引用格式:吴波,沈璇,耿君峰,等. 基于复合式神经网络的二进制软件漏洞检测方法[J]. 信息对抗技术,2022, 1(3):57-65. [WU Bo, SHEN Xuan, GENG Junfeng, et al. Binary software vulnerability detection method based on composite neural network[J]. Information Countermeasure Technology, 2022, 1(3):57-65. (in Chinese)]

基于复合式神经网络的二进制软件漏洞检测方法

吴 波,沈 璇*,耿君峰,刘 波

(国防科技大学信息通信学院,湖北武汉 430030)

摘 要 在软件漏洞检测领域,传统神经网络模型和图神经网络模型是已被验证的有效方法。目前,方案大多针对源代码进行漏洞检测,运用神经网络模型对二进制软件进行漏洞检测的研究相对较少,更是缺乏对图神经网络在二进制软件漏洞检测方面的研究。为充分研究神经网络模型在二进制软件漏洞检测方面的有效性,提出了一种基于复合式神经网络的二进制软件漏洞检测方法。将二进制代码向量化表示为同时支持传统神经网络模型和图神经网络模型训练的图数据结构;使用传统神经网络模型和图神经网络模型相结合的复合式神经网络模型对图数据结构进行学习和验证;在公开的二进制软件漏洞数据集上进行实验和对比分析,结果表明该方法能够有效提升漏洞检测能力,在准确率、精确度等性能指标方面都有明显提升。

关键词 漏洞检测;神经网络;图神经网络;二进制软件

中图分类号 TP 309 **文献标志码** A **文章编号** 2097-163X(2022)03-0057-09

DOI 10.12399/j.issn.2097-163x.2022.03.005

Binary software vulnerability detection method based on composite neural network

Wu Bo, Shen Xuan*, Geng Junfeng, Liu Bo

(College of Information and Communication, National University of Defense Technology, Wuhan 430030, China)

Abstract Both traditional neural network models and graph neural network models have been validated as effective methods in the field of software vulnerability detection, but most of the current solutions are aimed at source code for vulnerability detection, and relatively little research has been conducted on applying neural network models directly to binary software for vulnerability detection, especially graph neural networks for binary software vulnerability detection is lacking. In order to fully investigate the effectiveness of neural network models in binary software vulnerability detection, this paper proposed a composite neural network based binary software vulnerability detection method. Firstly, the binary code was vectorised into a graph data structure that supports both traditional neural network models and graph neural network models for training. Then, a composite neural network model that combines tradi-

收稿日期:2022-07-28 **修回日期**:2022-10-08

通信作者:沈璇,E-mail:shenxuan_08@163.com

作者简介:吴波(1985—),男,博士,讲师,研究方向为软件漏洞挖掘与分析; 沈璇(1990—),男,博士,副教授,研究方向为网络与信息安全; 耿君峰(1989—),男,讲师,研究方向为网络空间安全; 刘波(1995—),男,助理研究员,研究方向为智能运维

基金项目:陕西省自然科学基金资助项目(2019JQ-716)

tional neural network models and graph neural network models was used to learn and validate the graph data structure. Finally, experiments and comparative analysis were conducted on a publicly available binary software vulnerability dataset. The experimental results show that the method could effectively improve vulnerability detection capabilities, with significant improvements in performance metrics such as accuracy and precision.

Keywords vulnerability detection; neural network; graph neural network; binary software

0 引言

近年来，各行各业网络安全事件层出不穷，软件系统的安全性问题备受关注，软件漏洞检测技术成为研究的重点，特别是结合新理论、新方法来研究如何挖掘二进制软件漏洞成为一个亟待深入研究和解决的课题，对于确保信息系统安全有着重要意义。

在针对源代码软件的漏洞检测方面，机器学习方法的应用已经有较为充分的研究。文献[1]设计了一种基于机器学习算法的开源软件漏洞检测方法，利用代码可量化尺度以及提取代码库中的元数据来辅助代码审计人员更快地发现代码中的缺陷，主要通过学习 CVE 和代码提交日志的关联关系来发现漏洞，是一种半自动化方法。文献[2]设计了一种深度神经网络用于发现源代码中的漏洞，以代码中的 token 序列为特征进行学习训练，并通过特征选择算法降低搜索空间，最终发现源码中的漏洞。文献[3]通过在指针类型分析过程中引入机器学习方法，能够对比较难于发现的 UAF(use after free)漏洞类型进行有效检测，且误报率更低。文献[4]设计了一种针对源代码的表示方法，以及将源代码映射到向量空间的方法，并在此基础上研究了无监督学习下的 3 种典型机器学习算法，即降维、异常检测、聚类分析。文献[5]设计实现了一种能够应用于源代码的深度学习漏洞检测方法，通过源代码片段来表示被分析程序，然后使用神经网络对源码漏洞进行学习和检测。文献[6]设计了一个通用的深度学习框架用于挖掘软件中的漏洞，考虑综合利用语义、语法信息和适当的向量表示方法来增强学习和检测效果。文献[7]提出了一种基于深度学习的多分类漏洞检测系统，通过代码注意力机制准确捕获漏洞类型信息。

在针对二进制软件的漏洞检测方面，相关研究较少。二进制软件漏洞检测研究的一个重要制约因素是缺乏标记好有/无漏洞的数据集。尽管已经有一些源代码形式的数据集[8]，但一般都无法直接转化为二进制数据集，而且相关数据集构造的标准缺乏统一认识。文献[9]开发了一套源代码自动修复及编译工具，基于 NDSS 18 的源代码数据集构建了包含 32 281 个样本的二进制数据集，为后续针对二进制软件的机器学习研究提供了良好的基础。文献[10]设计了一种跨平台的面向二进制程序的漏洞检测方法，利用语义学习训练得到的模型对目标二进制函数进行预测，自动筛选出前 N 个可能存在漏洞的代码位置，并进一步使用模拟执行引擎进行验证，最终辅助安全分析人员粗粒度定位二进制程序中的漏洞函数。文献[11]提出了一种基于神经网络的函数签名识别方法，可为二进制软件漏洞检测提供语义支撑。

综上所述，传统神经网络模型在面向源代码和二进制软件的漏洞检测任务中均能表现出良好性能，图神经网络模型在源代码软件漏洞检测方面已得到充分应用，但是利用其进行二进制软件漏洞检测的研究相对较少，针对同时结合这 2 种神经网络模型进行漏洞检测的研究更是缺乏。为此，本文研究并提出了一种复合式神经网络模型，通过结合图神经网络模型和传统神经网络模型的优势，能够实现更好的漏洞检测效果。

1 相关工作

在基于深度学习的源代码漏洞检测方面，VulDeePecker[5]是将传统深度学习方法用于漏洞检测的典型代表，该研究提出使用源代码片段的方式来表示程序，通过代码片段来捕获高层的语义信息，其代码片段的抽取主要是基于关键点思想来进行，首先，设置库函数调用、数组使用等关键点；然后，基于关键点进行程序切片，并对切片后的程序代码片段进行标记和向量化表示；最后，将其用于训练 BLSTM 网络得到一个基于深

度学习的漏洞检测模型。基于深度学习的漏洞检测工具，VulDeePecker 给传统的软件漏洞检测提供了新思路，也为后续研究提供了软件漏洞数据集参考。

在图神经网络应用于源代码漏洞检测方面，文献[12]设计了一种基于图神经网络的源代码漏洞检测模型，首先采用 LLVM IR 中间语言来抽取目标程序的控制流图信息，其次使用词向量嵌入方法编码图数据结构的节点，最后使用一个多层残差连接的图神经网络模型对构造的图数据进行训练和漏洞检测。文献[13]通过考虑图节点中的不同边类型，使用 GGNN 模型对多关系图进行学习，最终检测函数级别源代码漏洞，有效提高了漏洞检出效果，并且在不同编程语言间进行漏洞检测的迁移学习方面做出了有益的尝试。

在基于深度学习的二进制软件漏洞检测方面，文献[9]在 VulDeePecker 数据集的基础上设计了一种最大距离自动编码模型，利用深度学习的方法自动学习二进制代码中的潜在特征，并用于实现漏洞检测。该研究结合传统的 RNN 神经网络和变分自动编码模型，能够从有漏洞和没有漏洞的二进制代码中学习出一种距离最大的表示方法，从而更好地支持漏洞检测分类。文献[9]的另一项重要工作是基于 VulDeePecker 源代码数据集构造了带标注的二进制软件漏洞检测数据集，为后续针对二进制软件的机器学习研究提供了数据集参考。

2　方案设计

随着人工智能技术的发展，越来越多基于机器学习的漏洞检测研究得以开展。本文从优势互补集成的角度，针对二进制软件漏洞检测问题，考虑传统神经网络模型和图神经网络模型的协作提升问题，提出了一种基于复合式神经网络的二进制软件漏洞检测方法。用扩展的图数据结构来表示二进制程序，并构建结合传统神经网络和图神经网络的漏洞检测模型，进而自动学习和挖掘二进制软件中的潜在漏洞。

2.1　总体框架设计

为得到符合复合式神经网络学习模型输入的向量，首先，对二进制代码进行反汇编，将每一行反汇编结果作为图的一个节点，然后对每行反汇编结果进行固定长度编码作为图节点的特征表述；其次，将反汇编生成的控制流图、定义使用链等信息编码为图节点之间的连接关系，并最终以图的方式表达二进制软件，从而同时支持在模型训练过程中使用的传统神经网络模型和图神经网络模型；最后，设计并实现了一个基于复合式神经网络的二进制软件漏洞检测原型系统，方法总体框架设计如图 1 所示。

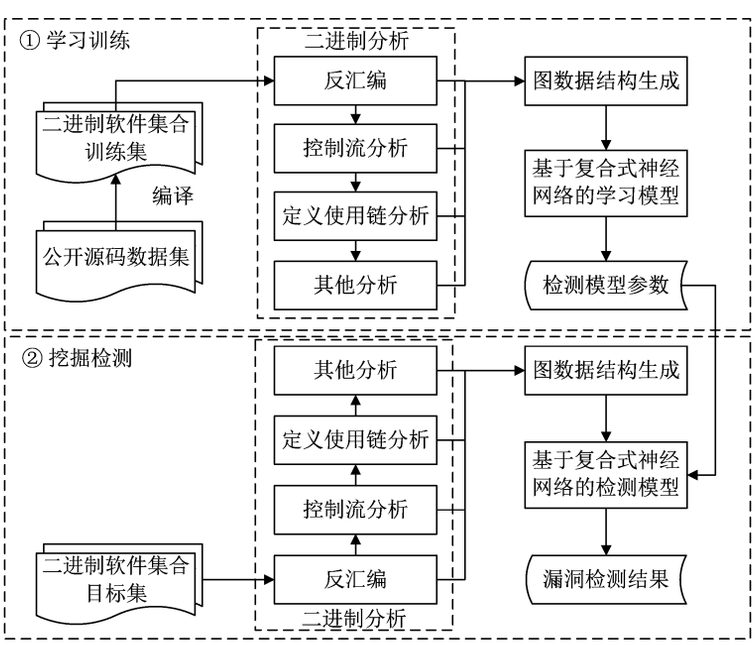


图 1　基于复合式神经网络的二进制软件漏洞检测方法框架图

Fig. 1　Overview of binary software vulnerability mining method based on composite neural network

与其他基于机器学习的方法一样，本文方法设计分为 2 个阶段：即学习训练和挖掘检测。在学习训练阶段，系统从已知漏洞的二进制软件训练集合中提取目标软件的图数据结构表示，然后基于软件的图数据结构表示训练复合式神经网络模型，得到能够自动检测二进制软件漏洞的检测模型参数；在挖掘检测阶段，首先对真实的二进制软件实施反汇编和控制流分析等操作，然后生成与训练一致的图数据结构，并通过复合式神经网络检测模型进行预测，输出最终的漏洞检测结果。

2.2 二进制分析设计

对于给定的一个二进制程序，如何将其表示为适合模型的输入，同时又尽量多地保留二进制软件中的语义信息是首先要考虑的问题。为此，本文提出了一种基于复合式神经网络的方法。首先，通过反汇编引擎对目标程序执行反汇编操作，并恢复程序的控制流图，与传统的以基本块为单位构造控制流图不同，本文将每一条反汇编得到的指令都作为控制流图的节点；然后，在恢复控制流图的基础上，继续对每一条指令中涉及变量的定义使用链进行分析，并在控制流图的基础上为定义使用链相关的节点添加一条新类型的边。

理论上，对反汇编结果可以同时执行控制流分析、数据流分析、定义使用链分析、到达定值分析等多种典型的程序分析方法，然后将分析结果用于补充完善图数据结构中的连接边信息，从而为二进制软件的图数据表示提供更好的表达能力。

2.3 图数据结构生成设计

针对二进制分析的结果，所有的节点都是汇编指令，节点之间的连接是不同类型的边。与文献[9]中数据预处理方式类似，本文对每一个节点进行编码的方式为：将每个汇编指令节点对应的二进制数据看作是十六进制数据序列，即 $I=[d_0,d_1,\cdots,d_n]$，其中，d_i 为每字节数据，n 为 1 条指令的数据长度；然后对数据序列的数据按字节进行频度排序，即 $I_f=[(v_0,f_0),(v_1,f_1),\cdots,(v_m,f_m)]$，其中，$v_i$ 为数据集中出现的数据值，f_i 为对应数值出现的频率，$m=255$；再将排序结果映射到长度为 256 的数据向量 $n_i=[e_0,e_1,\cdots,e_{255}]$，其中，$e_i=f_i$。本文对每一条边进行编码的方式为：使用源节点编号、目的节点编号和边类型构成的三元组来表示一条边，即 $E_i=(u_{\mathrm{src}},u_{\mathrm{dst}},t)$。对所有节点和边编码完毕后，整个二进制软件的图数据结构表示如式(1)所示：

$$\begin{cases} \boldsymbol{G}=\{\boldsymbol{n}_{\mathrm{data}},\boldsymbol{e}_{\mathrm{data}}\} \\ \boldsymbol{n}_{\mathrm{data}}=[n_0,n_1,\cdots,n_p]^{\mathrm{T}} \\ \boldsymbol{e}_{\mathrm{data}}=[e_0,e_1,\cdots,e_q]^{\mathrm{T}} \end{cases} \tag{1}$$

式中，$\boldsymbol{G}$ 为二进制软件图数据结构，$\boldsymbol{n}_{\mathrm{data}}$ 为所有节点的信息，$\boldsymbol{e}_{\mathrm{data}}$ 为所有边的信息，p 为所有数据集中最长的节点数量，q 为每个图数据结构中边的数量。

2.4 复合式神经网络模型设计

由于漏洞代码与其他代码在某个特性上存在不同，例如，缓冲区溢出漏洞可能是执行路径上缺少了对长度校验函数的调用，类似自然语言中缺乏限定词导致语句感情色彩发生变化，而神经网络具备分辨这种不同的能力。因此，很多研究结合神经网络来学习不确定数据或者伪噪声数据之间的联系。传统基于源代码的漏洞检测方法，其基本思想都是首先将源代码通过词向量算法进行矢量化，然后基于自然语言处理领域里成熟的神经网络模型，例如 LSTM、Bi-LSTM、TextCNN 等进行学习和训练。由于基于源代码分析保留的语义信息相对丰富，因此漏洞检测效果较好。但是，在二进制软件漏洞检测领域，由于缺乏源代码层丰富的语义信息，直接从二进制或反汇编代码着手，然后应用传统的自然语言处理算法模型很难获得良好的挖掘效果。

图神经网络与传统神经网络的最大不同点在于：传统神经网络处理的是欧式数据，即规则且排列整齐的数据类型，如图像(网格数据)、文本(序列数据)等，而图神经网络处理的是非欧式数据，即排列相对不整齐的数据类型，对于数据中的某个节点，其邻居节点较难被找到且数量不固定。对于二进制软件，从程序分析的角度看，通常可以表述为控制流图、数据流图等不同类型的图结构，而这正好是图神经网络模型能够处理的数据类型。因此，本文考虑结合传统神经网络模型和图神经网络模型进行二进制软件的漏洞检测训练和预测。

设计的复合式神经网络模型结构如图 2 所示，从图 2 可以看出，有 2 条相对独立的路径对输入数据进行处理。其中一条路径首先利用图数据结构中节点的特征向量进行传统神经网络模型处理，本文选用的是双向循环神经网络模型，

然后经过全连接神经网络层输出传统神经网络模型处理结果 O_1；另一条路径首先也是对输入的节点数据进行特征向量维度变换映射，然后将图数据结构节点数据和边数据通过基于注意力机制的图神经网络模型进行处理，再将处理结果经过基于注意力机制的全局池化层进行读数，最后由全连接神经网络层输出图神经网络模型处理结果 O_2。然后 2 条数据处理路径的输出结果 O_1、O_2 堆叠后送入基于注意力机制的分类层得到最终的模型输出。

模型具体的内部实现细节参见代码库：https://gitee.com/cherry_wb/compnet。

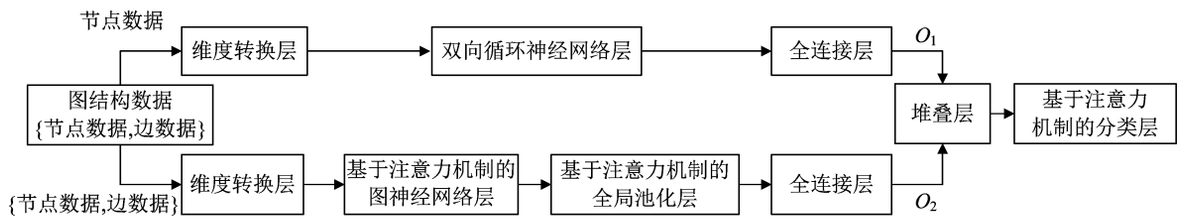


图 2　复合式神经网络模型结构

Fig. 2　Illustration of the composite neural network model

3　实验与评估

为了验证本文方法的有效性，对提出的数据处理算法和神经网络模型进行编程，并实现了一个简单的原型系统，包括二进制分析、图数据结构生成和模型训练与检测 3 个功能模块。

在二进制分析模块中，基于 Ghidra 平台[14]实现二进制软件反汇编，并在此基础上进行控制流分析和定义使用链分析，从而生成用于构造图数据结构的基础数据。

在图数据结构生成模块中，该模块作为原型系统的辅助功能来实现，直接读取分析完的二进制数据，并对所有的节点和边进行向量化编码，然后利用 DGL[15]图神经网络框架提供的编程接口构造图数据样本。

在模型训练与检测模块中，结合传统双向循环神经网络模型和图神经网络模型，基于 DGL[15]图神经网络框架设计和实现了复合式神经网络模型，并在选用的数据集上进行学习训练和漏洞检测。

为了评估本文所提方法的有效性，使用基准测试数据集对漏洞检测能力进行实验与分析。重点采用的评估指标包括准确率（A_{Acc}）、精确率（P_{Pre}）、召回率（R_{Rec}）以及 F_1。准确率是对于给定的测试数据集，系统正确判别为有漏洞数和无漏洞数与总样本数之比。精确率是系统正确检测为有漏洞的程序数与所有被系统检测为有漏洞的程序数之比，反映漏洞检测误报情况；召回率是正确检测为有漏洞的程序数与全部有漏洞的程序数之比，反映系统的完备性；F_1 是精确率和召回率的加权调和平均值，反映系统整体的检测效果，一般 F_1 越高，系统的整体性能越好。实验过程中主要关注以下 2 个问题：

（1）基于图神经网络是否能够实现二进制软件漏洞检测。

（2）复合式神经网络模型是否提高了软件漏洞检测的准确率、精确率、召回率以及 F_1 等度量指标。

本文实验环境为：Ubuntu 20.04 64 位操作系统，Intel Core i7-10875H CPU 处理器，32 GB 内存，RTX3070Ti 显卡，8 G 显存，数据集为 NDSS18 二进制数据集[8]和 Devign 二进制数据集[16-17]。数据集按 8∶1∶1 的比例划分为训练集、测试集和验证集，对传统神经网络模型、图神经网络模型和复合式神经网络模型进行训练，模型参数设置见表 1 所列，实验验证重点对不同模型的 A_{Acc}、P_{Pre}、R_{Rec}、F_1 等指标进行对比分析，最终实验结果分析如下。

表 1　模型参数设置

Tab. 1　Model parameter setting

参数	数值
输入层维度	256
隐藏层维度	256
RNN 层数	2
GNN 层数	3
学习率	10^3
迭代次数	100

3.1　模型复杂度比较分析

直观上看，复合式神经网络模型计算复杂度

和模型参数数量要大于传统神经网络模型。为了定量化比较不同模型的复杂度，基于 thop 模块对模型的计算复杂度(FLOPS)和参数数量进行了对比分析，结果见表 2 所列。从表 2 可以看出，复合式神经网络模型的计算复杂度和模型参数数量都比较大，其表达能力也更强。

表 2 复合式神经网络模型与传统神经网络模型复杂度比较

Tab. 2 The complexity comparison of composite neural network model and traditional neural network model

方法	FLOPS/G	参数数量/M
复合式神经网络模型	176.740	3.590
传统神经网络模型	41.757	1.905

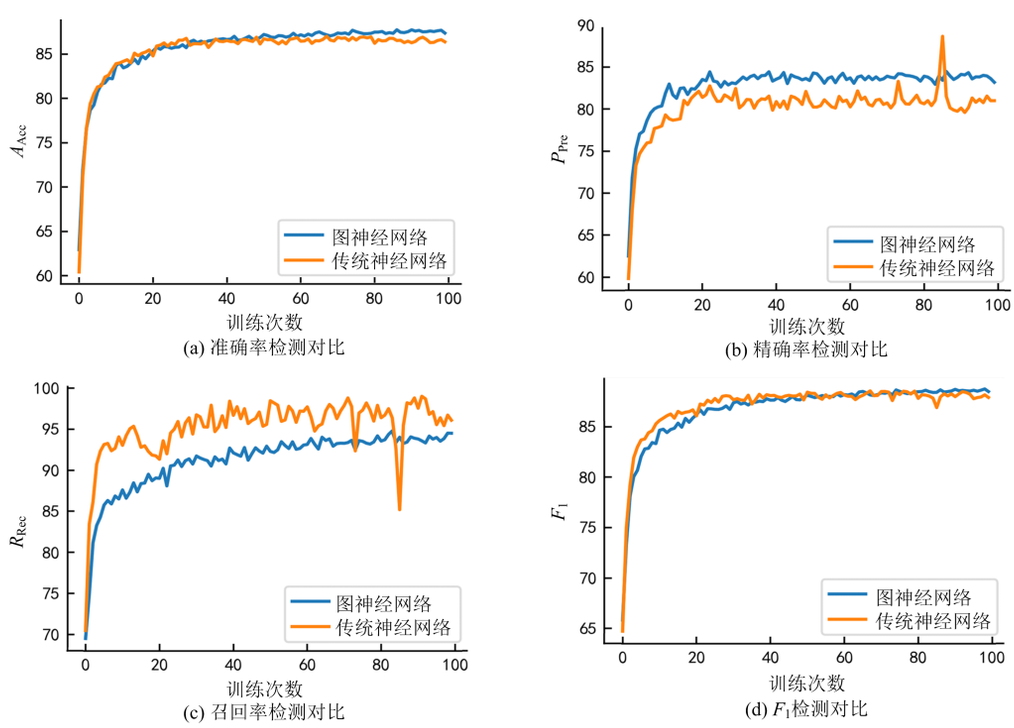
3.2 图神经网络模型性能分析

表 3 和图 3 为传统神经网络模型和图神经网络模型在 NDSS18 二进制数据集上的训练效果，可以看出，使用图神经网络模型能够在测试数据集上达到与传统神经网络模型整体性能相当的检测效果。因此，问题 1 成立，即图神经网络模型应用于二进制软件漏洞检测可以取得较好的效果。

表 3 图神经网络模型与传统神经网络模型检测效果

Tab. 3 The detection effect of graph neural network model and traditional neural network model

方法	A_{Acc} (%)	P_{Pre} (%)	R_{Rec} (%)	F_1 (%)
传统神经网络模型	87.00	87.67	99.00	88.55
图神经网络模型	87.73	84.48	94.78	88.74



(a) 准确率检测对比 (b) 精确率检测对比

(c) 召回率检测对比 (d) F_1检测对比

图 3 图神经网络模型与传统神经网络模型检测效果对比图

Fig. 3 The detection effect comparison of graph neural network model and traditional neural network model

3.3 复合式神经网络模型性能分析

表 4 和图 4 为传统神经网络模型和复合式神经网络模型在 NDSS18 二进制数据集上的训练效果，可以看出，相比传统神经网络模型，使用复合式神经网络模型在准确率和精确率上能够达到更好的效果，但 F_1 指标没有明显提升，召回率略低于传统神经网络模型，因此，问题 2 基本成立，即复合式神经网络模型应用于二进制软件漏洞检测可以取得更好的效果。在 NDSS18 数据集上，本文提出的复合式神经网络模型在精确率、准确率指标上均有所提升，因此相较于传统模型效果更优。

表 4 复合式神经网络与传统神经网络检测效果(NDSS18 数据集)

Tab. 4 The detection effect of composite neural network and traditional neural network(NDSS18 DataSet)

方法	A_{Acc} (%)	P_{Pre} (%)	R_{Rec} (%)	F_1 (%)
传统神经网络模型	87.00	87.67	99.00	88.55
复合式神经网络模型	88.04	87.86	91.53	88.60

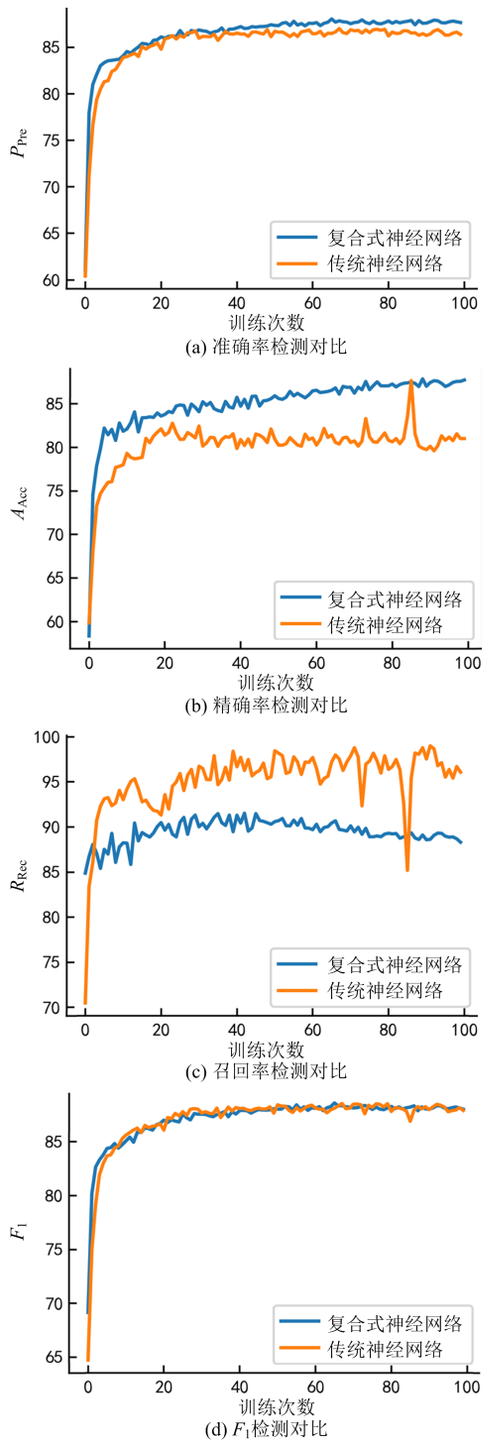


图 4　复合式神经网络模型与传统神经网络模型检测效果对比图(NDSS18 数据集)

Fig. 4　The detection effect of composite neural network and traditional neural network(NDSS18 DataSet)

图 5 和表 5 为传统神经网络模型和复合式神经网络模型在 Devign 二进制数据集上的训练效果。Devign 二进制数据集是基于真实软件构造的，仅公开了 QEMU 和 FFmpeg 的部分数据，从原作者公开的基于源代码的漏洞检测结果看，最高漏洞检测精确率仅为 74.33%，检测难度较大。本文基于该数据集生成的二进制数据集进行漏洞检测实验，检测难度比基于源代码更大。

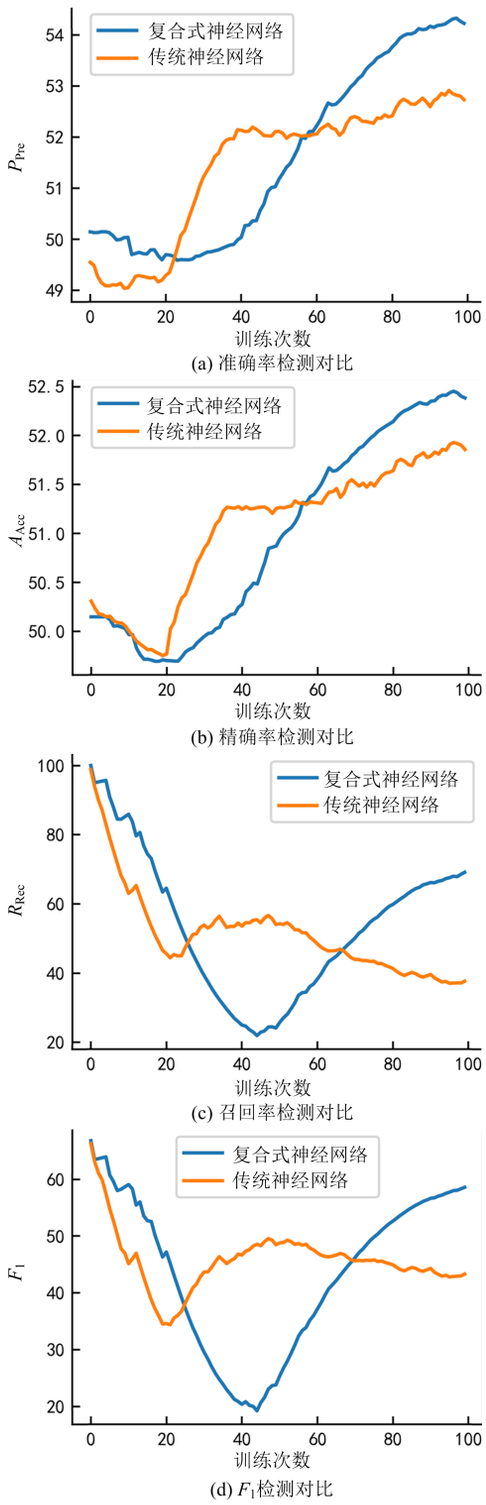


图 5　复合式神经网络模型与传统神经网络模型检测效果对比图(Devign 数据集)

Fig. 5　The detection effect comparison of composite neural network and traditional neural network(Devign DataSet)

由图 5 和表 5 可以看出，传统神经网络模型

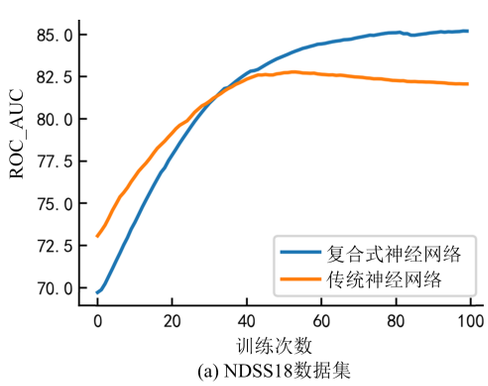
在该数据集上的 A_{Acc}，P_{Pre}，R_{Rec} 和 F_1 指标都表现较差，训练不出有用的检测模型；复合式神经网络模型经过 100 轮训练后在验证集上 F_1 指标为 58.59%，也达不到实际应用的要求，但是相较于传统神经网络模型，各项指标上都有提升，性能更优。因此，在 Devign 数据集上问题 2 成立。

表 5 复合式神经网络与传统神经网络检测效果（Devign 数据集）

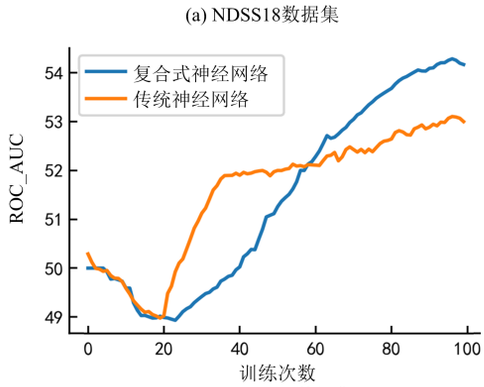
Tab. 5 The detection effect of composite neural network and traditional neural network(Devign DataSet)

方法	A_{Acc}（%）	P_{Pre}（%）	R_{Rec}（%）	F_1（%）
传统神经网络模型	52.92	51.63	39.61	44.31
复合式神经网络模型	54.33	52.45	69.14	58.59

此外，2 组实验的 ROC_AUC 曲线如图 6 所示，复合式神经网络模型在 NDSS18 数据集和 Devign 数据集上的性能均优于传统神经网络模型。总体来看，本文提出的复合式神经网络模型在应用于二进制软件漏洞挖掘方面，相较于传统神经网络模型效果更优。



(a) NDSS18数据集



(b) Devign数据集

图 6 复合式神经网络模型与传统神经网络模型 ROC_AUC 曲线图

Fig. 6 The ROC_AUC curves of composite neural network and traditional neural network

4 结束语

本文针对图神经网络模型在二进制软件漏洞检测领域缺乏研究，以及如何结合图神经网络模型和传统神经网络模型提升漏洞检测效果的问题，提出了一种基于复合式神经网络的二进制软件漏洞检测方法：首先，通过一种数据编码方法将二进制代码向量化表示为同时支持传统神经网络模型和图神经网络模型训练的数据结构，从而更好地保留二进制软件中的各类语义信息，提高模型训练检测效果；然后，使用复合式神经网络模型在二进制软件漏洞数据集进行训练和测试。在理论方法研究的基础上，实现了一个原型系统用于开展实验和对比分析，结果表明：本文提出的复合式神经网络模型能够同时利用两类不同神经网络模型的优势，在精确率、准确率等指标方面较传统神经网络均有所提高，有效提升了针对二进制软件的漏洞检测能力，为未来基于人工智能的二进制软件漏洞检测方法研究作出有益的探索。当然，由于缺乏二进制软件漏洞数据集，本文的研究工作目前仅在 NDSS18 和 Devign 公开数据集上进行了验证，方法的实际有效性还有待进一步证实，未来研究工作包括基于真实源代码软件漏洞数据集构造新的二进制软件漏洞数据集，以及探索不同神经网络模型组合模式的漏洞检测效果等。

参 考 文 献

[1] PERL H, DECHAND S, SMITH M, et al. VCCFinder: finding potential vulnerabilities in open-source projects to assist code audits[C]//Proceedings of ACM Sigsac Conference on Computer and Communications Security. [S. l.: s. n.], 2015: 426-437.

[2] PANG Y, XUE X, WANG H. Predicting vulnerable software components through deep neural network [C]//Proceedings of International Conference. [S. l.: s. n.], 2017: 6-10.

[3] YAN H, SUI Y, CHEN S, et al. Machine-Learning-Guided typestate analysis for static Use-After-Free detection[C]//Proceedings of Computer Security Applications Conference. [S. l.: s. n.], 2017: 42-54.

[4] YAMAGUCHI F. Pattern-based vulnerability discovery [D]. Germany: Georg-August-University of Göttingen, 2015.

[5] LI Z, ZOU D, XU S, et al. VulDeePecker: a deep learning-based system for vulnerability detection[C]// Proceedings of the 25th Annual Network and Distributed System Security Symposium. [S. l. : s. n.], 2018.

[6] LI Z, ZOU D, XU S, et al. SySeVR: a framework for using deep learning to detect software vulnerabilities[J]. IEEE Transactions on Dependable and Secure Computing, 2018, 19(4): 2244-2258.

[7] ZOU D, WANG S, XU S, et al. μVulDeePecker: a deep learning-based system for multiclass vulnerability detection[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(5): 2224-2236.

[8] VulDeePecker Dataset[EB/OL]. (2020-11-8) [2022-08-21]. https://github. com/CGCL-codes/VulDeePecker.

[9] LE T , NGUYEN T , LE T , et al. Maximal divergence sequential autoencoder for binary software vulnerability detection[C]//Proceedings of International Conference on Learning Representations. [S. l. : s. n.], 2019: 1-15.

[10] GAO J, YANG X, FU Y, et al. VulSeeker-pro: enhanced semantic learning based binary vulnerability seeker with emulation[C]//Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. [S. l. : s. n.], 2018: 803-808.

[11] Zheng L C, Shiqi S, Prateek S, et al. Neural nets can learn function type signatures from binaries[C]//Proceedings of the USENIX Security Symposium. Vancouver, BC, Canada: USENIX Association, 2017: 99-115.

[12] WANG H , YE G , TANG Z , et al. Combining graph-based learning with automated data collection for code vulnerability detection[J]. IEEE Transactions on Information Forensics and Security, 2020, 16: 1943-1958.

[13] 陈皓，易平. 基于图神经网络的代码漏洞检测方法[J]. 网络与信息安全学报，2021, 7(3): 37-45.
CHEN Hao, YI Ping. Code vulnerability detection method based on graph neural network[J]. Chinese Journal of Network and Information Security, 2021, 7(3): 37-45. (in Chinese)

[14] Ghidra[EB/OL]. (2022-07-29) [2022-08-21]. https://github. com/NationalSecurityAgency/ghidra.

[15] Deep Graph Library[EB/OL]. (2022-01-23) [2022-08-21]. https://www. dgl. ai/pages/start. html.

[16] ZHOU Y, LIU S, SIOW J, et al. Devign: effective vulnerability identification by learning comprehensive program semantics via graph neural networks[C]// Proceedings of Neural Information Processing Systems. [S. l. : s. n.], 2019.

[17] Devign Binary Dataset[EB/OL]. (2021-02-26)[2022-08-21]. https://github. com/facebookresearch/nbref.

责任编辑　董　莉