doi:10.11835/j.issn.1000.582X.2024.12.007

基于混合风格迁移的智能合约漏洞检测方法

李 敏[1],时瑞浩[2],张 莹[3],袁海兵[4],姜立标[5,6],缪丹云[5]
(1.广汽埃安新能源汽车股份有限公司,广州 510440;2.广州汽车集团股份有限公司,广州 511458;3.星河智联汽车科技有限公司,广州 510335;4.广汽能源科技有限公司,广州 511453;5.广州城市理工学院 机械工程学院与机器人工程学院,广州 510800.6.华南理工大学 机械与汽车工程学院,广州 510641)

摘要:研究提出了一种基于混合风格迁移的智能合约漏洞检测方法,旨在解决智能合约新漏洞出现时数据集不足和无法有效检测未知漏洞问题。首先,从智能合约源代码中提取抽象语法树,使用图注意力网络捕获节点间的依赖关系和信息流;然后,采用最大均值差异实现从旧漏洞到新漏洞的有效知识迁移,从而增加深度学习模型训练的数据量;最后,在分类器中融入MixStyle技术增强模型的泛化能力并提高对新型漏洞类型的识别准确度。实验结果表明,在4种漏洞类型的检测上,该方法在F_1、ACC、MCC指标上优于BLSTM-ATT、BiGAS、Peculiar方法。

关键词:智能合约;漏洞检测;迁移学习;MixStyle;最大均值差异
中图分类号:TP391 **文献标志码:**A **文章编号:**1000-582X(2024)12-070-13

Smart contract vulnerability detection method based on MixStyle transfer

LI Min[1], *SHI Ruihao*[2], *ZHANG Ying*[3], *YUAN Haibing*[4], *JIANG Libiao*[5,6],*MIAO Danyun*[5]
(1.GAC AION New Energy Automobile CO. LTD., Guangzhou 511400, P. R. China; 2. GAC R&D Center, Guangzhou 511458, P. R. China; 3. Syncore Autotech Co., Ltd., Guangzhou 510335, P. R. China; 4. GAC Energy Technology Co., LTD., Guangzhou 511453, P. R. China; 5. School of Mechanical Engineering and School of Robotics Engineering, Guangzhou City University of Technology, Guangzhou 510800, P.R.China;6. School of Mechanical and Automotive Engineering , South China University of Technology, Guangzhou 510641, P.R.China)

Abstract: This study presents a smart contract vulnerability detection method using MixStyle transfer to address challenges related to limited datasets and the detection of unknown vulnerabilities when new ones arise in smart contracts. The method first extracts the abstract syntax tree from the smart contract source code and uses a graph attention network to capture dependencies and information flow between nodes. Then, maximum mean discrepancy(MMD) is used to facilitate effective knowledge transfer from known vulnerabilities to emerging ones, thus expanding the dataset available for deep learning model training. Finally, the MixStyle technique is incorporated into the classifier to enhance model generalization and improve the accuracy of identifying novel vulnerability types. Experimental results show that this method outperforms BLSTM-ATT, BiGAS, and Peculiar

收稿日期:2024-05-08
基金项目:国家自然科学基金(61602345)。
Supported by National Natural Science Foundation of China(61602345).
作者简介:李敏(1984—)男,工程师,主要从事智能网联汽车方向研究,(E-mail)11109493@qq.com。
通信作者:姜立标,男,副教授,(E-mail)jlb620620@163.com。

methods in F1, ACC, and MCC metrics for detecting four types of vulnerabilities.

Keywords: smart contracts; vulnerability detection; transfer learning; MixStyle; maximum mean discrepancy

随着区块链技术的迅速发展,智能合约作为最为革命性的应用之一,已经在金融、法律、物联网等多个领域展现出巨大潜力。智能合约是自动执行、控制或文档化法律事件和行动的计算机程序,其操作完全透明、无需信任的第三方介入,并且具有不可逆转的特性。然而,正如所有新兴技术一样,智能合约带来便利的同时也暴露出多种安全漏洞和风险。由于智能合约的自动执行特性,一旦部署,智能合约代码即在没有人工干预的情况下自动执行,此时区块链无法轻易修复出现问题的智能合约,导致安全事故频发,给用户和机构造成巨大的经济损失,引起社会各界的高度关注。因此,开发高效、准确的智能合约漏洞检测工具和方法,对保障区块链应用的安全性和稳定性具有至关重要的意义。

迄今为止,研究人员开发了许多有效的智能合约漏洞检测工具[1-3],如基于程序分析技术、基于形式化验证技术、基于模糊测试技术和基于符号执行技术等。虽然在检测智能合约漏洞方面普遍有效,但都是基于专家知识,在应用这些方法之前,预先手动总结检测缺陷的规则和模式。然而,由于智能合约的数量在快速增加,仅靠专家总结相应的规则变得耗时、耗力。

针对这一局限,研究者们提出了多种方法,利用深度学习自动学习智能合约缺陷的特征。例如,Smart Embe[1]使用深度学习模型,通过计算与 bug 中合同的相似度来判断检测到的合同是否脆弱。为了利用合约代码中的语法和语义信息,Tann 等[1]提出利用深度学习算法—长短期记忆网络(long short-term memory,LSTM)对智能合约漏洞进行连续学习的方法,较快地发现新的攻击趋势,获得更安全的智能合约。然而,随着智能合约编程语言的发展和新应用场景的出现,新漏洞类型不断涌现,现有方法面临数据集不足和无法有效检测未知漏洞问题。尤其是基于深度学习的智能合约漏洞检测方法,严重依赖标签化训练数据,导致模型在面对少数据量的新漏洞时表现不佳。

为解决上述问题,本研究提出一种基于混合风格迁移的智能合约漏洞检测方法 GMCVD(Graph-based mixstyle transfer for smart contract vulnerability detect),该方法从智能合约源代码中提取抽象语法树(abstract syntax tree,AST),使用图注意力网络(graph attention networks,GAT)捕获节点间的依赖关系和信息流,并在分类器中融入 MixStyle 技术增强模型泛化能力。同时,为了解决新漏洞类型智能合约数据集少的问题,通过迁移学习中的最大均值差异(maximum mean discrepancy,MMD)实现从源合约漏洞(旧漏洞)到目标合约漏洞(新漏洞)的有效知识迁移,增加深度学习模型训练的数据量,提高模型识别新漏洞类型的准确性。经过实验证明,该方法相比现有技术显著提升了模型在面对新漏洞智能合约数据稀缺情况下的检测能力,有效缓解新漏洞类型缺乏足够标注数据的问题。

1　相关工作

1.1　智能合约漏洞检测

传统智能合约漏洞检测主要围绕静态分析和动态分析方法展开。静态分析可进一步划分为形式验证、程序分析和符号执行。例如,Bhargavan 等[7]提出了 1 个形式模型,使用 Isabelle/HOL 工具验证智能合约字节码。Luu 等[8]对智能合约进行符号执行,并根据专家定义的规则检查错误。Tsankov 等[9]进行高级程序分析,以推断智能合约数据流中的语义事实。Gao 等[10]开发了 1 个溢出检测器,该检测器采用基于污点分析的跟踪技术检测以太坊中潜在的溢出漏洞。

动态分析方法通过执行合约代码发现智能合约中的潜在漏洞。Nguyen 等[11]引入了 Serum,利用动态污点跟踪合约执行期间监控的数据流,自动检测和防止高级攻击。Choi 等[12]展示了 sFuzz,通过采用基于分支距离的模糊测试技术识别漏洞。Smartian[13]利用基于数据流的反馈找到有意义的交易序列顺序,触发更多合约状态检测漏洞。

最近的研究已经探索使用深度学习网络进行漏洞检测。Eshghie 等[14]采用序列模型处理智能合约字节

码序列。Zhuang等[15]使用动态漏洞检测框架,从交易数据中提取特征,使用机器学习模型对有害交易进行分类。Liu等[16]提议将合约源代码转换为图结构,构建图神经网络作为检测模型。Zhang等[17]探索将深度学习与专家模式结合,以可解释的方式检测漏洞。Dai等[18]展示了1个混合深度学习模型,结合不同模型特征检测智能合约错误。

1.2 迁移学习

迁移学习近年来受到广泛关注,它可以在目标域标记数据有限的情况下,通过源域的信息促进模型学习。迁移学习在文档分类[19]、机器翻译[20]、视觉脑机接口[21]、数据流分类[22]等领域取得良好成效。迁移学习通常适用于目标场景数据样本量较小,相关领域数据样本丰富的情况。迁移学习可实现低资源学习、领域适应和领域泛化[23]。一般来说,根据是否存在标记数据,迁移学习可分为传导迁移学习、无监督迁移学习和衍生迁移学习[24]。根据特征空间的异同,迁移学习又分为同质迁移学习和异质迁移学习。同质迁移学习是指源域和目标域的分布不同,但数据维度相同。

以前的浅层域适应方法包括重新加权训练数据,使其更接近测试分布[25],在低维流形中找到1个变换,使源和目标子空间更接近[26]。此外,通过在深度特征学习流程中嵌入域适应模块提取域不变表示的研究,已取得最新进展[27]。第1种是基于统计矩匹配的方法,即最大均值差异(maximum mean discrepancy,MMD)[28],中心矩差异(central moment discrepancy,CMD)[29]和二阶统计匹配[30]。第2种常用方法是基于对抗损失的方法,该方法鼓励来自不同域的样本对域标签无歧视,即借鉴生成对抗网络(generative adversarial network,GAN)的域对抗网络基础适应方法[31-32]。

2 本文模型

在本研究中,提出了一种基于迁移学习的智能合约漏洞检测方法GMCVD,旨在有效识别和预测智能合约中的新漏洞类型。GMCVD方法主要包含3个步骤:

1)对智能合约的源码提取代码语义图(semantic graph,SG),使用图注意力网络(graph attention networks,GAT)生成图嵌入。

2)使用最大均值差异(MMD)进行领域适应。

3)使用插入了MixStyle的分类器对合约进行分类。

GMCVD方法的框架如图1所示:首先,模型从智能合约的源代码中提取抽象语法树(abstract syntax tree,AST)。AST是代码结构的树状表现形式,能准确反映程序的语法结构。在获得AST的基础上,进一步构建代码语义图,代码语义图(SG)中包含代码的静态结构信息及整合了变量和函数间的动态交互关系捕捉代码的执行逻辑。获得的代码语义图使用Word2Vec编码,对代码语义图(SG)中的节点进行向量化表示,将代码中的符号和结构转化为连续的向量空间,采用图注意力网络提取节点间复杂的依赖关系。图注意网络通过对邻居节点赋予不同的注意力权重,捕捉节点间的相互作用和信息流。源智能合约的中间特征和

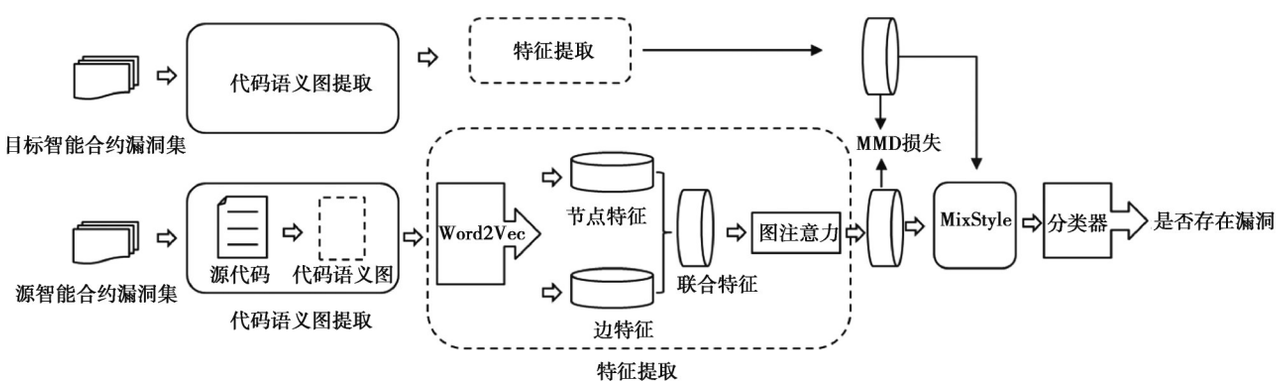

图1 整体框架图

Fig. 1 Overall framework

目标智能合约的中间特征通过计算MMD损失来实现领域适应,通过减小源合约漏洞和目标合约漏洞之间的分布差异,实现源漏洞到目标漏洞之间的有效知识迁移。同时,创新性地将MixStyle技术融合到图卷积神经网络分类器中,该技术通过模拟训练过程中的多样性,使模型学习到更泛化的特征表示,提高对新环境的适应能力。

2.1　构建代码语义图

抽象语法树(AST)包含源代码的语法信息,但缺乏数据流、语句执行顺序等信息,此外,抽象语法树(AST)包含一些不会导致漏洞但可能误导神经网络训练的信息。因此,模型在AST的基础上创建代码语义图(SG),删除无用的节点简化抽象语法树(AST),添加边来表示数据和控制流。代码语义图(SG)的生成包括2个步骤。第1步是构建每个智能合约功能的抽象语法树(AST);第2步是根据抽象语法树(AST)获取代码语义图(SG)。智能合约的抽象语法树(AST)表征了智能合约代码的语法信息,包括:1)数据,如参数及其属性;2)操作,如编程语言提供的语句、运算符、扩展库等。

模型首先构建智能合约函数的抽象语法树,通过算法1所示的算法SG-Generation处理抽象语法树。删除无用的节点简化智能合约抽象语法树,通过在节点之间添加边构建基于简化的抽象语法树的SG,表示智能合约代码中的数据流和控制流。算法SG-Generation的输入是1个抽象语法树,包括语句、运算符、扩展库等。算法SG-Generation的输出是构造的SG=(V,E'),包括节点集V和边集E'。SG-Generation算法包括3个阶段。

第1阶段:通过删除变量名称和类型之外的参数列表和变量属性等无用节点简化抽象语法树(算法1第1行,SG节点集V被初始化为抽象语法树节点集,在节点集V中添加1个新节点Enter来表示SG的开始(算法1第4行),将E'设置为简化抽象语法树的边集,并将有向边< Enter, AST.root >添加到E'中(算法1第5行)。

算法 1: SG − Generation
输入: AST
输出: SG = (V, E')
1.删除AST中的参数列表和变量属性(变量名称和类型除外)
2.初始化V为简化后的AST节点集
3.初始化E'为简化后的AST边集
4.V ← V + Enter　/*添加1个Enter节点标识程序入口*/
5.E' ← E' + < Enter, AST.root >　/*添加1条Enter到AST根节点的边*/
6.ControlFlowGeneration(simplified AST, E')
7.DataFlowEdgeGeneration(simplified AST, E')
8.return V, E'

第2阶段:通过算法2中所示的ControlFlowEdgeGeneration算法构建控制流。ControlFlow EdgeGeneration算法的输入是简化的AST或其子树和边集E',算法对其进行更新,在构建控制流期间,以深度优先搜索(depth-first search,DFS)顺序遍历树,对于节点cn,检查cn是否是1个语句(算法2第4行),如果是,则将1条从cn到下1个兄弟节点的边添加到边集E'中表示语句的执行顺序(算法2第5行),以cn为根的子树递归运行算法ControlFlowEdgeGeneration(算法2第6行),最后,SG就可以表征语句的执行顺序。

第3阶段:通过算法3中所示的算法DataFlowEdgeGeneration构造数据流。算法DataFlowEdge Generation的输入是1棵树和边集E',首先,按照深度优先搜索顺序对树节点进行排序,并将放入列表Ln中,然后遍历Ln来找到相同的变量,在它们之间添加边来表示智能合约代码中的数据流,经过这3个阶段后,算法SG-Generation得到SG=(V,E'),SG的节点又分为语句、运算符、扩展库、变量和其他节点,而边表示控制流和数据流,这样构建的SG不仅包含智能合约代码的语法信息,还可以表征丰富的语义信息,包括数据流、语句执行顺序、指令数据关系等。

算法 2：ControlFlowEdgeGeneration

输入：tree，E'

1. if tree.root has children then /*检查根节点是否有子节点*/
2. $cn \leftarrow$ tree.root.firstchild /*获得根节点的第 1 个子节点*/
3. while cn.nextsibling is not null do /*遍历当前节点的所有兄弟节点，即根节点的所有子节点*/
4. if cn is a statement then /*如果当前节点是 1 个声明*/
5. $E' \leftarrow E' + <cn, cn.\text{nextsibling} >$ with cn being the root，E' /*在边集合中添加 1 条当前节点到它的下 1 个兄弟结点的边，并且这条边以当前节点为根*/
6. end
7. $cn \leftarrow cn$.nextsibling /*移动节点，继续遍历*/
8. end
9 .end

算法 3：DataFlowEdgeGeneration

输入：tree，E'

输出：SG = (V, E')

1. 按深度优先顺序对树节点进行排序，并将它们放入列表 Ln
2. for node in Ln do /*遍历 Ln 中的节点*/
3. if node is a variable then /*判断节点是否为变量*/
4. while node.next is note null do /*遍历当前节点的后续节点*/
5. node' ← node.next /*移动到下1个节点*/
6. if node == node' then /*检查 2 个节点是否是相同的变量*/
7. $E' \leftarrow E' +$ < node，node' > ' /*在 2 个相同变量节点之间添加边*/
8. end
9. end
10. end
11. end

2.2 图嵌入

获得源代码的代码语义图后，模型基于图注意网络(GAT)的架构来学习源代码的高级图语义嵌入 $\mathcal{G}_s$。来提取节点之间的依赖关系。图嵌入提取由 2 个阶段组成，即消息传播阶段和聚合阶段，在消息传播阶段，网络按照代码中的顺序沿边缘传递信息，如在时间步 s，消息流经第 t 时间边 B 并更新 B 末端节点的隐藏状态，此后，GAT 通过关注其邻居来计算每个节点的隐藏状态

$$\vec{h}_i' = \sigma\left(\sum_{j \in N_i} a_{ij} \boldsymbol{W} \vec{h}_j\right), \tag{1}$$

其中：σ 是非线性激活函数；N_i 表示图中节点 i 的邻居；$\boldsymbol{W}$ 是权重矩阵；a_{ij} 表示注意力系数，由下式给出

$$a_{ij} = \frac{\exp\left(\tau\left(\vec{\boldsymbol{a}}^{\mathrm{T}}\left[\boldsymbol{W}\vec{h}_i \oplus \boldsymbol{W}\vec{h}_j\right]\right)\right)}{\sum_{K \in N_i} \exp\left(\tau\left(\vec{\boldsymbol{a}}^{\mathrm{T}}\left[\boldsymbol{W}\vec{h}_i \oplus \boldsymbol{W}\vec{h}_K\right]\right)\right)}, \tag{2}$$

其中：$\oplus$ 表示级联；$\vec{\boldsymbol{a}}$ 是单层 MLP 的权重向量；$\mathcal{T}$ 是 LeakyReLU 函数，依次遍历所有边后，GAT 通过聚合图中所有参与节点的隐藏状态，生成最终的高级图语义嵌入

$$G = \sum_{i=1}^{V} \sigma\left(P_{\text{gate}}\left(\boldsymbol{M}_1 \vec{h}_i' + \boldsymbol{b}_1\right)\right) \odot P\left(\boldsymbol{M}_2 \vec{h}_i' + \boldsymbol{b}_2\right), \tag{3}$$

其中：$\odot$ 表示逐元素乘积；σ 是激活函数；矩阵 $\boldsymbol{M}_j$ 和偏差向量 $\boldsymbol{b}_j$，下标 $j \in \{1,2\}$，是可训练的网络参数。V 表示节点数，P 是多层感知器。

2.3　领域适应

由于不同漏洞之间固有的数据分布不一致,意味着直接将1个场景(源漏洞合约)下训练得到的模型应用到另1个不同场景(目标漏洞合约)可能效果不佳,因为2种场景中的漏洞数据可能存在显著的分布差异。为了克服这一挑战,模型使用最大均值差异(maximum mean discrepancy, MMD)[28]来桥接源漏洞数据和目标漏洞数据之间的分布差异。MMD是一种强大的统计工具,通过再现核希尔伯特空间(reproducing kernel hilbert space, RKHS)来比较和测量2个不同数据分布之间的差异。通过在RKHS中映射数据,MMD计算2个分布的均值的差异,这可以被视为2个分布之间距离的1个度量。在漏洞检测的应用中,模型通过选择合适的核函数来映射源漏洞数据和目标漏洞数据至高维空间,然后计算和最小化这2个高维表示的MMD值。这种方法不仅是非参数的,即不依赖数据分布的具体形式,而且是无监督的,不需要标记数据就能评估分布之间的差异。同时要求源漏洞和目标漏洞的分布在经过图注意力提取之后的中间特征表示下变得相似,模型的目标是优化源漏洞标记数据上的分类误差J_C和源漏洞合约和目标漏洞合约之间的分布差异J_{MMD}。最终的优化目标为

$$\min J_C+\lambda J_{\mathrm{MMD}}\ , \tag{4}$$

其中,λ为正则化参数。

对于分类误差J_C,令$\Theta=\{w,b\}$表示分类器参数的集合,分类器的经验损失为

$$J_C=\frac{1}{m}\sum_{i=1}^{m}J\left(\Theta\left(\boldsymbol{X}_i,\boldsymbol{Y}_i\right)\right), \tag{5}$$

其中:$J(\cdot)$是交叉熵损失函数;$\Theta(X_i)$是分类器将$\boldsymbol{X}_i$分配给标签$\boldsymbol{Y}_i$的条件概率;m表示源漏洞合约实例的数量。

对于分布差异J_{MMD}的测量,采用MMD,可以根据RKHS中2个数据集的样本均值之间的距离来比较不同分布,给定P_{s}和P_{t}作为源漏洞合约和目标漏洞合约的边际分布,m和n作为源漏洞合约和目标漏洞合约实例的数量,令$\boldsymbol{x}_{\mathrm{s}}=\{x_1,\cdots,x_m\}\in\mathbb{R}^{m*d}$表示源漏洞合约的实例,$\boldsymbol{x}_{\mathrm{t}}=\{x_1,\cdots,x_n\}\in\mathbb{R}^{n*d}$表示目标漏洞合约的实例。MMD(又称$J_{\mathrm{MMD}}$)的计算公式为

$$J_{\mathrm{MMD}}=\mathrm{MMD}\left(P_{\mathrm{s}},P_{\mathrm{t}}\right)=\left\|\frac{1}{m}\sum_{i=1}^{m}\phi\left(\boldsymbol{X}_i\right)-\frac{1}{n}\sum_{j=m+1}^{m+n}\phi\left(\boldsymbol{X}_j\right)\right\|_H^2=\frac{1}{m^2}\boldsymbol{K}-\frac{2}{m^2}\boldsymbol{k}^{\mathrm{T}}+\mathrm{const}, \tag{6}$$

其中:$\|\cdot\|_{\mathrm{H}}^2$是RKHS范数;$\boldsymbol{\varphi}(\cdot)$表示RKHS上的数据匹配;$\boldsymbol{k}_{ij}=\boldsymbol{k}(\boldsymbol{X}_i,\boldsymbol{X}_j)$,$\boldsymbol{K}\in\mathbb{R}^{m*m}$,且$k_i=\frac{n}{m}\sum_{j=m+1}^{m+n}\boldsymbol{k}(\boldsymbol{X}_i,\boldsymbol{X}_j)$,在方法中使用的是高斯核函数$k_{ij}=\exp\left(-\frac{\left\|\boldsymbol{X}_i-\boldsymbol{X}_j\right\|^2}{2\sigma}\right)$,其中,$\sigma$是核宽度的超参数,结合式(4)~(6),优化目标函数可表示为

$$\min_{\Theta}\frac{1}{m}\sum_{i=1}^{m}J\left(\Theta\left(\boldsymbol{X}_i,\boldsymbol{Y}_i\right)\right)+\lambda\mathrm{MMD}\left(P_{\mathrm{s}},P_{\mathrm{t}}\right)。 \tag{7}$$

基于上述损失函数的计算方法,可以通过多轮学习提取可迁移特征,通过最小化源和目标数据在特征空间中的MMD值,有效调整源漏洞特征,使其与目标漏洞特征的分布更为一致。原本仅适用于源漏洞检测的分类器通过这种调整后,其检测知识和能力能更好地迁移到目标漏洞上,提高在目标场景中的漏洞检测性能。

2.4　MixStyle

为了提升分类模型的鲁棒性,研究在智能合约的漏洞检测中引入了MixStyle技术。智能合约开发主要关注功能逻辑,而漏洞通常源自运行逻辑的错误。智能合约的图形化是行业常规,通常通过CNN提取图中特征。在图像领域的研究表明,CNN在处理时常忽略底层的风格信息,仅保留高层的语义特征。基于发现,提出了一种新方法,将MixStyle应用于智能合约的图形特征,通过在模型底层混合特征,增强模型对不同类型合约代码的学习能力,提高对未知样本的泛化能力和有效的知识迁移。在具体的实现上,MixStyle被插入到CNN架构中的层之间,将2个实例的特征统计数据与随机凸权重混合。MixStyle模块内部的计算可以概

括为3个步骤。首先,给定2个实例的2组特征图$\boldsymbol{f}$和$\boldsymbol{f}'$,MixStyle计算特征统计量$(\mu(\boldsymbol{f}),\sigma(\boldsymbol{f}))$和$(\mu(\boldsymbol{f}'),\sigma(\boldsymbol{f}'))$,其次,MixStyle生成特征统计量的混合

$$\gamma_{\text{mix}}=\lambda\sigma(\boldsymbol{f})+(1-\lambda)\sigma(\boldsymbol{f}'), \tag{8}$$

$$\beta_{\text{mix}}=\lambda\mu(\boldsymbol{f})+(1-\lambda)\mu(\boldsymbol{f}'), \tag{9}$$

其中,λ是从beta分布中采样的特定于实例的随机权重,$\lambda\sim\text{Beta}(a,a)$,$a\in(0,\infty)$是超参数,实际中设置$\alpha=0.1$。

最后,将特征统计的混合应用于样式归一化的$\boldsymbol{f}$

$$\text{MixStyle}(\boldsymbol{f},\boldsymbol{f}')=\gamma_{\text{mix}}\odot\frac{\boldsymbol{f}-\mu(\boldsymbol{f})}{\sigma(\boldsymbol{f})}+\beta_{\text{mix}}, \tag{10}$$

文中使用0.5的概率来决定在前向传递中是否激活MixStyle。在测试时不使用MixStyle,$\mu(\cdot)$和$\sigma(\cdot)$的计算图中的梯度被阻挡,防止增强效果被擦除。

2.5 分类模块

GMCVD方法获得大小为$K\times\sum_{t=1}^{M}C^t$的矩阵$\boldsymbol{T}$来表示SG节点的特征,t表示第t个CNN层输出的维度,以及大小为$K\times C$的矩阵$\boldsymbol{S}$来表征SG边的特。首先,对$\boldsymbol{T}$和$\boldsymbol{S}$进行按行重塑,分别获得$K(\sum_{t=1}^{M}C^t)\times 1$向量$\tilde{\boldsymbol{T}}$和$KC\times 1$向量$\tilde{\boldsymbol{S}}$,将其连接成矩阵$\boldsymbol{G}$作为全局代码特征。其次,经过MisStyle模块添加扰动之后,输出扰动后的特征,使用Conv1D层在全局代码特征描述符上顺序应用滤波器,其中,滤波器大小和步长均为$\sum_{t=1}^{M}C^t$。应用多个Conv1D层和最大池层从全局代码特征中学习局部模式。最后,采用多个全连接层和1个sigmoid函数来获得分类结果。

3 实验

展示实验相关设置,将GMCVD模型与3个基准方法在公开数据集上进行性能比较以及对GMCVD模型进行消融实验和超参数选择实验。

3.1 实验设置

实验数据集:在评估所提出的方法时,使用了来自Liu等[34]的1个数据集,这个数据集包括了在Etherscan上验证的智能合约。该数据集涉及4种漏洞类型:区块号依赖(block number dependency,BN)、以太币冻结(ether frozen,EF)、危险的以太币严格等值(dangerous ether strict equality,SE)和时间戳依赖(timestamp dependency,TP)。详细信息汇总在表1中。

数据集使用设置:采用对比实验来衡量GMCVD方法的性能。在实验中,GMCVD方法使用完整的源智能合约,并假设目标合约中有10%的合约被标记,剩余90%未标记,模拟新漏洞数据有限的场景。为了全面评估GMCVD方法和基线方法的有效性,在目标域合约的第2阶段训练中采用了十折交叉验证。将目标域数据分为10个子集,交替使用1个子集(10%)作为训练集,其余9个子集(90%)作为测试集。这种方法能够获得一种漏洞类型的10个测试结果,每种类型的最终结果是通过平均这些结果计算得出的。值得注意的是,与GMCVD方法不同,基线模型使用10%标记的智能合约数据进行训练,并预测同一类型智能合约中的漏洞。

表1 从智能合约数据集中选择4种漏洞类型

Table 1 Four vulnerability types chosen from smart contract dataset

漏洞类型	BN	EF	SE	TP
无漏洞样本	1 676	324	1 938	2 523
有漏洞样本	128	84	50	310
样本总数	1 804	408	1 988	2 833

实验环境:Intel Core i7 11700 CPU,16GB RAM 和 1 个 GPU(NVIDIA RTX 3070)8 GB 内存的服务器,操作系统为 Ubuntu 20.04 LTS。

基准方法:将 GMCVD 方法与 3 种创新的深度学习技术进行比较,用于识别智能合约中的漏洞。

BLSTM-ATT [35]:BLSTM-ATT 是基于 Word2Vec 的一种先进方法,使用 Word2Vec 学习语义特征,并将其与双向长短期记忆网络(long short-term memory,LSTM)和注意力机制结合起来。

BiGAS[36]:BiGAS 也是基于 Word2Vec 的一种先进方法,利用 Word2Vec 学习语义特征,并将其与双向 GRU(Bi-GRU)结合,使用支持向量机(support vector machine,SVM)作为分类器。

Peculiar[37]:Peculiar 是近年来引入的一种创新方法,依赖于预训练的语言模型。通过 GraphCodeBERT 适应关键数据流,学习语义和结构信息。

消融实验:为了评估 GMCVD 方法中各个模块的性能,设计并进行了消融实验,涉及 2 种不同的场景。第 1 种场景是移除 MixStyle 模块,为了评估 MixStyle 在学习不同合约代码的“风格特征”方面的作用。第 2 种场景是移除迁移学习的 MMD 模块,衡量 MMD 在拟合源域数据和目标数据之间的分布差异方面的效果。这些消融实验的设置和参数与比较实验保持一致,并且同样采用了十折交叉验证策略。

超参数选择实验:为了确定模型中最优的超参数设置,进行了一系列的超参数选择实验。第 1 种场景探索了不同 β 分布形状参数 α 对模型性能的影响。通过调整 α 值,评估了其对模型性能的优化程度,找到能够最大化模型性能的最佳 α 值。第 2 种场景则测试了不同的高斯核宽度 σ,评估其对模型训练效率和准确率的影响。选择适当的 σ 值是关键,直接影响到模型在处理数据分布差异时的灵敏度和效果。

评价指标:为了评估各种方法的性能,采用了一系列在智能合约漏洞检测研究中得到广泛认可的指标来评估不同方法的性能。这些指标包括 F_1 度量、准确度(accuracy,ACC)、马修斯相关系数(matthews correlation coefficient,MCC)和接收者工作特征曲线(ROC)下面积(AUC)。这些指标的定义如下

- True Positive (TP):被准确识别为脆弱性的实例;
- True Negative(TN):被正确归类为非漏洞的实例;
- False Positive(FP):被错误地标记为漏洞的非漏洞实例;
- False Negative(FN):被错误归类为非漏洞的缺陷实例。

在所有被预测为阳性的实例中,被正确识别的实际阳性实例所占比例用 Precision 来衡量,计算公式为

$$\text{Precision}=\frac{TP}{TP+FP}, \tag{11}$$

Recall 评估准确识别的实际阳性比例,定义如下

$$\text{Recall}=\frac{TP}{TP+FN}, \tag{12}$$

F_1 度量是 Precision 和 Recall 的调和平均值,它提供了模型准确性和完整性的平衡视角

$$F_1=\frac{2\times \text{Precision}\times \text{Recall}}{\text{Precision}+\text{Recall}}, \tag{13}$$

ACC 评估一种方法在二元分类中的整体效果,反映正确分类的正例和负例的比例

$$\text{ACC}=\frac{TP+TN}{TP+TN+FP+FN}, \tag{14}$$

MCC 提供了一种平衡评价,在处理不平衡数据集时尤其有用

$$\text{MCC}=\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TN+FP)(TN+FN)(TP+FN)}}。 \tag{15}$$

3.2　实验分析

将所提出的方法与 3 种现有基线方法在 4 种漏洞类别上进行比较分析,详细信息列在表 2~4 中。实验部分评估了每种漏洞类型的方法性能,并计算了每种方法的 F_1 度量、准确率(ACC)和马修斯相关系数(MCC)的平均值。随后,为了进一步评估平均改进程度,并通过胜/平/负(W/T/L)统计显示 GMCVD 方法的优越性。值

得注意的是，GMCVD方法显示出相当的效力，其平均F_1度量，ACC和MCC值分别达到了69.2，93.5和66.2。

表2 GMCVD模型与基准方法在F_1上的性能对比

Table 2 Comparison of the performance of the GMCVD model and the baseline method on F_1

模型	BLSTM-ATT	BiGAS	Peculiar	**GMCVD**
BN	40.1	29.2	20.1	**65.8**
EF	44.2	45.1	43.5	**72.5**
SE	68.5	31.2	20.6	**75.1**
TP	52.5	52.1	48.2	**63.2**
平均	51.3	39.4	33.1	**69.2**
提升/%	34.8	75.6	109.1	—
W/T/L	3/0/0	3/0/0	3/0/0	—

注：加粗数据表示最优性能。

表3 GMCVD模型与基准方法在ACC上的性能对比

Table 3 Comparison of the performance of the GMCVD model and the baseline method on ACC

模型	BLSTM-ATT	BiGAS	Peculiar	**GMCVD**
BN	90.2	92	54.5	**94.6**
EF	76.8	84.5	67.6	**88.3**
SE	97.5	96.6	70.5	**98.9**
TP	87.5	82.4	75.8	**92.3**
平均	88.0	88.9	67.1	**93.5**
提升/%	6.3	5.2	39.3	—
W/T/L	3/0/0	3/0/0	3/0/0	—

注：加粗数据表示最优性能。

表4 GMCVD模型与基准方法在MCC上的性能对比

Table 4 Comparison of the performance of the GMCVD model and the baseline method on MCC

模型	BLSTM-ATT	BiGAS	Peculiar	**GMCVD**
BN	36.5	25	22.4	**63.0**
EF	30.8	42.3	32.4	**65.9**
SE	65.8	38.9	25.4	**76.8**
TP	53.4	48.5	46.7	**58.9**
平均	46.6	38.7	31.7	**66.2**
提升/%	42.0	71.2	108.7	—
W/T/L	3/0/0	3/0/0	3/0/0	—

注：加粗数据表示最优性能。

当GMCVD与基于语义的深度学习方法BLSTM-ATT和BiGAS直接比较时，在所有4种漏洞数据类型上均表现出色。具体来说，GMCVD至少在F_1度量上实现了34.8%的显著提升，在ACC上至少提高了5.2%，在MCC上至少增强了42.0%。与基于GraphCodeBERT的方法Peculiar相比，GMCVD在各种评估指标上显示出显著的性能提升。特别是在所有漏洞类型上的平均F_1度量提高了109.1%，ACC提高了39.3%，MCC提高了108.7%。为了更加直观地体现GMCVD优势，图2给出了对比不同算法的迁移效果。

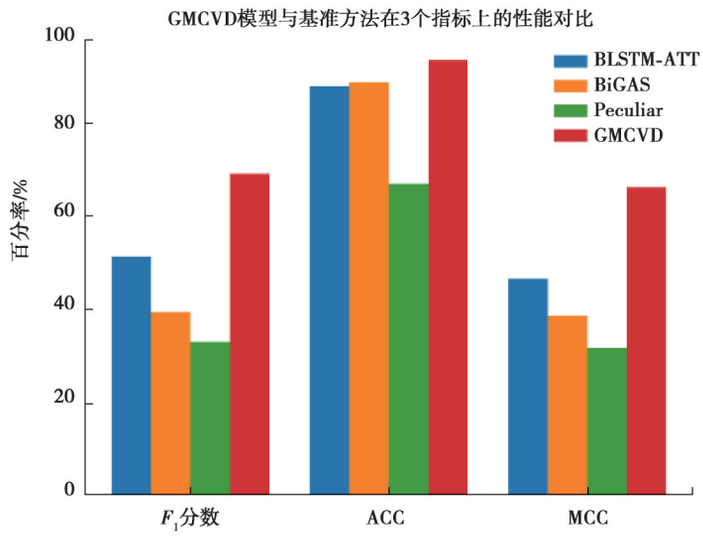

图 2　不同算法的迁移效果图

Fig. 2　Migration effect diagram of different algorithms

基于上述分析,可以得出,在缺乏特定漏洞数据的情况下,GMCVD有效利用了来自其他漏洞的标记数据知识,并实现了与最先进方法相当的性能。在研究中,对比分析了所提出的GMCVD方法与2个消融方法在4种漏洞类别上的性能,详细信息列在表5~7中。在与移除MixStyle模块(Non-MixStyle)的直接比较中,GMCVD在所有4种漏洞类型上均展现了卓越的性能。具体来说,GMCVD在F_1度量上至少实现了13.6%的显著提升,在ACC上至少提高了1.5%,在MCC上至少增强了15.2%。结果明显表明,MixStyle模块在提升模型适应不同代码风格的能力上发挥了至关重要作用。通过学习和应用不同合约代码的风格特征,MixStyle显著增强了模型在处理多样化数据时的鲁棒性和效率。

表 5　GMCVD模型与消融方法在F_1上的性能对比

Table 5　Comparison of the performance of the GMCVD model and the ablation method on F_1

模型	Non-MixStyle	Non-MMD	**GMCVD**
BN	52.9	55.0	**65.8**
EF	60.8	62.3	**72.5**
SE	72.7	72.8	**75.1**
TP	57.2	58.0	**63.2**
平均	60.9	62.0	**69.2**
提升/%	13.6	11.6	—
W/T/L	3/0/0	3/0/0	—

注:加粗数据表示最优性能。

表 6　GMCVD模型与消融方法在ACC上的性能对比

Table 6　Comparison of the performance of the GMCVD model and the ablation method on ACC

模型	Non-MixStyle	Non-MMD	**GMCVD**
BN	94.2	93.2	**94.6**
EF	85.2	86.2	**88.3**
SE	98.7	98.7	**98.9**
TP	90.3	90.1	**92.3**
平均	92.1	92.1	**93.5**
提升/%	1.5	1.6	—
W/T/L	3/0/0	3/0/0	—

注:加粗数据表示最优性能。

表7 GMCVD模型与消融方法在MCC上的性能对比

Table 7 Comparison of the performance of the GMCVD model and the ablation method on MCC

模型	Non-MixStyle	Non-MMD	**GMCVD**
BN	50.0	55.0	**63.0**
EF	52.3	60.5	**65.9**
SE	75.2	76.0	**76.8**
TP	52.3	55.5	**58.9**
平均	57.5	61.8	**66.2**
提升/%	15.2	7.2	—
W/T/L	3/0/0	3/0/0	—

注:加粗数据表示最优性能。

与移除MMD模块(Non-MMD)相比,GMCVD在各项评估指标上也显示出显著的性能提升。在所有漏洞类型上,GMCVD的平均F_1度量提高了11.6%,ACC提高了1.6%,MCC提高了7.2%。结果强调MMD模块在较少源域和目标域之间的分布差异,提升模型性能。MMD模块通过调整和适应不同域间的数据分布,有效提高了模型对目标域数据的预测准确性,证明其在实现跨漏洞的智能合约漏洞检测的关键作用。图3直观给出了不同组件消融的效果。

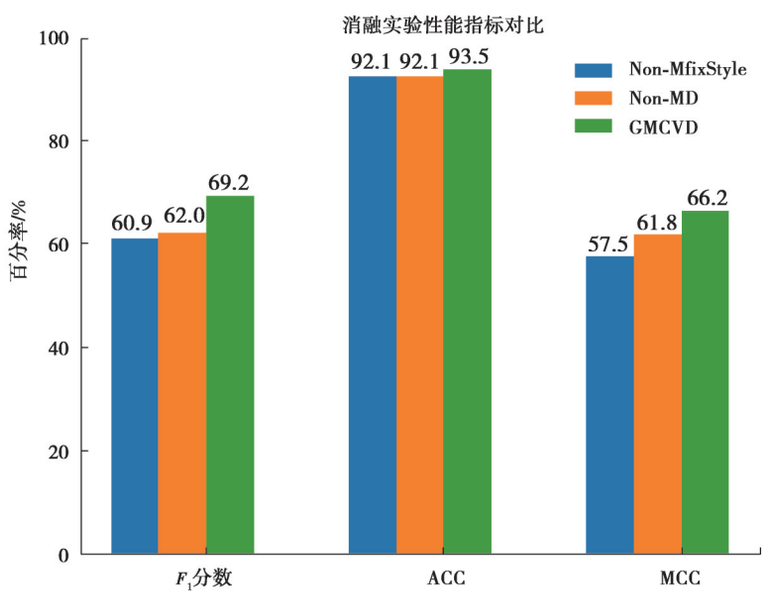

图3 GMCVD模型和消融方法在3个指标上的性能比较

Fig. 3 Comparison of the performance of the GMCVD model and the ablation method on three metrics

从上述分析可以看出,每个模块在GMCVD方法中都扮演了不可或缺的角色。MixStyle模块通过增强模型对不同编程风格的适应能力,提高了模型的泛化性。同时,MMD模块的引入则确保了模型能有效处理来自不同源域的数据,优化模型在新环境中的应用效果。这些模块的协同工作显著提升了整体模型性能,使GMCVD在跨漏洞的智能合约漏洞检测中展现出高效与精准的双重优势。

在超参数选择实验中,观察到β分布形状参数α的选择对ACC有显著影响,如图4所示,α值在0.1时准确率最高,随着α的增加,ACC,F_1度量和MCC呈现下降趋势,表明较低的α值有助于模型性能优化。在高斯核宽度σ的选择方面,如图5所示,σ值为5时,F_1度量、ACC和MCC均达到近似最优。结果表明,适中的σ值能够平衡模型的细节捕捉能力与泛化能力。过大或过小的σ值均对模型的性能表现不佳,因为高σ值导致模型丢失细节,而低σ值则过度强调这些细节,均会损害模型的泛化能力。

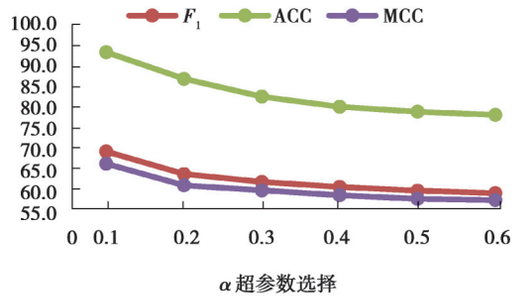

图 4　不同 α 超参数的选择对 GMCVD 模型性能的影响

Fig. 4　Effect of different choices of α hyper parameters on the performance of GMCVD models

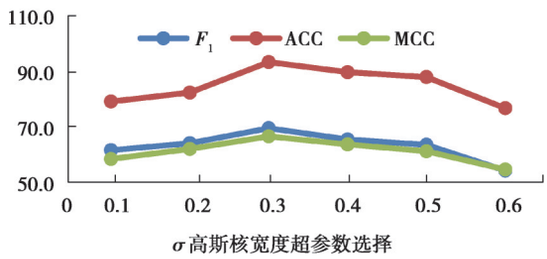

图 5　不同 σ 高斯核宽度超参数的选择对 GMCVD 模型性能的影响

Fig. 5　Effect of the choice of hyper parameters with different σ Gaussian kernel widths on the performance of the GMCVD model

4　结束语

在这项工作中,提出了名为 GMCVD 的创新方法,用于跨域智能合约漏洞检测。通过从智能合约源代码中提取 AST,构建了代码语义图,这 2 者为智能合约的结构和语义提供了深入的表示。利用图注意力机制来分析代码语义图,捕捉代码中的关键特征和节点间的复杂依赖关系。采用 MMD 来调整和优化源域和目标域之间的分布差异,确保模型能在不同的域中有效识别漏洞。最后,引入了 MixStyle 技术,通过在模型训练中模拟特征分布的多样性,增强了模型对新环境的适应能力和泛化性。实验表明,所提出的 GMCVD 方法在 4 种漏洞类型的 3 种评价指标上优于现有的主流方法。

在未来的工作中,计划评估所提出的 GMCVD 方法在不同场景下的性能,并探索其他域适应方法的整合,以进一步提高其性能。这些措施预计将使 GMCVD 方法在智能合约安全领域更加高效和实用。

参考文献

[1] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, et al. Smartcheck: static analysis of ethereum smart contracts[C]//1st International Workshop on Emerging Trends in Software Engineering for Blockchain. New York:ACM, 2018: 9-16.

[2] Tsankov P, Dan A, Drachsler-Cohen D, et al. Securify: practical security analysis of smart contracts[C]//2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2018: 67-82.

[3] Jiang B, Liu Y, Chan W K. Contractfuzzer: fuzzing smart contracts for vulnerability detection[C]//33rd ACM/IEEE International Conference on Automated Software Engineering. Piscataway:IEEE, 2018: 259-269.

[4] Torres C F, Schütte J, State R. Osiris: hunting for integer bugs in ethereum smart contracts[C]//34th Annual Computer Security Applications Conference. New York: ACM, 2018: 664-676.

[5] Gao Z, Jayasundara V, Jiang L, et al. Smartembed: A tool for clone and bug detection in smart contracts through structural code embedding[C]//2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). Piscataway,: IEEE, 2019: 394-397.

[6] Tann W J W, Han X J, Gupta S S, et al. Towards safer smart contracts: a sequence learning approach to detecting security threats[EB/OL]. (2019-06-07)[2024-05-12].https://arxiv.org/abs/1811.06632.

[7] Bhargavan K, Delignat-Lavaud A, Fournet C, et al. Formal verification of smart contracts: Short paper[C]//2016 ACM Workshop on Programming Languages and Analysis for Security, 2016: 91-96.

[8] Luu L, Chu D H, Olickel H, et al. Making smart contracts smarter[C]//2016 ACM SIGSAC Conference on Computer and Communications Security. New York:ACM, 2016: 254-269.

[9] Tsankov P, Dan A, Drachsler-Cohen D, et al. Securify: practical security analysis of smart contracts[C]//2018 SIGSAC Conference on Computer and Communications security. NewYork:ACM, 2018: 67-82.

[10] Gao J, Liu H, Liu C, et al. Easyflow: keep ethereum away from overflow[C]//2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). Piscataway: IEEE, 2019: 23-26.

[11] Nguyen T D, Pham L H, Sun J, et al. sfuzz: an efficient adaptive fuzzer for solidity smart contracts[C]//ACM/IEEE 42nd International Conference on Software Engineering. Piscataway:IEEE, 2020: 778-788.

[12] Choi J, Kim D, Kim S, et al. Smartian: enhancing smart contract fuzzing with static and dynamic data-flow analyses[C]//2021

36th IEEE/ACM International Conference on Automated Software Engineering (ASE). Piscataway: IEEE, 2021: 227-239.
[13] Liu Z, Qian P, Yang J, et al. Rethinking smart contract fuzzing: Fuzzing with invocation ordering and important branch revisiting[J]. IEEE Transactions on Information Forensics and Security, 2023, 18: 1237-1251.
[14] Eshghie M, Artho C, Gurov D. Dynamic vulnerability detection on smart contracts using machine learning[C]//25th International Conference on Evaluation and Assessment in Software Engineering. New York: ACM, 2021: 305-312.
[15] Zhuang Y, Liu Z, Qian P, et al. Smart contract vulnerability detection using graph neural networks[C]//International Conference on International Joint Conferences on Artificial Intelligence. San Francisco, CA:Margan Kaufmann, 2021: 3283-3290.
[16] Liu Z, Qian P, Wang X, et al. Smart contract vulnerability detection: from pure neural network to interpretable graph feature and expert pattern fusion[EB/OL].(2021-06-17)[2024-05-12].https://arxiv.org/abs/2106.09282.
[17] Zhang L, Chen W, Wang W, et al. Cbgru: a detection method of smart contract vulnerability based on a hybrid model[J]. Sensors, 2022, 22(9): 3577.
[18] Dai W, Xue G R, Yang Q, et al. Co-clustering based classification for out-of-domain documents[C]//13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. NewYork：ACM，2007: 210-219.
[19] Maimaiti M, Liu Y, Luan H, et al. Enriching the transfer learning with pre-trained lexicon embedding for low-resource neural machine translation[J]. Tsinghua Science and Technology, 2021, 27(1): 150-163.
[20] Lin J, Liang L, Han X, et al. Cross-target transfer algorithm based on the volterra model of SSVEP-BCI[J]. Tsinghua Science and Technology, 2021, 26(4): 505-522.
[21] Wu Q, Wu H, Zhou X, et al. Online transfer learning with multiple homogeneous or heterogeneous sources[J]. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(7): 1494-1507.
[22] Zhuang F, Qi Z, Duan K, et al. A comprehensive survey on transfer learning[J]. IEEE Access, 2020, 109(1): 43-76.
[23] Pan S J, Tsang I W, Kwok J T, et al. Domain adaptation via transfer component analysis[J]. IEEE Transactions on Neural Networks, 2010, 22(2): 199-210.
[24] Jiang J, Zhai C X. Instance weighting for domain adaptation in NLP[C]//45th Annual Meeting of the Association Computational Linguistics. Prague:ACL, 2007:264-271.
[25] Wang J, Chen Y, Feng W, et al. Transfer learning with dynamic distribution adaptation[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2020, 11(1): 1-25.
[26] Zhu Y, Zhuang F, Wang J, et al. Multi-representation adaptation network for cross-domain image classification[J]. Neural Networks, 2019, 119: 214-221.
[27] Long M, Cao Y, Wang J, et al. Learning transferable features with deep adaptation networks[C]//International conference on Machine Learning. Cambridge MA: JMLR, 2015: 97-105.
[28] Zellinger W, Grubinger T, Lughofer E, et al. Central moment discrepancy (cmd) for domain-invariant representation learning [EB/OL].(2019-05-02)[2024-05-12].https://arxiv.org/abs/1702.08811.
[29] Sun B, Saenko K. Deep coral: Correlation alignment for deep domain adaptation[C]//Computer Vision-ECCV 2016 Workshops. Amsterdam, Netherlands: Springer, 2016: 443-450.
[30] Ganin Y, Ustinova E, Ajakan H, et al. Domain-adversarial training of neural networks[J]. Journal of Machine Learning Research, 2016, 17(59): 1-35.
[31] Hoffman J, Tzeng E, Park T, et al. Cycada: cycle-consistent adversarial domain adaptation[C]//International Conference on Machine Learning. Cambridge MA: JMLR, 2018: 1989-1998.
[32] Tzeng E, Hoffman J, Saenko K, et al. Adversarial discriminative domain adaptation[C]//IEEE conference on computer vision and pattern recognition. Piscataway:IEEE, 2017: 7167-7176.
[33] Zhou K, Yang Y, Qiao Y, et al. Mixstyle neural networks for domain generalization and adaptation[J]. International Journal of Computer Vision, 2024, 132(3): 822-836.
[34] Liu Z, Qian P, Yang J, et al. Rethinking smart contract fuzzing: fuzzing with invocation ordering and important branch revisiting[J]. IEEE Transactions on Information Forensics and Security, 2023, 18: 1237-1251.
[35] Qian P, Liu Z, He Q, et al. Towards automated reentrancy detection for smart contracts based on sequential models[J]. IEEE Access, 2020, 8: 19685-19695.
[36] Zhang L, Li Y, Guo R, et al. A novel smart contract reentrancy vulnerability detection model based on BiGAS[J]. Journal of Signal Processing Systems, 2023: 1-23.
[37] Wu H, Zhang Z, Wang S, et al. Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques[C]//2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE). Piscataway: IEEE, 2021: 378-389.

（编辑 侯 湘）