一种结合随机森林(Random Forest, RF)算法和极限梯度增强算法(Extreme Gradient Boosting, XGBoost)对流量内容进行分析的 WebShell 检测模型, 实现了比黑名单更优越的检测性能。动态检测方法不依赖于 WebShell 文件的特定特征, 有效规避了代码重写、伪装和混淆带来的检测难题。然而, 这种方法需要持续监控网络流量的变化, 不仅维护成本高, 而且可能将正常的网络活动误判为异常, 导致误报。且只能检测出已经被触发的 WebShell 攻击, 不能检测出未被触发的 WebShell 文件, 也不能检测出其他类型的 Web 攻击。

静态检测方法是当前 WebShell 检测的主流方法, 重点在于分析 WebShell 文件本身的特征来实现检测。在传统的静态检测方面, Li 等[6]基于随机森林算法开发了一种 WebShell 检测模型, 该模型通过分析文件的语法和语义特征来识别 WebShell。此外, PHP 操作序列码(Operation code, Opcode)也被作为检测 WebShell 文本的重要特征[7-8], 利用 PHP 源文件的统计特征, 包括信息熵和重合指数, 并将其与 Opcode 合并, 以提高 WebShell 检测效率。Yong 等[9]提出了一种集成学习的 WebShell 检测方法。将 PHP 代码转换为 Opcode 后, 利用词袋模型和词频率-逆文档频率(Term Frequency-Inverse Document Frequency, TF-IDF)法选择高频可识别词, 该算法的准确率、F_1 值和查全率均超过 96%, 但表现仍可以进一步提高。Fang 等[10]提出了一种结合 WebShell 静态特征和 Opcode 的 WebShell 检测方法, 将 WebShell 文本的最长单词长度、信息熵等统计特征与 FastText 模型建模的 Opcode 相结合输入到随机森林算法进行检测, 该方法获得了较高的准确率, 但处理过程复杂, 且不能用于除 PHP 语言以外的 WebShell。

传统的静态检测方法依赖规则库, 不能有效地检测变化多样的新型 WebShell, 且需要人工频繁更新规则库。基于深度学习的静态检测方法, 近年来在 WebShell 检测领域有了广泛的应用。Liu 等[11]从 PHP 代码中提取 Opcode 序列重要特征, 同时结合静态统计特征构造特征矩阵, 用支持向量机(Support Vector Machine, SVM)、卷积神经网络(Convolutional Neural Network, CNN)等分类模型检测 WebShell。Lv 等[12]用 CNN 检测 WebShell, 先对代码进行分词处理, 再用 Word2Vec[13]生成词向量, 该方法不用人工特征, 但实际表现较差。Zhou 等[14]提出了一种基于循环神经网络(Recurrent Neural Network, RNN)的 WebShell 检测方案, 采用两层门控循环单元(Gated Recurrent Unit, GRU)结构, 实验结果表明, 该方案比 CNN 具有更高的准确率。章嘉昊[15]针对 WebShell 特征挖掘过程中文本语义信息考虑不充分的问题, 提出了基于深度过参数化卷积的改进文本卷积神经网络(TextCNN)算

法, 有效提升了检测性能。Liu 等[16]提出了一种基于双向 GRU 和注意力机制的 WebShell 检测方法, 应用于多种语言的 WebShell 脚本, 取得了较高的检测准确率。

与传统的机器学习方法相比, 基于深度学习的解决方案可以获得更高的准确率和召回率, 并具有识别新样本的能力, 但目前的研究仍然存在特征挖掘和构建不充分的问题。为了解决现有 WebShell 检测技术在提取文本语义和结构特征方面的不足, 本文提出了基于动态词向量和图卷积的 WebShell 检测方法来提高检测性能。

3 方法设计

3.1 整体框架

本文提出的 JLBertGCN 模型架构如图 1 所示, JLBertGCN 首先使用 Bert 预训练模型处理 WebShell 文本数据, 生成初始向量表示。生成的初始向量表示随后被用于构建一个包含脚本节点和单词节点的异构图, 作为 GCN 分类模型的输入。接着, 通过两层 GCN 层在异构图上迭代更新脚本表示, 从而捕捉脚本间复杂的关系。最后使用 Softmax 分类器进行预测。

本文提出的 JLBertGCN 模型的框架中, 特别强调了联合训练的应用。联合训练是一种集成学习方法, 旨在通过将多个模型或任务在一个优化目标下进行训练, 以提高模型在不同数据上的性能。具体地, 将基于 Bert 模块的分类预测和基于 BertGCN 模块的分类预测分别输入到带有 Softmax 激活函数的全连接层中, 然后通过线性插值方法对这两个预测结果进行加权平均。这样可以利用 Bert 模块和 BertGCN 模块各自的优势, 优化训练模型参数。

3.2 WebShell 数据预处理

鉴于目前缺乏用于 WebShell 检测的标准数据集, 本文采取了多项措施构建了一个适合研究用途的数据集。该数据集主要源自 GitHub 等开源平台, 其中包含了大量的 WebShell 样本。为确保数据的多样性, 本文还从其他开源的服务器端项目中收集了正常的样本。

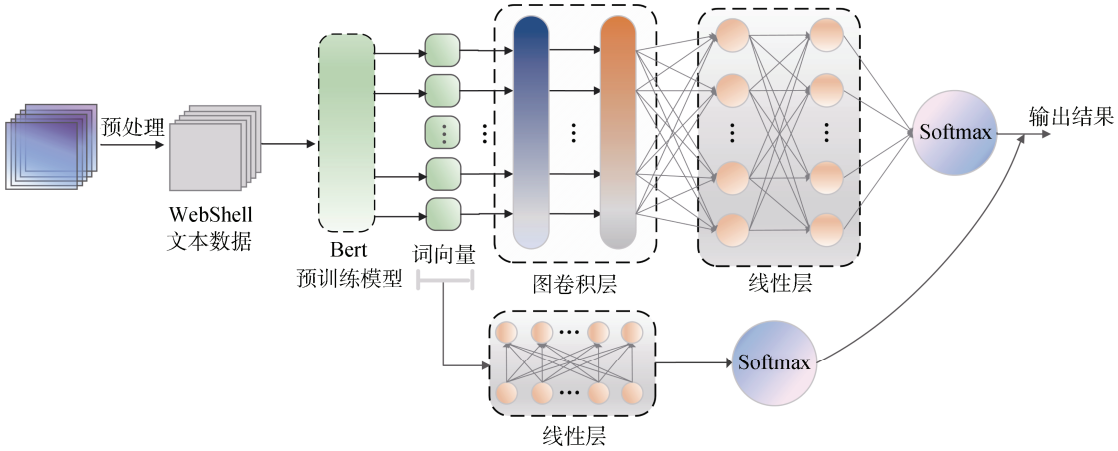

图 1 JLBertGCN 模型架构图

Figure 1 JLBertGCN model architecture diagram

收集的 WebShell 文本数据主要包括 3 种类型的服务器端脚本语言文件: PHP、ASP 和 JSP。这些样本不仅涵盖了大多数已知的 WebShell, 而且相同功能的 WebShell 会以不同的编程语言呈现或者具有不同的版本, 充分展示了 WebShell 的多样性和复杂性。

本文对收集的 WebShell 和正常脚本文件进行了数据去重工作, 具体分为两个步骤。首先, 使用 MD5 哈希算法对内容完全相同的文件进行去重。接着, 针对 PHP 语言的数据集, 使用 Vulcan 逻辑反汇编器(Vulcan Logic Disassembler, VLD)生成 Opcode, 并对 Opcode 相同的文件进行去重。在完成数据去重后, 为了平衡正常样本和 WebShell 样本的数量以避免模型训练过程中的偏差, 本文使用随机下采样技术从正常样本中随机选取与 WebShell 样本相同数量的样本, 并将这些经过处理的不同脚本文件集合合并成一个全面的数据集。

随后本文对这些脚本文件进行预处理, 涉及数据清洗和分词两个关键步骤。本文在数据清洗环节去除了那些不影响代码逻辑理解的元素(如注释、多余的空格和特殊格式符号)以减少引入不必要的噪声, 并将所有字母转换为小写形式。在自然语言处理中, 分词是一项基础且关键的操作, 它将文本中的长句子或段落拆解为单个词汇, 从而使数据结构更适合后续处理和分析。恰当的分词不仅可以保留必要的

信息量, 也便于信息的提取和应用, 这对文本数据的理解和分析至关重要。

本文使用 Basic Tokenizer 与 Wordpiece Tokenizer 进行分词, 首先, 通过 Basic Tokenizer 去除文本中的非法字符和多余空格, 并按空格进行初步分词, 形成初级词汇表。接着, 使用 Wordpiece Tokenizer 进一步细化词汇, 将词汇表中的 token 分割成更小的子单元。例如, 若“WebShell”和“Shell”都存在于词汇表中, 那么“WebShell”将被分割为“Web”和“Shell”。分词完成后, 这些标准化的数据将被向量化, 使其可以被模型有效地进行处理和学习。

3.3 动态词向量构建

本文使用 Bert 方法来构建词向量。Bert 是由 Google 在 2018 年提出的一种基于深度学习的方法, 它通过利用 Transformer[17]多层自注意力机制, 从大量未标注的文本中学习到通用的语言表示。Bert 的特点在于它能够同时处理单词左右两侧的上下文信息, 生成更丰富和准确的词向量。

与传统的基于统计的方法(如词袋模型、TF-IDF 等)和基于预测的方法(如 Word2Vec 等)相比, Bert 的优势在于其能根据不同的任务和数据动态地调整词向量。传统的基于统计的方法通常只能捕捉词的表层特征, 例如出现频率和共现关系, 忽略了词序和深层语义信息。而基于预测的方法虽能捕捉词的语义和语法信息, 但这些方法通常是单向的, 只能考虑单词的前向或后向上下文, 无法同时考虑两个方向的信息。Bert 则通过其创新的双向处理机制, 克服了这些局限, 提供了一种更全面和灵活的词向量生成方式。

如图 2 所示, Bert 模型的整体网络结构在数据处理过程中主要包含 3 个输入部分: 词汇嵌入向量、文本嵌入向量和位置嵌入向量。这些向量相加后构成了模型的最终输入。词汇嵌入是通过预训练将文本中的词汇转换为向量形式实现的。对于输入的 WebShell 脚本, 首先通过分词器进行分词, 将文本划分为基本的语义单元, 通常是单词或子单词。例如, WebShell 命令“system($command);”会被拆分为“system”、“(”、“$command”、“)”、“;”等基本单元, 并转化为向量。在构建输入序列前, 会添加两个特殊标记: [CLS]和[SEP]。[CLS]标记位于输入序列的开始处, 用于分类任务的输出, 而[SEP]标记用于分隔文本中的不同句子或命令, 有助于模型理解文本的结构。文本嵌入向量用于区分两个句子的向量表示, 在 WebShell 脚本的处理中, 这有助于模型区分不同的命令或函数调用。位置嵌入向量提供了词汇在脚本中的位置信息, 这使得模型能够更准确地理解词序和语法结构, 从而有效地解析控制流语句。

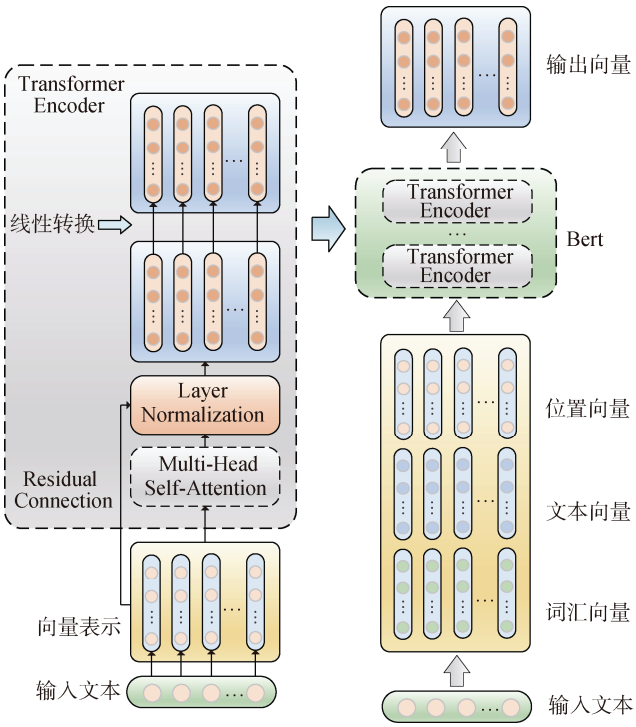

图 2 Bert 模型结构图

Figure 2 Bert model structure diagram

Transformer 编码器层是 Bert 模型的核心组成部分, 其由多个结构相同的 Transformer 编码器块构成。这些编码器块通过多头自注意力机制处理输入序列, 使模型能够同时关注序列中的多个位置, 从而更有效地理解复杂的 WebShell 脚本。每个 Transformer 编码器块主要包括以下几个关键组件: 多头自注意力层(Multi-Head Self-Attention)、残差连接(Residual Connection)、层归一化(Layer Normalization)以及线性转换层。这种设计不仅提高了模型对长距离依赖的捕捉能力, 还增强了模型在处理序列数据时的稳定性和效率。

多头自注意力层的作用是让模型在处理输入序列时关注输入的不同部分, 有助于捕捉长距离依赖关系。例如, 在处理 WebShell 脚本时, 多头自注意力可以有效地帮助模型理解$command 变量是如何在脚本中被赋值和使用的。在 Bert 模型中, 多头关注输入序列中不同的信息片段, 从而提高了模型对上下文的整体理解能力。多头自注意力机制的计算步骤主要包括以下几个关键部分:

(1) 线性映射: 输入序列首先通过 3 个线性映射处理, 分别得到查询(Q)、键(K)、值(V)的表示。这 3 个映射各自独立, 每个都具备自己的权重矩阵。经过

此步骤, 序列中每个位置的词汇都将得到 3 种不同的表示, 这一过程可以用式(1)表示, 其中 X 表示由输入序列构成的矩阵, W_Q、W_K 和 W_V 分别表示以上 3 个映射对应的权重矩阵。

$$Q = XW_Q, K = XW_K, V = XW_V \tag{1}$$

(2) 注意力计算: 在 Transformer 模型的多头注意力机制中, 每个头部分别计算自己的注意力权重矩阵。注意力权重矩阵是通过对查询 Q 和键 K 进行点积运算得出的, 接着将结果除以一个缩放因子 $\sqrt{d_k}$, 其中 d_k 是查询和键的维度。接着使用得到的注意力权重对值 V 进行加权求和, 最终得到自注意力的输出 Z, 其具体计算过程如式(2)所示。

$$Z = \text{Soft max}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

(3) 多头连接: 将多头注意力的输出拼接在一起, 并通过线性映射得到最终的输出。这一过程可用式(3)表示, 其中 Z_i 表示第 i 个头的输出, W_o 表示权重矩阵, h 表示注意力头的个数。

$$\text{MultiHead}(X) = \text{Concat}\left(Z_1, Z_2, \ldots, Z_h\right)W_o \tag{3}$$

残差连接和层归一化在 Bert 模型中的应用是至关重要的, 它们确保了即使在深层网络中, 每个编码器块也能有效地学习。这对于处理 WebShell 脚本中的复杂结构和语义尤为重要。在残差连接层中, 通过将输入与该层的输出相加, 使得模型能够缓解梯度消失问题, 从而提高训练过程的稳定性和效率。此外, 层归一化通过对每个编码器块的输出进行标准化, 有助于加速训练过程并提高模型的鲁棒性, 确保模型在不同数据和环境下都能保持良好的性能。其计算过程可用式(4)表示, 其中 x 表示 Sublayer 层的输入。

$$\text{LayerOutput} = \text{LayerNorm}(x + \text{Sublayer}(x)) \tag{4}$$

在式(4)中, LayerNorm 表示层归一化的操作, 它的定义如式(5)所示, 其中, $\boldsymbol{x}$ 表示输入向量, μ 和 σ 分别表示输入向量的均值和标准差, γ 和 β 分别表示可学习的缩放和偏移参数。

$$\text{LayerNorm}\left(x\right) = \frac{\gamma}{\sigma}\left(x - \mu\right) + \beta \tag{5}$$

最后的线性转换层对每个位置的向量执行全连接操作。该层主要负责将从前面层获得的复杂特征转换成最终的输出格式, 这有助于模型精确捕捉和处理文本数据的细微差异。通过这种线性变换, 模型能够将深层特征映射到更高维的空间, 增强其表达和分类的能力, 尤其是在处理复杂和多样化的输入如 WebShell 脚本时。具体计算过程如式(6)所示, 其中 $\boldsymbol{x}$ 表示输入向量, W_1、W_2、b_1 和 b_2 分别表示不同的权重矩阵和偏移向量, ReLU 表示线性整流单元, 是一种常用的激活函数。

$$FFN\left(x\right) = W_2\left(\text{ReLU}\left(W_1 x + b_1\right)\right) + b_2 \tag{6}$$

编码器层通过堆叠多个编码器块, 逐层提取输入序列的高级语义表示。这种架构不仅保留了每个位置的原始信息, 而且通过自注意力机制实现了全局上下文的建模。通过其预训练的 Transformer 模型, Bert 能够深入理解文本语境, 为 WebShell 脚本提供丰富且多维的向量表示。

3.4 基于图卷积模型的分类算法实现

在本文提出的 JLBertGCN 模型中, 使用 Bert 模型来初始化文本图中文档节点的表示, 这些表示随后被用作 GCN 的输入。GCN 对这些表示进行迭代更新, 基于图结构优化文档的嵌入表示。GCN 的输出作为文档节点的最终表示, 输入到 Softmax 分类器进行分类预测。具体而言, 本文借鉴了 TextGCN[18]的方法构建了一个融合了单词节点和文档节点的异构图。在构建的异构图中, 文档和单词间的边通过 TF-IDF 来定义, 而单词间的边则基于点对点互信息(Positive pointwise mutual information, PPMI)来确定, 两个节点之间的边的权重定义如式(7)所示。

$$A_{i,j} = \begin{cases} PPMI\left(i,j\right), & i,j \text{ are words and } i \neq j \\ TF\text{-}IDF\left(i,j\right), & i \textit{ is } \text{document}, j \text{ is word} \\ 1, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

在 TextGCN 中, 初始节点特征矩阵是一个身份矩阵, 即每个节点的特征向量中只有一个元素为 1, 其余元素为 0。这种设计确保了节点之间的差异性, 但却缺乏语义信息。为了克服这一缺点, TextGCN 通过 GCN 的多层传播学习节点的语义表示。相比之下, JLBertGCN 的初始节点特征矩阵由 Bert 模型生成的文档嵌入矩阵与一个全零矩阵拼接而成, 形成一个更丰富的特征表达, 即:

$$X = \begin{pmatrix} X_{\text{doc}} \\ 0 \end{pmatrix}_{(n_{\text{doc}} + n_{\text{word}}) \times d} \tag{8}$$

其中, X_{doc} 是由 Bert 模型对每个文档进行编码得到的嵌入矩阵, 维度为 $n_{\text{doc}} \times d$, n_{doc} 是文档节点的数量, d 是嵌入向量的维度。0 是一个全零矩阵, 维度为 $n_{\text{word}} \times d$, n_{word} 是单词节点的数量。然后, 将 X 输入到 GCN 模型中, 模型会在训练和测试示例之间迭代传播消息。具体来说, 经过多层图卷积操作, 得到每个节点的更新后的嵌入向量, 其中第 i 层的输出特征矩

阵 $L^{(i)}$ 通过式(9)得到:

$$L^{(i)} = \rho\left(\tilde{A}L^{(i-1)}W^{(i)}\right) \tag{9}$$

其中, ρ 是激活函数, $\tilde{A}$ 是归一化邻接矩阵, $W^{(i)}$ 是第 i 层的权重矩阵, $L^{(0)} = X$ 是模型的输入特征矩阵。

最后, 使用 GCN 输出作为文档节点的最终表示, 然后将其输入到一个 Softmax 层中进行分类, 如式(10)所示, 其中 g 表示 GCN 模型, X 表示 GCN 的输入特征矩阵, A 表示邻接矩阵。

$$Z = \text{Soft max}\left(g\left(X, A\right)\right) \tag{10}$$

为了训练 JLBertGCN 模型, 使用交叉熵损失函数来联合优化模型参数 θ, 即:

$$L(\theta) = -\sum_{y,\hat{y}} y \log \hat{y} \tag{11}$$

其中, y 是文档节点的真实标签, $\hat{y}$ 是模型的预测结果。

使用 GCN 使模型能够在代码数据中建模复杂的结构关系, 有助于捕捉文档之间的上下文信息。与 Bert 结合使用, GCN 使得 JLBertGCN 在 WebShell 脚本检测中更具优越性, 特别是在处理包含多义性和上下文依赖的脚本时。GCN 的图结构学习与 Bert 的深层次语境理解相互补充, 为复杂的代码文本数据提供了高效的表示和检测能力。

3.5 联合训练

联合训练是一种集成学习方法, 其目的是通过将多个模型或任务在一个优化目标下进行训练, 提高模型在不同数据上的性能。在联合训练中, 每个模型或任务都有自己的损失函数, 然后这些损失函数被组合在一起以形成一个总的损失函数。联合训练的主要思想是利用不同模型或任务之间的共享信息和特异信息, 从而提高模型的泛化能力和预测能力。这些模型或任务可以是不同类型的神经网络, 例如 CNN 和 RNN, 也可以是不同类型的机器学习任务, 例如分类、回归、聚类等。联合训练的优点是可以有效地整合多个模型或任务的预测结果, 从而提高预测的性能和稳定性。此外, 通过将多个模型或任务在一个优化目标下进行训练, 可以减少模型的过拟合问题。Hwang 等[19]提出了一种利用多个相关任务来进行机器学习的方法, 通过让一个模型在多个任务上同时学习, 来提高模型的效果。其实验表明, 联合训练可以使模型在语音识别、手写识别、自然语言理解等任务上都有显著的提升。Li 等[20]提出了联合训练来训练 GCN, 并通过大量实验验证了联合训练方法有助于多层 GCN 模型克服内在的如梯度消失或过度平滑等问题, 所提方法提高了 GCN 的学习效果, 使 GCN 有了更好的性能表现。

在本文方法的训练中, 使用联合训练的方法加速 JLBertGCN 模型的收敛速度和提高其性能, 利用辅助分类器和线性插值来联合训练 Bert 模块和 BertGCN 模块。具体地, 如图 3 所示, 方法将文档嵌入(表示为 X)直接输入到一个带有 Softmax 激活函数的全连接层, 得到一个基于 Bert 的分类预测(表示为 Z_{Bert})。同时, 将文档嵌入输入到 GCN 模块, 该模块后同样是一个带有 Softmax 激活函数的全连接层, 得到另一个分类预测(表示为 Z_{BertGCN})。为了综合两个模块的优势, 本章使用线性插值的方法, 将两个预测结果进行加权平均, 得到最终的分类预测, 如式(12)所示。

$$Z = \alpha Z_{\text{BertGCN}} + (1-\alpha) Z_{\text{Bert}} \tag{12}$$

其中, α 是一个动态调节的参数, 用于控制 Bert 模块和 BertGCN 模块的预测结果的权重。当 $\alpha = 1$ 时, 相当于只使用 BertGCN 模块; 当 $\alpha = 0$ 时, 相当于只使用 Bert 模块; 当 $\alpha \in (0,1)$ 时, 相当于使用了一个

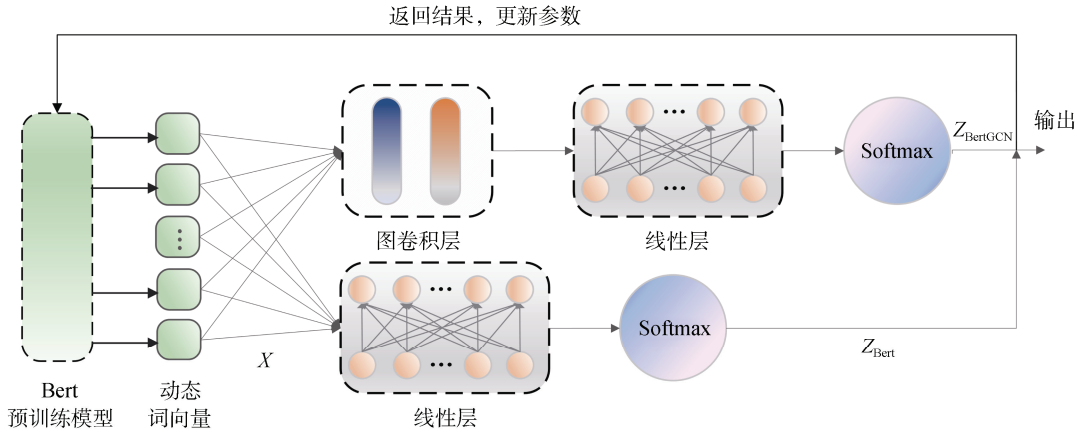

图 3 JLBertGCN 联合训练示意图

Figure 3 JLBertGCN joint training diagram

混合的模型。训练过程中, 通过动态调节 α 的值, 模型会找到一个最优的平衡点, 使得 JLBertGCN 模型能够在 WebShell 检测任务上达到最佳的效果[21]。这种优化方法可以使模型更好地利用 Bert 的大规模预训练能力和 GCN 图结构学习优势, 其学习策略为模型提供了额外的信息, 增强了 JLBertGCN 模型的泛化能力, 为模型提供了丰富的上下文信息, 从而提高模型的准确性和鲁棒性。

4 实验结果与分析

4.1 实验数据

为了全面评估 JLBertGCN 模型在 WebShell 检测任务中的性能, 本文构建了一个经过预处理的数据集。该数据集包含 WebShell 和正常脚本, 各类样本数量已经过均衡配置, 以确保实验结果的公正性和准确性。数据集的详细组成如表 1 所示, 表中详细展示了本文使用的不同类别脚本的样本数量。

表 1 数据集描述

Table 1 Dataset description

脚本文件类型	PHP	ASP	JSP
正常样本数	1119	434	403
WebShell 样本数	1119	434	403

4.2 实验设置

本文实验设置包含两个部分: 实验环境设置和训练参数设置。具体的实验环境如表 2 所示。

表 2 实验环境

Table 2 Experimental environment

	型号
处理器	Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz
内存	16GB
操作系统	Windows10, 64 位操作系统
显卡	Nvidia GeForce GTX1650
其他	Pytorch 1.9.1、Anaconda 3、Python3.7 等

本文采用了 Bert-base 作为 JLBertGCN 模型的预训练语言模型, 并设置 GCN 的层数为 2。为了评估不同 GCN 层数对模型性能的影响, 本文进行了 1~4 层的实验测试。经过对比分析, 使用两层 GCN 不仅能够提供优异的检测性能, 而且避免了更多层数带来的过拟合问题和更高的计算成本。此外, 为了确定最佳词向量长度, 本文在长度范围 100~400 中进行了一系列实验, 步长为 50, 并尝试了一些常用尺寸, 如 128 和 256, 结果表明在词向量长度为 256 时, 模型在数据集上的表现最佳。JLBertGCN 模型的其他训练参数详见表 3。

表 3 JLBertGCN 参数设置

Table 3 JLBertGCN parameter settings

参数名称	参数含义	参数值
Bert_init	预训练语言模型	Bert-base
batch_size	批处理大小	32
gcn_layers	GCN 的层数	2
n_hidden	GCN 隐藏层的维度	200
Bert_lr	Bert 的学习率	1e-5
gcn_lr	GCN 的学习率	1e-3
step_size	学习率调整时的步长	10
gamma	学习率衰减的乘数因子	0.1
optimizer	优化器	Adma
dropout	dropout 比例	0.5

4.3 实验标准

在本文中, 使用了准确率、精确率、召回率和 F_1 分数来评估分类器的性能。准确率衡量了分类器的总体表现, 精确率反映了分类器在每个类别上的正确识别能力, 召回率描述了分类器识别出的相关实例占总相关实例的比例, 而 F_1 分数是精确率和召回率的调和平均。在模型评估中, TP 表示正样本被正确地预测为正样本, TN 表示负样本被正确地预测为负样本, FP 表示负样本被错误地预测为正样本, FN 表示正样本被错误地预测为负样本。

准确率(Accuracy): 表示被正确分类的样本数量与总样本数量的比例。

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

精准率(Precision): 表示被正确识别为某一类别的样本数量与被预测为该类别的所有样本数量之比。

$$\text{Precision} = \frac{TP}{TP + FP} \tag{14}$$

召回率(Recall): 表示被正确分类为某一类别的样本数量与实际属于该类别的所有样本数量之比。

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

F_1 分数(F_1-score): 精确率和召回率的调和平均值, 用于评估模型的整体效能, 特别适用于类别不平衡的情况。

$$F_1 - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

4.4 消融实验

为了验证 Bert 动态词向量构建模块、GCN 模块

及联合训练策略在 WebShell 检测任务中的有效性, 本文在 PHP 数据集上进行了详尽的消融实验。

在消融实验中, 本文比较了使用传统词向量方法(TF-IDF、HashVect、One-Hot)和动态词向量模型(Word2Vec、Glove 和 Bert)对 WebShell 代码的词向量构建效果。进一步地, 本文结合深度学习算法 CNN 分析了不同词向量构建方式对 WebShell 检测性能的影响。随后, 引入了 GCN 模块和联合训练方法, 以探究这些性能增益模块对改善分类性能的有效性。对比结果如表 4 所示, 结果表明, 引入性能增益模块的模型在各项分类性能指标上都显示出了明显的提升。

表 4 消融实验结果

Table 4 Ablation experimental results

分类模型	准确率(%)	精确率(%)	召回率(%)	F_1 分数(%)
TF-IDF+CNN	94.79	95.68	93.66	94.66
HashVect+CNN	91.82	91.07	92.45	91.75
One-Hot+CNN	94.64	94.29	94.86	94.58
Word2Vec+CNN	96.54	96.86	96.20	96.53
Glove+CNN	97.21	94.69	95.13	91.73
Bert+CNN	98.06	98.22	98.57	98.39
Bert+GCN	98.97	98.48	99.45	98.96
JLBertGCN	99.25	98.77	99.06	99.38

由表 4 可知:

(1) 动态词向量构建模型 Bert 在比较其他词向量构建方法时表现出显著优势。具体来说, 与 Word2Vec 和 GloVe 相比, Bert 的准确率分别提高了 1.52%和 0.85%。此外, 使用 Bert 模型得到的精确率、召回率和 F_1 分数分别为 98.22%、98.57%和 98.38%, 这些指标在所有比较的模型中均为最高。这种性能提升主要归因于 Bert 能够综合考虑单词左右两侧的上下文信息, 生成更全面和精确的词向量。Bert 模型还采用了掩码语言模型(Masked Language Model, MLM)和下一句预测(Next Sentence Prediction, NSP)这两种预训练目标, 有效地捕捉了词汇的多义性和歧义性, 以及句子间的复杂关系, 从而深入理解文本语境。这些特性使得 Bert 在词向量构建上更为准确, 为最终的分类判断提供了更可靠的依据。

(2) 引入 GCN 模块后模型准确率和 F_1 分数达到 98.97%和 98.96%, 相比 CNN 模型分类分别提升了 0.91%和 0.57%。因为该模块能捕捉到文本数据当中不规则结构的语义信息, 从而捕捉 WebShell 代码数据集中的不规则结构形成的语义关联, 使提取的特征更加丰富, 提高模型的分类效果。

(3) 最后使用联合训练方法后准确率和 F_1 分数达到 99.25%和 99.38%, 这表明联合训练利用不同模型或任务之间的共享信息和特异信息, 利用 Bert 的大规模预训练能力和 BertGCN 的直推学习能力, 从而提高模型的检测性能。

4.5 对比实验

为了验证 JLBertGCN 相较于当前主流的 WebShell 检测方法在性能上的优势, 本节对比了其他典型的 WebShell 检测算法:

(1) 文献[15]提出了一种基于语义分析和改进 TextCNN 的 WebShell 检测方法, 该方法先使用语义分析技术对 WebShell 代码进行预处理, 然后利用改进的 TextCNN 模型进行特征提取和分类。

(2) 文献[16]提出了一种基于双向 GRU 和注意力机制的 WebShell 检测方法, 该方法使用双向 GRU 网络对 WebShell 代码进行编码, 并结合多头注意力机制对编码结果进行融合, 最后通过全连接层进行分类。

(3) 文献[12, 22-23]利用不同向量构建方式结合 CNN 模型进行 WebShell 检测任务, 并取得了良好的检测效果。

(4) 文献[24]提出了使用单层和多层长短期记忆网络(Long Short-Term Memory Network, LSTM)对 WebShell 脚本进行编码的方法, 随后使用全连接层进行 WebShell 检测。

表 5 展示了 JLBertGCN 模型在整个数据集上与其他模型的 WebShell 检测性能对比。从表 5 中可知 JLBertGCN 模型在准确率、精确率、召回率和 F_1 分数 4 个主要性能指标上均表现出较高的效果, 其准确率达到了 99.26%, 精确率达到了 99.21%, 召回率为 98.89%, 而 F_1 分数则达到了 99.05%。相较于表 5 中其他几种常用的检测模型, JLBertGCN 显示了更为稳定和优越的性能表现。

表 5 整体数据集上与其他模型分类结果对比

Table 5 Comparison of classification results with other models on the entire dataset

方法	准确率(%)	精确率(%)	召回率(%)	F_1 分数(%)
文献[15]	98.89	97.94	98.15	98.04
文献[16]	98.77	98.99	97.31	98.14
文献[22]	96.54	96.48	97.45	96.96
文献[24]	96.55	96.61	96.22	96.41
本文方法	**99.26**	**99.21**	**98.89**	**99.05**

JLBertGCN 在 PHP 文本数据集上与其他典型方法对比时, 其在准确率、精确率、召回率和 F_1 分数 4 个指标上的表现均为最优, 其中准确率达到了

99.25%, 精确率达到了 98.77%, 召回率达到了 99.06%, F_1 分数达到了 99.02%。表 6 展示了 JLBertGCN 与其他 4 种方法的详细对比结果。

表 6 PHP 数据集上与其他模型分类结果对比
Table 6 Comparison of classification results with other models on PHP dataset

方法	准确率(%)	精确率(%)	召回率(%)	F_1 分数(%)
文献[15]	99.12	98.19	97.40	97.79
文献[16]	98.66	98.79	98.49	98.64
文献[22]	96.54	96.86	96.20	96.53
文献[24]	97.11	96.83	97.41	97.12
本文方法	**99.25**	**98.77**	**99.06**	**99.38**

表 7 给出了 JLBertGCN 与其他模型在 ASP 数据集上分类性能的对比结果。从表 7 可以看出, JLBertGCN 模型在准确率上最高, 达到了 99.32%, 比改进的 TextCNN 模型高了 1.11%, 比其他 3 种方法高了 0.41%以上。在精确率上, 本文方法稍低于双向 GRU 模型, 为 98.59%, 但仍然比改进的 TextCNN 模型高了 1.37%, 比 CNN 模型和 LSTM 模型高了 4.16%以上。在召回率上, JLBertGCN 模型为 98.00%, 比改进后的 TextCNN 模型略低, 比其他模型高了 2.27%以上。在 F_1 分数上, JLBertGCN 模型为 98.29%, 比改进后的 TextCNN 模型高了 0.16%, 比双向 GRU 模型高了 0.83%, 比 LSTM 模型高了 4.82%, 比 CNN 模型高了 1.96%。

表 7 ASP 数据集上与其他模型分类结果对比
Table 7 Comparison of classification results with other models on ASP dataset

方法	准确率(%)	精确率(%)	召回率(%)	F_1 分数(%)
文献[15]	98.21	97.22	99.06	98.13
文献[16]	98.91	99.25	95.73	97.46
文献[22]	95.58	93.75	99.06	96.33
文献[24]	94.62	93.48	92.63	93.47
本文方法	**99.32**	**98.59**	**98.00**	**98.29**

表 8 展示了本文方法与其他模型在 JSP 数据集上的对比结果。从表中可以看出, JLBertGCN 模型在准确率上最高, 达到了 99.23%, 比改进的 TextCNN 模型高了 0.27%, 比其他 3 种方法高了 0.32%以上。在精确率上, 本文方法稍低于双向 GRU 模型, 为 98.70%, 但仍然比改进的 TextCNN 模型高了 0.64%。在召回率上, JLBertGCN 模型为 99.13%, 比改进后的 TextCNN 模型略低, 比其他模型高了 2.37%以上。在 F_1 分数上, JLBertGCN 模型为 98.91%, 稍低于改进后的 TextCNN 模型, 比双向 GRU 模型高了 1.55%, 比 LSTM 模型高了 0.86%, 比 CNN 模型高了 0.15%。

表 8 JSP 数据集上与其他模型分类结果对比
Table 8 Comparison of classification results with other models on JSP dataset

方法	准确率(%)	精确率(%)	召回率(%)	F_1 分数(%)
文献[15]	98.96	98.04	99.24	99.01
文献[16]	98.91	99.25	95.73	97.46
文献[22]	97.57	98.35	99.17	98.76
文献[24]	97.09	99.38	96.76	98.05
本文方法	**99.23**	**98.70**	**99.13**	**98.91**

JLBertGCN 在所有数据集中均实现了最优性能, 实验结果充分验证了 JLBertGCN 方法的有效性。在词向量构建方面, Bert 相较于传统的文本处理方法, 能够更有效地处理 WebShell 脚本中的多义性和上下文关系, 从而生成更全面的词向量, 增强了模型的分类能力。此外, JLBertGCN 采用的双通道联合训练策略有效结合了 Bert 和 BertGCN 两部分的特征学习能力, 显著提高了模型的整体学习效果和性能表现。

5 结论

本文提出的 JLBertGCN 模型有效结合了动态词向量和图卷积技术, 显著提升了 WebShell 检测性能。通过 Bert 模型构建的动态词向量加强了对 WebShell 代码中多义词的区分能力及上下文理解, 优化了文本数据的语义信息和特征提取。此外, 模型利用 GCN 将代码片段转化为图结构, 节点由 Bert 预训练的向量初始化, 边表示代码依赖关系, 进一步增强了结构特征的学习。采用双通道联合训练策略, 模型融合了 Bert 和 BertGCN 的输出, 充分利用了 Bert 的预训练能力与 GCN 的图结构学习优势, 形成了精确的文本表示。实验结果表明, 该模型在整体数据集上的准确率和 F_1 分数分别达到 99.26%和 99.05%, 明显优于现有方法。JLBertGCN 在处理复杂的代码语义和结构特征方面的表现显著, 能有效应对高度混淆和动态变化的 WebShell 攻击场景。

参考文献

[1] Zhu Z N, Jin J Q. Research and practice of WebShell emergency response detection method[J]. *Internet of Things Technologies*, 2023, 13(8): 108-110, 114.
(朱振南, 金京犬. WebShell 应急响应检测方法研究与实践[J]. *物联网技术*, 2023, 13(8): 108-110, 114.)

[2] Sun H X, Cao L, Wu D F, et al. Webshell detection overview[J]. *Information Security and Communications Privacy*, 2022, 20(9):

82-90.
(孙昊翔, 曹浪, 吴迪锋, 等. Webshell 检测综述[J]. *信息安全与通信保密*, 2022, 20(9): 82-90.)

[3] Yu T D, Zou F T, Li L S, et al. An Encrypted Malicious Traffic Detection System Based on Neural Network[C]. *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2020: 62-70.

[4] Le H V, Du H P, Nguyen H N, et al. A proactive method of the Webshell detection and prevention based on deep traffic analysis[J]. *International Journal of Web and Grid Services*, 2022, 18(4): 361-383.

[5] Sasikala D, Chandrakanth D, Sai Pranathi Reddy C, et al. Inhibiting Webshell attacks by random forest ensembles with XGBoost[J]. *Journal of Information Technology and Digital World*, 2022, 4(3): 153-166.

[6] Li Y, Huang J, Ikusan A, et al. *ShellBreaker*: Automatically detecting PHP-based malicious Web Shells[J]. *Computers & Security*, 2019, 87: 101595.

[7] Le H V, Nguyen T N, Nguyen H N, et al. An efficient hybrid webshell detection method for Webserver of marine transportation systems[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2021, 24(2): 2630-2642.

[8] Kang B, Jung J, Kim H, et al. Efficient Webshell Detection Using Statistical Features and Opcode Analysis in PHP Source Files[J]. IEEE Transactions on Information Forensics and Security, 2022, 17: 1234-1247.

[9] Kang W Z, Zhong S P, Chen K Z, et al. RF-AdaCost: WebShell Detection Method that Combines Statistical Features and Opcode[C]. *Frontiers in Cyber Security*, 2020: 667-682.

[10] Yong B B, Wei W, Li K C, et al. Ensemble machine learning approaches for Webshell detection in internet of things environments[J]. *Transactions on Emerging Telecommunications Technologies*, 2022, 33(6): e4085.

[11] Fang Y, Qiu Y Y, Liu L, et al. Detecting Webshell Based on Random Forest with FastText[C]. *The 2018 International Conference on Computing and Artificial Intelligence*, 2018: 52-56.

[12] Liu X, Zhang Y L, Yu Q C, et al. SmartEagleEye: A cloud-oriented Webshell detection system based on dynamic gray-box and deep learning[J]. *Tsinghua Science and Technology*, 2024, 29(3): 766-783.

[13] Lv Z H, Yan H B, Mei R. Automatic and Accurate Detection of Webshell Based on Convolutional Neural Network[C]. *Cyber Security*, 2019: 73-85.

[14] Church K W. Word2Vec[J]. *Natural Language Engineering*, 2017, 23(1): 155-162.

[15] Zhou L, Wang C, Shi Y. Research of Webshell detection based on RNN[J]. *Computer Engineering and Applications*, 2020, 56(14): 88-92.
(周龙, 王晨, 史峄. 基于 RNN 的 Webshell 检测研究[J]. *计算机工程与应用*, 2020, 56(14): 88-92.)

[16] Zhang J H. Research on Webshell Detection Algorithm Based on Deep Learning[D]. Nanchang: Nanchang University, 2022.
(章嘉昊. 基于深度学习的 Webshell 检测算法的研究[D]. 南昌: 南昌大学, 2022.)

[17] Liu Z Q, Li D F, Wei L L. A new method for WebShell detection based on bidirectional GRU and attention mechanism[J]. *Security and Communication Networks*, 2022, 2022(1): 3434920.

[18] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. *Advances in Neural Information Processing Systems*, 2017: 30.

[19] Yao L, Mao C S, Luo Y. Graph Convolutional Networks for Text Classification[C]. *The Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019: 7370-7377.

[20] Hwang J W, Park J, Park R H, et al. Audio-visual speech recognition based on joint training with audio-visual speech enhancement for robust speech recognition[J]. *Applied Acoustics*, 2023, 211: 109478.

[21] Li Q M, Han Z C, Wu X M. Deeper insights into graph convolutional networks for semi-supervised learning[J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, 32(1): 3538-3545.

[22] Dong Y N, Liu Q W, Du B, et al. Weighted Feature fusion of convolutional neural network and graph attention network for hyperspectral image classification[J]. *IEEE Transactions on Image Processing*, 2022, 31: 1559-1572.

[23] Cheng K F, Wang H D, Hu G J, et al. Research on Webshell Detection Based on Semantic Analysis and Text-CNN[C]. *2021 17th International Conference on Computational Intelligence and Security*, 2022: 529-534.

[24] Wu Y L, Song M L, Li Y K, et al. Improving Convolutional Neural Network-Based Webshell Detection through Reinforcement Learning[C]. *Information and Communications Security*, 2021: 368-383.

[25] Zhou Z H, Li L, Zhao X. Webshell Detection Technology Based on Deep Learning[C]. *2021 7th IEEE Intl Conference on Big Data Security on Cloud, IEEE Intl Conference on High Performance and Smart Computing, and IEEE Intl Conference on Intelligent Data and Security*, 2021: 52-56.

翟江涛 于 2013 年在南京理工大学自动化学院获得博士学位。现任南京信息工程大学副教授, CCF 会员。研究领域为智能网络流量解析、装备试验鉴定、装备维修与保障。Email: jiangtaozhai@nuist.edu.cn

王涛 于 2022 年在南京信息工程大学计算机与软件学院获得学士学位。现在南京信息工程大学电子信息专业攻读硕士学位。研究领域为智能网络流量解析。Email: 987786148@qq.com

周桥 于 2021 年在安徽新华学院电子与通信工程学院获得学士学位。2024 在南京信息工程大学电子信息专业获得硕士学位。研究领域为网络流量解析。Email: 934722451@qq.com

董燚 于 2022 年在盐城师范学院物理与电子工程学院获得学士学位。现在南京信息工程大学电子信息专业攻读硕士学位。研究领域为智能网络流量解析。Email: dy_5830@163.com

王子豪 于 2022 年在南京邮电大学通达学院通信工程专业获得学士学位。现在南京信息工程大学电子信息专业攻读硕士学位。研究领域为智能网络流量解析。Email: 4everpok1@gmail.com