源代码处理任务中的深度学习模型对抗攻防研究综述

潘海为[1] 马宝英[1,2] 张可佳[1] 杨晓阳[1] 秦颖鑫[1] 卢国强[2] 范书平[3]

摘 要 随着智能软件的发展, 深度学习模型在缺陷检测与定位等源代码处理任务中的应用日益广泛, 但其鲁棒性不足的问题也逐渐凸显. 众多学者对源代码对抗攻击与防御方法进行深入研究. 然而, 现有综述鲜有从源代码任务特性出发总结模型特点, 也缺乏对模型窃取、后门防御和防御蒸馏等典型对抗攻防方法的梳理与分析. 本文从模型架构视角入手, 首先系统归纳面向源代码处理任务的深度学习模型, 分析其在对抗攻击环境下的表现与适应性. 随后对源代码对抗攻击与防御方法进行全面分类与综述, 并汇总相关基准数据集. 最后分析现有研究的不足, 提出未来的潜在研究方向.

关键词 深度学习; 源代码处理; 对抗攻防方法; 鲁棒性

引用格式 潘海为, 马宝英, 张可佳, 杨晓阳, 秦颖鑫, 卢国强, 范书平. 源代码处理任务中的深度学习模型对抗攻防研究综述. 自动化学报, 2026, **52**(4): 638−665

DOI 10.16383/j.aas.c250331 **CSTR** 32138.14.j.aas.c250331

Survey on Adversarial Attack and Defense Methods for Deep Learning Models in Source Code Processing Tasks

PAN Hai-Wei[1] MA Bao-Ying[1,2] ZHANG Ke-Jia[1] YANG Xiao-Yang[1]
QIN Ying-Xin[1] LU Guo-Qiang[2] FAN Shu-Ping[3]

Abstract With the development of intelligent software, the application of deep learning models in source code processing tasks, such as defect detection and localization, has become increasingly widespread. But their lack of robustness has also become increasingly evident. Many researchers have conducted in-depth studies on adversarial attack and defense methods for source code. However, existing surveys rarely summarize model characteristics from the perspective of source code task-specific properties, and there is a lack of systematic review and analysis of typical adversarial attack and defense methods such as model stealing, backdoor defense, and defensive distillation. Firstly, from the perspective of model architecture, we systematically outline deep learning models for source code processing tasks, and analyze their performance and adaptability under adversarial attack environments. Subsequently, we conduct a comprehensive review and classification of adversarial attack and defense methods for source code, and summarize the relevant benchmark datasets. Finally, we analyze the limitations of existing research and propose potential directions for future research.

Keywords deep learning; source code processing; adversarial attack and defense methods; robustness

Citation Pan Hai-Wei, Ma Bao-Ying, Zhang Ke-Jia, Yang Xiao-Yang, Qin Ying-Xin, Lu Guo-Qiang, Fan Shu-Ping. Survey on adversarial attack and defense methods for deep learning models in source code processing tasks. *Acta Automatica Sinica*, 2026, **52**(4): 638−665

近年来, 深度学习 (deep learning, DL) 技术广泛应用于医疗诊断[1]、交通预测[2]、农业管理与决策[3] 以及机器翻译[4] 等场景, 随着研究的不断深入, 其影响力逐步扩展到软件工程领域[5]. 在源代码处理任务中, 以循环神经网络、卷积神经网络以及 Transformer 为代表的深度学习架构优势显著, 这些深度学习架构能够有效处理源代码的结构和语义信息, 并应用到多种代码处理任务中, 如源代码功能分类[6]、缺陷检测[7]、代码克隆检测[8]、代码注释生成[9] 以及缺陷定位[10] 等.

尽管深度学习模型提升了源代码处理任务的自

收稿日期 2025-07-17 录用日期 2025-11-14
Manuscript received July 17, 2025; accepted November 14, 2025
黑龙江省重点研发计划 (2024ZXDXA09), 船舶 CAE 软件典型场景研究与应用项目 (CBZ01N23-02), 黑龙江省自然科学基金 (PL2024F022) 资助
Supported by Key Research and Development Program of Heilongjiang Province (2024ZXDXA09), the Research and Application Project of Typical Scenarios for Ship CAE Software (CBZ01N23-02), and Natural Science Foundation of Heilongjiang Province (PL2024F022)
本文责任编委 赫然
Recommended by Associate Editor HE Ran
1. 哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001 2. 牡丹江医科大学卫生管理学院 牡丹江 157011 3. 牡丹江师范学院计算机与信息技术学院 牡丹江 157011
1. College of Computer Science and Technology, Harbin Engineering University, Harbin 150001 2. College of Health Management, Mudanjiang Medical University, Mudanjiang 157011 3. School of Computer and Information Technology, Mudanjiang Normal University, Mudanjiang 157011

动化水平和效率, 但其易受到对抗攻击的影响[11]. 攻击者通过精心设计的扰动生成对抗样本, 能有效诱导模型产生偏差, 这揭示了深度学习模型在鲁棒性方面的不足[12−15]. 与图像和文本等数据上的对抗攻击相比[16−18], 源代码因其具有结构化表示、严格的语法规范与语义约束以及离散性等特点, 导致深度学习模型的设计和对抗攻击技术具有特殊性[19−20]. 具体表现为以下三个方面: 一是源代码的语法结构和语义信息通常是以抽象语法树 (abstract syntax tree, AST)、控制流图或数据流图等结构化的方式来表示, 对代码的扰动操作需要借助静态分析工具等手段进行验证; 二是源代码具有严格的语法规范与语义约束, 任何扰动需要在满足语法正确的同时, 又要保证扰动前后代码语义不变; 三是源代码的扰动空间呈离散分布状态, 往往针对变量名或函数名等离散元素, 并且受到语法和语义的约束, 从而进一步压缩了对抗样本的生成空间. 这些独有特点导致源代码的建模机制与对抗攻击技术相比于图像和文本更具挑战性.

源代码的上述特性决定深度学习模型在处理相关任务时, 必须结合代码的语法结构和语义信息[21−22], 同时也影响该领域对抗攻击技术的设计策略. 与 NLP 中主要围绕词替换或语法扰动的攻击方法不同, 面向源代码处理任务的对抗攻击需要在语法正确和功能不变的前提下构造对抗样本. 其常用构造方式包括重命名标识符、插入冗余语句以及结构等价转换等扰动操作, 以诱导模型预测错误. 这种语法与语义规则的双重约束增加了源代码对抗攻击的难度, 也使深度学习模型的鲁棒性面临更大的挑战.

近年来, 针对源代码处理任务中的深度学习模型及其鲁棒性问题, 已有较多综述从不同视角展开探索[21−30]. 相关工作主要聚焦于三个方面: 源代码表示方法与建模机制、深度学习模型的鲁棒性研究以及对抗攻防技术的研究. 在源代码建模方面, 文献 [21−25] 从代码表示方法、模型构建机制以及典型处理任务等维度出发, 系统总结了当前源代码处理技术的研究进展. 在模型鲁棒性研究方面, 文献 [26] 侧重研究漏洞预测场景下通过干扰数据和模型交互过程, 观察攻击前后鲁棒性验证指标的变化量, 以衡量 AI 系统的鲁棒性; 文献 [27] 则概述语言模型在代码智能任务中面临的数据分布不平衡、标签噪声等问题, 并提出相应的解决方法. 在对抗攻防技术研究方面, 文献 [28] 着重探讨后门攻击与防御技术以及对抗攻击与防御技术, 按照不同技术类别对所收集的论文进行系统的梳理和总结; 文献 [29] 梳理深度学习赋能下的恶意代码攻防关键技术; 文献 [30] 则聚焦于面向鲁棒性的对抗攻击策略研究. 这些研究为理解源代码处理任务中的深度学习模型应用与鲁棒性研究提供了重要参考.

然而, 以上针对源代码处理任务的对抗性研究综述仍存在不足. 一方面, 尽管该领域已广泛采用多种类型的深度学习模型, 但相关综述普遍缺乏从源代码任务特性出发, 对这些模型进行系统且有针对性的分类分析[21−25, 27−28]. 因此, 难以全面体现源代码任务在建模层面的独特需求. 另一方面, 现有源代码对抗攻防方面综述多集中于恶意代码检测等特定应用场景 (见文献 [29]), 缺乏对模型窃取与后门防御等典型对抗攻防技术的探讨[26, 30], 导致当前源代码对抗攻防的分类体系仍不够完善.

针对上述文献存在的不足, 本文围绕源代码处理任务, 系统研究并分析深度学习模型及其对抗攻防方法. 首先, 鉴于当前研究在面向源代码处理任务的深度学习模型 (deep learning for source code processing, DL-SCP) 分类方面仍存在不足, 本文从语义建模视角出发, 深入梳理面向源代码处理任务的深度学习模型, 并将其划分为语义对齐模型、程序行为模型与单模态语义模型三大类. 其次, 为支撑对抗攻防方法的研究与验证, 总结当前常用的源代码数据集, 明确其任务类型与适用范围; 在此基础上, 系统归纳面向 DL-SCP 的对抗攻击方法, 并对其进行细粒度划分, 涵盖规避攻击、中毒攻击与模型窃取三类方法; 针对上述攻击威胁, 进一步梳理面向 DL-SCP 中的防御方法, 重点分析对抗训练、后门防御、防御蒸馏与数据检测四类方法. 随后, 深入剖析这些攻防方法的原理、技术实现、适用范围与局限性, 以有效解决现有综述在源代码对抗攻防体系覆盖不足的问题. 最后, 提出 DL-SCP 对抗攻防研究面临的关键挑战, 并展望未来可能的研究方向, 旨在为该领域的深入发展提供理论支持与方法参考.

本文结构框架如图 1 所示. 第 1 节阐述对抗攻击及对抗防御的基本概念; 第 2 节从语义对齐模型、程序行为模型以及单模态语义模型的角度, 介绍当前主流的面向源代码处理任务的深度学习模型及其存在的局限性; 第 3 节概述面向 DL-SCP 的典型对抗攻击方法; 第 4 节详细探讨针对 DL-SCP 对抗攻击的防御策略; 第 5 节介绍常用于 DL-SCP 对抗攻防研究的数据集; 第 6 节讨论面临的关键挑战与未来研究展望; 第 7 节进行总结.

1 相关概念和理论

为便于后续讨论, 本节首先介绍对抗攻击与防御技术的基本概念, 并对本文中使用的符号进行解释, 具体见表 1.

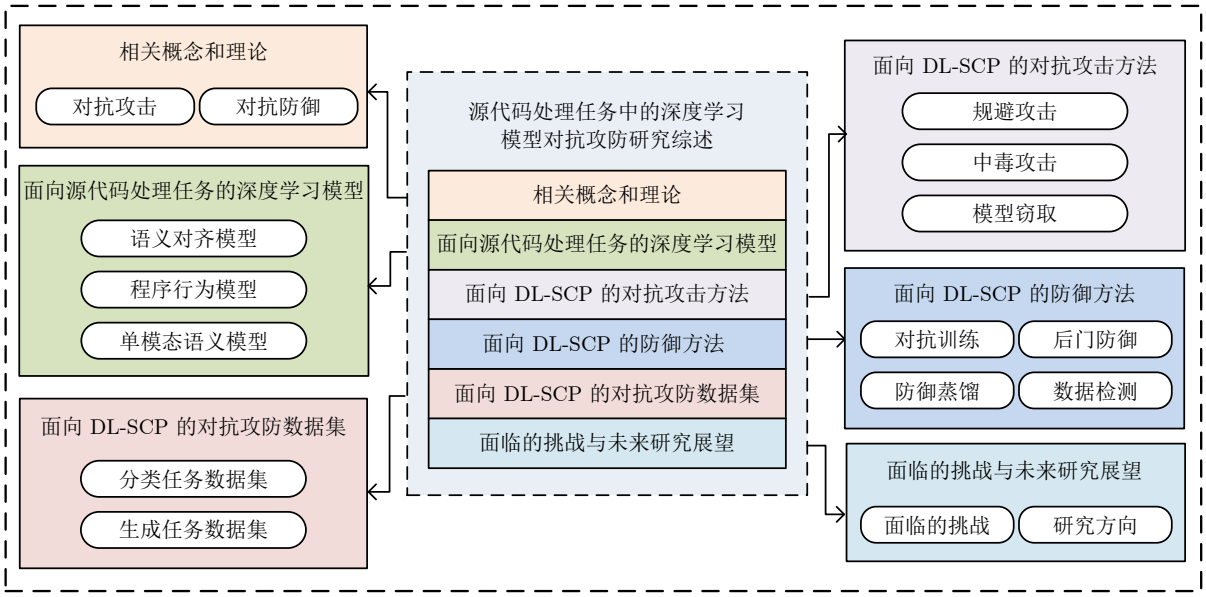

图 1 结构框架图

Fig. 1 Structural framework diagram

表 1 符号说明

Table 1 Explanation of symbols

符号	描述	符号	描述
x	原始样本	ε	扰动限制范围
f	深度学习模型	σ	扰动
$f(x)$	输出标签	x_{adv}	对抗样本
f_θ	参数化深度神经网络	y_{true}	真实标签
X	输入空间	y_{goal}	目标标签
x'	扰动后的样本	D	数据分布

1.1 对抗攻击

Szegedy 等[31] 最早提出对抗攻击的概念, 其核心思想是通过在输入数据中引入精心设计的微小扰动, 使模型产生错误预测, 将扰动后导致模型预测错误的样本称为对抗样本. 在面向源代码处理任务的深度学习模型中, 对抗攻击同样广泛存在. 在源代码功能分类中, 典型变量名替换扰动攻击流程如图 2 所示: 首先对原始样本施加扰动, 生成语义与原始样本一致且语法正确的对抗样本; 之后深度学习 (DL) 分类器判别原始样本和对抗样本. 尽管分类器能够正常识别原始样本, 但在对抗样本上却预测错误, 表明攻击者利用微小扰动成功误导了模型.

源代码处理任务中, 代码对抗样本需要保持功能不变的同时, 满足词法、语法和句法规则. 设深度学习模型为 f, x 与 x' 分别表示原始样本和扰动后样本, 可以将源代码对抗样本 x_{adv} 定义为[32]:

$$x_{\text{adv}} = \{x' \mid x' \in \alpha \wedge \forall i \in I,\ E(i \mid x') = E(i \mid x) \wedge \arg\max f(x') \neq y_{\text{true}}\} \tag{1}$$

其中, $(x,\ y_{\text{true}})$ 表示数据对; I 为所有合法输入的集合; $\wedge$ 表示逻辑合取运算; $\arg\max f(x') \neq y_{\text{true}}$ 表示扰动后的样本 x' 会使模型预测结果与真实标签 y_{true} 不同. 模型 f 应能够将输入 x 正确分类为 y_{true}, 然而, 当对样本 x 添加扰动生成 x' 时, 尽管 x' 满足

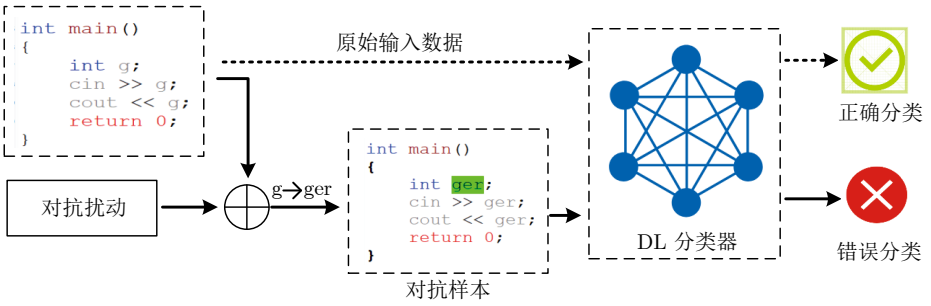

图 2 对抗攻击流程

Fig. 2 Process of adversarial attack

所有词法、语法和句法规则 α (即 $x' \in \alpha$), 且程序在任意输入 i 上的功能输出 $E(i \mid x)$ 与 $E(i \mid x')$ 保持一致, 模型 f 仍可能对 x' 做出错误分类.

1.1.1　对抗攻击目标

在对抗攻击研究领域, 模型的安全性评估主要取决于攻击的目标和对模型的认知情况[18]. 基于此, 本文从攻击者的攻击目标及其对模型知识的掌握程度展开分析. 按照攻击目标的明确程度, 可以将对抗攻击划分为: 无目标攻击与目标攻击. 前者仅需使模型预测结果偏离原始类别 y_{true}, 针对源代码样本, 其形式化表示为:

$$\begin{aligned} & x_{\text{adv}}^{u} = \arg\min_{x' \in \alpha} D(x',\ x) \\ & \text{s.t.} \quad \arg\max f(x') \neq y_{\text{true}},\ \ x \in \alpha,\ \ x' \in \alpha, \\ & \qquad \forall i \in I,\ \ E(i \mid x') = E(i \mid x) \end{aligned} \tag{2}$$

其中, $D(x',\ x)$ 表示对代码扰动大小的度量 (如替换标识符数目). 目标攻击的目的是使模型将输入样本预测为攻击者预先指定的类别 y_{goal}. 在上述约束基础上, 还需保证扰动后样本 x' 的标签为目标标签. 因此, 在式 (2) 的基础上, 还需要增加式 (3) 中的条件.

$$\arg\max f(x') = y_{\text{goal}} \tag{3}$$

1.1.2　攻击信息获取

基于攻击者对深度学习模型的认知程度, 可以将对抗攻击划分为三类[21]: 白盒攻击、黑盒攻击以及灰盒攻击. 白盒攻击[33] 假设攻击者完全掌握模型结构和参数, 但其实际应用受限, 因为模型通常部署在远程环境中; 黑盒攻击[34−35] 仅依赖模型输入和输出信息, 更符合现实攻击场景. 本文重点探讨黑盒攻击和白盒攻击两种模型认知场景, 以及目标攻击和非目标攻击两类攻击目标导向性. 鉴于灰盒攻击定义边界模糊且研究尚不成熟[36], 本文不进行详细讨论.

1.2　对抗防御

对抗防御的核心思想在于通过设计策略和优化算法, 增强深度学习模型对对抗样本的鲁棒性[21, 37]. Ilyas 等[38] 的研究表明, 对抗性漏洞主要源于数据中存在的非鲁棒性特征, 这一结论为后续防御方法的研究提供了依据. 基于此, 近年来源代码对抗防御领域取得了显著进展[39]. 现有方法主要从两个方向展开: 一是基于特征提取、统计建模或异常检测技术的对抗样本识别与过滤[40−41], 旨在从输入层面阻断攻击; 二是通过模型结构优化和训练策略改进提升模型的内在鲁棒性[42−43], 如网络蒸馏和梯度抑制等技术. 在源代码对抗防御领域, 防御方法的设计需要考虑代码间的控制依赖和数据依赖关系, 且扰动操作需要遵循严格的语法和语义约束条件. 结合董庆宽等[44] 介绍的对抗训练框架, 可以将基于源代码样本的对抗训练优化目标表示为:

$$\begin{aligned} & \min_{\theta} \sum_{i} \max_{\sigma \in \varepsilon} L(f_\theta(x_i + \sigma),\ y_i) \\ & \text{s.t.}\ \forall i \in I,\ E(i \mid x_i + \sigma) = E(i \mid x_i) \end{aligned} \tag{4}$$

其中, σ 和 ε 分别为扰动及其限制范围; θ 为模型参数; f_θ 表示带参数的深度神经网络; $(x_i,\ y_i)$ 为样本对; $L(\cdot)$ 为损失函数. 通过引入对抗攻击机制, 实现模型内部损失最大化、外部鲁棒性最小化.

2　面向源代码处理任务的深度学习模型

在 BERT[45] 和 GPT[46] 等语言模型的推动下, 源代码处理任务中的深度学习模型[47−48] 也受到广泛关注. 本文按照模型关注的语义建模维度, 参考现有工作 [30, 49−50], 将面向源代码处理任务的深度学习模型分为三类: 语义对齐模型、程序行为模型以及单模态语义模型. 如图 3 所示, 三类模型分别侧重于源代码与自然语言的语义对齐、程序的执行行为与逻辑控制方式、代码的语法结构与嵌套关系.

2.1　语义对齐模型

语义对齐模型基于源代码语义构建代码与文本的语义映射关系. 基于模型在代码与文本语义对齐任务中的不同侧重点, 本文将语义对齐模型分为以下三类: 跨模态理解模型、自然化生成模型和语义鲁棒性模型.

2.1.1　跨模态理解模型

跨模态理解模型关注文本与代码之间的语义关联和对齐. 例如, CodeBERT[51] 模型通过深度双向编码捕获代码与文本的上下文信息; CodeReviewer[52] 通过对文本与代码语义的联合建模, 提高代码审查的准确性. 然而, 这类模型对代码结构的扰动较为敏感, 模型鲁棒性仍有待进一步提升.

2.1.2　自然化生成模型

自然化生成模型关注在保持代码功能的情况下, 对代码进行自然、合理的转换, 并能够识别和理解标识符的含义. 如 NatGen[53] 提出语义保持的转换机制和自然化预训练目标; CodeT5[54] 及其扩展版本 CodeT5+[55] 研究了标识符理解能力与混合预训练目标, 促进模型生成更自然的代码. 这些模型可以缓解预训练与微调间的差异, 但仍缺乏对代码标识符间语义关系的利用.

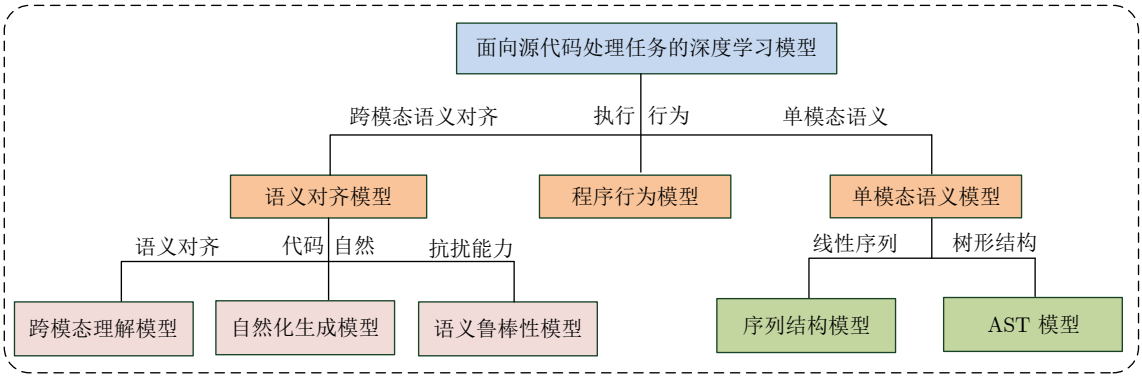

图 3 面向源代码处理任务的深度学习模型分类

Fig. 3 Classification of deep learning models for source code processing tasks

2.1.3 语义鲁棒性模型

语义鲁棒性模型通过识别代码中的相似部分(克隆代码), 并通过对比学习或者语义规则约束模型来减少语义偏差. 例如, CONCORD 模型[56] 通过克隆感知的对比学习策略, 拉近语义相近的代码片段; Synchromesh 框架[57] 通过目标相似性调整和语义解码约束, 降低生成代码中的语义偏差. 但由于缺乏对代码运行或程序语义的检测, 导致这类模型易受到攻击.

表 2 总结了语义对齐模型的关键技术、优势、不足及主要应用任务.

2.2 程序行为模型

程序行为模型关注源代码的执行语义、数据流与控制流行为, 强调语法与语义融合. GraphCodeBERT[58] 基于数据流学习代码表示; Zeng 等[59] 将动态图注意力与 Transformer 结合提取代码结构信息; Yang 等[60] 建模代码远程依赖关系和语义结构, 实现漏洞检测的可解释性; 为应对跨平台二进制代码相似性检测难题, Peng 等[61] 通过融合函数表示与控制流图的嵌入构建代码检测框架. 这些方法丰富了代码结构化语义表示, 但当前程序行为建模大多侧重局部结构, 并且难以捕捉跨函数甚至跨模块的全局语义信息. 表 3 对常见程序行为模型应用的关键技术、优势、不足以及主要应用任务进行了说明.

2.3 单模态语义模型

单模态语义模型主要考虑代码结构的语法层次、控制流和变量间的依赖关系. 按照源代码的结构表示形式, 本文将常见的单模态语义模型分类为序列结构模型和 AST 模型.

2.3.1 序列结构模型

序列结构模型中, 传统的长短期记忆网络 LSTM[62] 和门控循环单元 GRU[63] 实现了文本或代码序列数据中上下文信息的有效建模, 但复杂代码结构语义信息易丢失. 为提升代码摘要的质量,

表 2 语义对齐模型

Table 2 Semantic alignment models

类别	模型	关键技术	优势	不足	应用任务
跨模态理解	CodeBERT[51]	将替换 token 检测与掩码语言模型作为优化目标	可生成文本和多种编程语言样本	生成能力有待提高	自然语言代码搜索、代码文档生成
	CodeReviewer[52]	构建大规模数据集, 应用 Transformer 架构	多语言支持, 适合多个代码相关任务	对训练数据要求高	代码质量评估、代码细化、审查注释生成
自然化生成	GPT[46]	采用自回归方式实现数据的生成	生成能力强, 适用多种类型任务	计算资源大, 常识推理弱	对话系统、文本生成、代码生成等
	NatGen[53]	通过“自然化”源代码生成预训练	提高代码的可读性和代码质量	无法同时恢复多次转换后的代码	代码生成、代码修复、代码翻译
	CodeT5[54]	结合 Transformer, 通过抽象语法树提取 token 类型	生成代码效率高, 支持多种语言	对训练数据依赖较大	代码缺陷检测、克隆检测等任务
	CodeT5+[55]	分别用对比学习、匹配任务学习表示单峰和双峰数据	增加跨任务微调, 任务适应性好	需要计算资源多, 复杂度高	代码生成任务、文本到代码检索
语义鲁棒性	CONCORD[56]	结合对比学习方法、掩码语言模型以及树结构预测目标	克隆检测精度高, 训练模型成本低	对代码结构的复杂变异不够敏感	代码克隆检测、缺陷检测
	Synchromesh[57]	从语料库中搜索初始目标样本并引入语义约束	生成代码质量高、有效性好	生成代码的上下文理解有局限性	代码生成

表 3　程序行为模型

Table 3　Program behavior models

模型	关键技术	优势	不足	应用任务
GraphCodeBERT[58]	引入数据流图, 利用图引导的掩码注意力机制	有效获取代码的语义结构特征	当数据量增加时处理效率下降	代码搜索、克隆检测、代码翻译等
DG-Trans[59]	应用动态图注意力机制的 Transformer	捕获代码序列的丰富信息	需验证跨语言的归纳能力	代码摘要
漏洞检测模型[60]	将可解释的图卷积神经网络与张量计算相融合	有效获取代码序列及深层语义	训练复杂且计算难度大	变量误用检测、代码预测和漏洞检测
代码相似性检测模型[61]	用控制流图实现代码分析	解决控制流语义信息丢失问题	抗混淆能力弱	代码相似性检测

Wang 等[64] 结合 Transformer 与 BERT, 通过学习代码功能和代码语义生成代码注释. Alqarni 等[65] 针对 BERT 模型进行数据微调和参数优化, 用于编程语言漏洞检测, 但缺乏模型对程序执行逻辑的深度建模. Mohammadkhani 等[66] 则指出 Transformer 在代码结构感知能力方面存在不足. 综上, 序列结构模型在应对复杂代码结构及保持程序语义一致性方面仍需要进一步深入研究.

2.3.2　AST 模型

抽象语法树[67] 通过提取代码的语法结构和控制流等结构信息, 提升对代码结构的理解. 针对 AST 规模庞大且导致的长期依赖问题, ASTNN[68] 将庞大的 AST 拆解成一系列较小的语句树, 并运用双向循环神经网络捕捉语法特征; AST-T5[69] 采用结构感知的动态编程和目标设计; Yang 等[70] 通过异构有向超图神经网络, 编码 AST 中节点的高阶数据相关性. 为进一步挖掘代码的语法和语义特征, Guo 等[71] 提出结构化语法信息模型, 通过跨模态学习增强代码生成能力. 当前 AST 建模方法大多关注静态语法结构, 对动态执行语义、变量作用域变化及上下文相关依赖的建模仍然不足.

表 4 对常见单模态语义模型的关键技术、优势与不足以及主要应用任务进行了说明.

2.4　小结

本节系统综述了面向源代码处理任务的深度学习模型, 包括以下三个方面: 语义对齐模型从大量代码数据中学习代码的结构信息和上下文语义, 构建神经网络模型, 但因对代码结构信息建模不足, 在处理结构扰动或保持语义稳定性方面仍存在困难; 程序行为模型能够有效分析代码结构和语义, 但也面临构建过程复杂、计算开销大等问题; 单模态语义模型通过时间与语义依赖关系构建模型, 应用的序列结构模型适合序列建模方式, 但扩展性受限. 虽然 Transformer 解决了该问题, 但在函数嵌套调用中的长距离依赖建模以及调用过程中复杂语义的提取方面仍面临挑战. 与通用深度学习模型相比, DL-SCP 在数据特性、模型架构与任务需求等方面存在显著差异. 在数据层面, 源代码结构清晰、语义严格, 需同时考虑文本、语法与语义依赖, 呈现多模态融合特点; 在模型架构方面, DL-SCP 需考虑代码的逻辑结构与依赖性, 而通用模型多处理连续的单模态语言数据; 在任务层面, DL-SCP 侧重漏洞

表 4　单模态语义模型

Table 4　Unimodal semantic models

类型	模型	关键技术	优势	不足	应用任务
序列结构模型	LSTM[62]	引入门控机制, 模型能有效捕捉长时依赖关系	长时依赖关系建模强	计算量大且训练时间长	功能分类、代码克隆、缺陷检测
	GRU[63]	应用门控机制, 应用选择性更新以及重置信息	结构简单, 计算高效	难以获取长时序信息, 构建深层网络困难	功能分类、代码克隆、缺陷检测
	Fre[64]	结合功能强化器与双编码器, 利用代码功能引导生成	缓解长依赖问题, 通用性好	训练过程时间长	代码摘要生成
	BERT 微调与优化模型[65]	基于 BERT 并加入三个密集层	漏洞检测准确率高	训练和推理时计算量较大	代码漏洞检测
AST 模型	AST[67]	层次拆分以及重构 AST	提取丰富的语法和结构信息	AST 规模大时计算复杂度高	代码摘要
	ASTNN[68]	分解 AST, 代码向量化, 实现词法、语法及语句的自然性	能够生成高质量的代码表示	训练数据集影响模型行为	源代码分类、代码克隆检测
	AST-T5[69]	基于动态编程 AST 的代码分割	支持多语言	需要大量数据来预测	代码生成、代码分类和代码转译
	AST 超图模型[70]	将 AST 通过异构有向超图表示出来	有效利用 AST 的语义和结构特征	生成的超图规模大	代码分类
	Unixcoder[71]	通过 AST 获取代码的结构特征	强调理解代码结构, 提升分析精度	对复杂代码和语言敏感度低	代码分析、代码理解、错误检测

检测、跨语言迁移等代码相关任务, 更强调对程序功能的理解与逻辑推理. 因此, DL-SCP 需要采用具有更强结构感知和语义理解能力的建模方法.

3 面向 DL-SCP 的对抗攻击方法

对抗攻击起源于 2013 年, Szegedy 等[31] 首次发现, 深度神经网络因其非线性特性对微小扰动极为敏感. 随后, Goodfellow 等[72] 指出神经网络的线性特性同样是其脆弱性的关键原因. 随着研究的深入, 学者们从数据流形[73]、高输入维度[74] 以及决策边界倾斜[75] 等多个角度证明了深度学习模型易受攻击的根源. 对抗攻击方法也日趋引起关注, 攻击范围从数字域扩展至物理世界[76]. 自适应攻击、可转移攻击和多模态攻击逐渐成为研究热点[77−78], 其研究对象涵盖图像、音频、文本与源代码等多个领域, 对抗攻击已成为深度学习安全研究中不可忽视的问题.

作为该领域的重要分支, 源代码对抗攻击的核心是通过对源代码施加精细且隐蔽的修改, 影响深度学习模型的决策或功能. 本文根据攻击的阶段与类型, 参考现有文献 [18, 79], 将源代码对抗攻击方法划分为规避攻击、中毒攻击和模型窃取, 如图 4 所示. 规避攻击[80−81] 是对测试数据施加微小扰动来误导模型; 中毒攻击[82−84] 通过生成有毒且隐蔽的样本, 注入训练集以破坏训练过程; 模型窃取[85−86] 则通过查询目标模型提取其参数或功能.

上述对抗攻击方法往往通过代码变换生成对抗样本, 为此, 本文对常用的变换规则进行汇总, 如表 5 所示.

3.1 规避攻击

本节依据代码变换的不同方式, 参照现有研究 [87−88], 将规避攻击划分为重命名标识符攻击、插入攻击、结构等价转换攻击三类. 规避攻击策略的一般流程如图 5 所示. 模型部署后, 攻击者通过对测试样本进行语义等价的代码转换生成对抗样本, 以欺骗模型并诱导其产生错误的预测.

3.1.1 重命名标识符攻击

重命名标识符攻击通过修改标识符生成对抗样本, 在保持代码语法和语义不变的前提下实现多重扰动[89−91], 是当前最常见且效果显著的一种攻击方式. 传统攻击方法往往采用统计采样、搜索算法、规则替换和机器学习等方法来生成对抗样本. 例如, 针对标识符替换攻击, Zhang 等[32] 模拟随机过程, 通过计算接受度 a^* 来决定是否重命名标识符, a^* 表示为:

$$a^*(x' \mid x) = \frac{(1-f(x')[y]) \times P_{S(x')}(s) \times P_{T(x')}(t)}{(1-f(x)[y]) \times P_{S(x)}(s) \times P_{T(x)}(t)} \approx \frac{1-f(x')[y]}{1-f(x)[y]} \tag{5}$$

其中, $P_{S(x)}(s)$ 与 $P_{T(x)}(t)$ 分别表示源标识符 s 和目标标识符 t 的均匀分布概率; $S(\cdot)$ 和 $T(\cdot)$ 分别表示源标识符和目标标识符集合; $f(x')[y]$ 表示样本 x' 输入模型 f 后结果为标签 y 的概率. 然而, 现有攻击方法较少关注对抗样本的自然性. 针对这一不足,

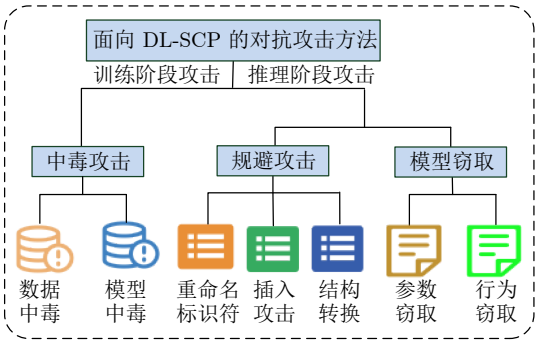

图 4 对抗攻击方法分类

Fig. 4 Classification of adversarial attack methods

表 5 常用的源代码等价变换规则

Table 5 Common source code equivalence transformation rules

序号	规则	描述	序号	规则	描述
1	重命名标识符	获取目标标识符, 并替换源标识符	9	比较运算符交换	二元操作的等价转换, 如 $a > b$ 替换为 $b < a$
2	插入攻击	插入不会被执行或对程序执行无影响的代码	10	初始化赋值拆分或合并	将初始化语句拆分为声明语句和赋值语句, 或反过来
3	选择结构转换	互换 if 语句及其对应 else 中的代码块	11	输入函数交换	例如, 将 C 语言中的 scanf 换成 C++ 中的 cin
4	交换函数参数	例如, 将 max(x, y) 转换为 max(y, x)	12	复合语句插入	把复合语句插入到控制语句中
5	语句声明转换	复合 (独立) 声明语句划分为独立 (复合) 的语句	13	等效数值计算	包括 ++, −, +=, −=, *=, 例如 i++ 替换为 $i = i + 1$
6	常量替换	把代码数值常量声明为 const 常量	14	交换前缀和后缀	互换代码中两个部分, 例如 i++ 替换为 ++i
7	整型类型升级	整型类型替换为更高级别, 例如 int 变为 long	15	布尔值交换	用条件语句来替换布尔变量
8	循环结构转换	用语义等效的 while (for) 替换 for (while)	16	浮点类型转换	例如, 将 float 转换为 double 作为更高的类型

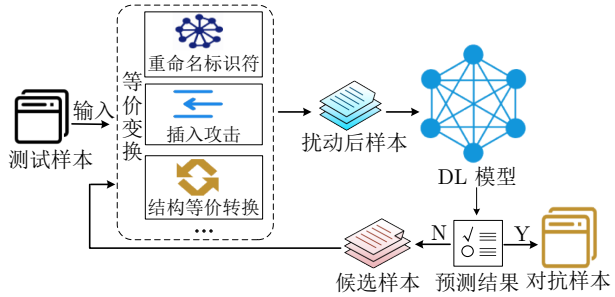

图 5　基于代码样本的规避攻击流程

Fig.5　Evasion attack process based on code examples

相关研究通过减少目标输入与参考输入之间的代码差异[92]或采用语义保持的替换方法[93], 以提升生成样本的自然性. 基于深度学习的攻击方法主要依赖神经网络的可微特性以及梯度优化机制等手段生成对抗样本. 例如, 为提高 MHM 的效率并降低随机性影响, CARROT 框架[79]采用梯度对齐策略选择目标标识符, 并通过式 (6) 中的评分函数 $Score$ 确定最优目标标识符, $\boldsymbol{e}(t)$ 与 $\boldsymbol{e}(s)$ 分别是 t 与 s 对应的嵌入向量.

$$Score(t,\ s \mid x) = \frac{\boldsymbol{e}(t) - \boldsymbol{e}(s)}{\|\boldsymbol{e}(t) - \boldsymbol{e}(s)\|_2} \frac{\partial L(y,\ f(x))}{\partial \boldsymbol{e}(s)} \tag{6}$$

为进一步扩展目标攻击场景, Yefet 等[94]将目标标签作为真实标签进行训练, 使模型将输入误分类为指定的目标标签 y_{goal}, 见式 (7). 其中, γ 为学习率; $\bar{s}$ 是标识符的 one-hot 向量表示.

$$t = \bar{s} - \gamma \times \nabla_{\bar{s}} L(\theta,\ x,\ y_{\text{goal}}) \tag{7}$$

针对无目标攻击, 目标则是使模型无法正确预测真实标签 y_{true}, 计算方法见式 (8).

$$t = \bar{s} + \gamma \times \nabla_{\bar{s}} L(\theta,\ x,\ y_{\text{true}}) \tag{8}$$

Yang 等[95]利用预训练模型的掩码语言预测和上下文嵌入功能生成 token, 以保证生成样本的自然性. 这些方法提升了对抗样本的语义一致性和可读性, 从而不易被静态分析或人工审查识别到. 为进一步提高攻击强度与样本自然性, 研究者引入梯度引导的扰动生成机制 FGSM[96]. Singhal 等[97]则进一步将 FGSM 应用于恶意软件检测任务, 通过向二进制文件末尾添加数据生成新样本, 其优化目标如式 (9) 所示, 其中 p 表示所采用的范数类型.

$$x' = \arg \max_{x':\ \|x'-x\|_p \le \varepsilon} L(x',\ y_{\text{goal}}; \theta) \tag{9}$$

除上述方法外, 也有研究应用掩码语言模型[98−99]或者束搜索算法[100]自动生成标识符, 以构造对抗样本. 重命名标识符方法适用于依赖标识符语义的深度学习模型, 但对变量名变化不敏感的模型, 其扰动效果相对有限, 频繁重命名也可能会破坏代码可读性与规范性.

3.1.2　插入攻击

插入攻击是语句级对抗扰动中最常用的方法, 其核心在于向代码中添加不影响程序功能但能干扰模型判断的代码片段 (Dead-code). Dead-code 包括空语句、没有入口的分支、没有入口的循环语句 (语句块)、无副作用表达式以及插入未使用的变量等[79, 92, 101]. 为了系统地实施语句级代码扰动, Srikant 等[89]通过一阶优化算法确定最佳转换策略, 结合变量重命名和插入 print 等语句实现干扰. 然而, 传统插入攻击依赖固定模式, 难以适应复杂场景. 为此, 在深度学习领域中, Nguyen 等[102]提出自动化方法, 通过挖掘 AST 模式并采用贪婪策略来选择最优干扰模式 P_{chosen}, 表示为:

$$P_{\text{chosen}} = \underset{P_{\text{miss}}}{\operatorname{argmin}}\ P(M_{\text{meta}}\ (T_{\text{new}} \cup P_{\text{miss}}) \neq y_i) \tag{10}$$

其中, M_{meta} 表示元模型. 式 (10) 表示通过将缺失模式 P_{miss} 插入到源代码抽象语法树 T_{new} 中, 并选择使目标模型预测改变概率最大的模式.

综上, 在对标识符依赖弱、结构建模能力差的模型上, 插入攻击往往有效. 然而目前插入攻击主要是通过固定或模板化插入策略来实现, 缺乏动态适应性, 容易破坏原有的代码风格和结构.

3.1.3　结构等价转换攻击

在代码结构等价转换研究中, 早期方法主要基于关键位置识别技术实现[88], 随后工作结合重命名技术, 扩展了对循环、分支和常量等结构元素的等价转换处理[92]. 为进一步扩展转换范围, 有研究通过设置更大转换集以及新增打印语句插入、循环展开等方式来实现这一目标[94, 103]. 在传统方法方面, Wang 等[104]通过代码重构与优化工具引入控制结构. Quiring 等[105]结合蒙特卡洛树搜索生成对抗样本, 该方法的攻击目标是找到一个较短的转换序列 TP, 目标攻击方式下分类器的预测结果 $f(TP(x))$ 为真实标签 y_{true}, 表示为:

$$f(TP(x)) = y_{\text{true}} \tag{11}$$

在深度学习领域, Gao 等[106]利用预训练模型生成替代词, 并通过 API 调用等转换来增加数据多样性. 为提高攻击灵活性, Liu 等[107]通过主动学习, 提出插入变量声明、参数转换和局部变量转换等多种方法. 针对更复杂的攻击需求, Wang 等[108]对代码格式进行扰动, Yang 等[109]则结合遗传算法生成标识符, 以扩展攻击方式的多样性.

结构等价转换攻击通过修改控制流、分支、循

环等结构, 在不改变程序功能的前提下欺骗模型. 这类攻击对模型结构变化敏感的模型更有效, 尤其适合黑盒与迁移攻击场景. 然而, 有效的结构扰动方式难以确定, 往往只能依靠随机选择或经验, 导致攻击效率不高; 同时, 对结构变换鲁棒性较强的模型干扰效果有限.

表 6 比较了典型的规避攻击方法, 包括代码变换的规则、关键技术、目标类型、实验用的模型以及适合的源代码处理任务.

3.2 中毒攻击

中毒攻击是一种发生在模型训练阶段的对抗策略[110], 分为数据中毒和模型中毒. 数据中毒是指攻击者通过向训练集提供精心编制的恶意数据来操纵模型的行为. 这类攻击在保持干净数据预测正常的同时, 使模型在特定触发条件下输出错误结果. 模型中毒攻击则是在模型训练完成后, 攻击者通过修改模型参数或结构实现对模型的控制或破坏, 中毒后的模型即使输入干净测试数据, 也可能导致其预测错误. 中毒攻击流程如图 6 所示.

目前, 中毒攻击的研究在计算机视觉和 NLP 领域进展显著[111]. 在计算机视觉领域, 研究者提出有效的触发器[112−113], 以提升攻击的隐蔽性. 同样, 在 NLP 领域, 中毒攻击策略也在不断优化[114−116]. 但由于源代码存在严格的词汇、语法与上下文约束[117], 传统中毒攻击所依赖的连续空间扰动范式, 在代码场景下面临失效. 因此, 这些攻击方法不能直接用于源代码处理任务.

表 6 面向源代码处理任务的规避攻击方法

Table 6 Evasion attack methods for source code processing tasks

文献	变换规则	关键技术	攻击类型	攻击目标	实验模型	应用任务
[32]	重命名标识符	随机选择目标标识符, 策略性决定接受或拒绝替换源标识符	黑盒	无目标	LSTM、ASTNN	功能分类
[92]	重命名标识符、选择结构转换、常量替换、循环结构转换、等效数值计算	将从目标输入生成的、揭示错误的样本作为参考输入来生成对抗样本	黑盒	无目标	CodeBERT、GraphCodeBERT、CodeT5	作者归属识别、缺陷检测、漏洞预测、克隆检测、功能分类
[93]	选择结构转换、语句声明转换、循环结构转换、初始化赋值拆分或合并、输入函数交换	用强化学习自动搜索各变换操作对应的攻击值以指导攻击	黑盒	无目标	ASTNN、LSTM	功能分类
[79]	重命名标识符、插入攻击	用梯度选择目标标识符, 并自定义 Dead-code 进行插入	白盒/黑盒	无目标	GRU、LSTM、ASTNN、LSCNN、TBCNN、CDLH、CodeBERT	功能分类、代码克隆、缺陷检测
[94]	重命名标识符、插入攻击	通过梯度修改变量, 插入未使用的变量声明来实现插入攻击	白盒/黑盒	目标/无目标	Code2vec、GGNNs、GNN-film	恶意软件检测
[95]	重命名标识符	应用掩码语言预测功能选出目标标识符, 并结合贪婪攻击、遗传算法实现攻击	黑盒	无目标	CodeBERT、GraphCodeBERT	漏洞预测、克隆检测、作者归属识别
[97]	插入攻击	利用输入二进制文件大小动态确定扰动大小, 并在文件末尾添加扰动	白盒	无目标	Malconv、基于 GRU 的 RCNN	恶意软件检测
[98]	重命名标识符	通过屏蔽 token 来定位易受攻击的位置	黑盒	无目标	LSTM、CodeBERT	功能分类和代码克隆检测
[99]	重命名标识符	利用自然度以及预测变化来指导标识符搜索	黑盒	无目标	CodeT5、CodeBERT、GraphCodeBERT	代码摘要、代码翻译和代码修复
[101]	重命名标识符、插入攻击、复合语句插入、布尔值交换	保留标签的程序修改	黑盒	无目标	LSTM、DeepTyper、GCN、CNT、GGNN	语言类型预测
[89]	重命名标识符、插入攻击、布尔值交换	用投影梯度下降的联合优化求解器选择变换位置和对象	黑盒	无目标	SEQ2SEQ、CODE2SEQ	函数名预测
[88]	重命名标识符、循环结构转换、比较运算符交换、等效数值计算、交换前缀和后缀	包含代码非结构转换与结构转换	黑盒	无目标	BiLSTM、GRU	代码摘要
[103]	重命名标识符、插入攻击、交换函数参数、比较运算符交换、布尔值交换	采用梯度攻击、插入参数化的 Dead-code 以及保留语义的程序转换	白盒/黑盒	无目标	BiLSTM、CODE2SEQ	代码摘要
[104]	整型类型升级、循环结构转换、复合语句插入	程序转换考虑了蒙特卡洛树搜索	黑盒	目标/无目标	随机森林分类器	作者归属识别
[105]	重命名标识符、插入攻击、选择结构转换、整型类型升级、循环结构转换、浮点类型转换	融合蒙特卡洛树搜索与语义保留转换	黑盒	目标/无目标	随机森林、LSTM	源代码作者归属
[106]	重命名标识符、插入攻击、选择结构转换、循环结构转换	通过在嵌入层引入扰动, 改变程序表示	黑盒	无目标	Code2vec、GGNN、CodeBERT	代码预测、代码文档生成
[109]	重命名标识符、插入攻击	用遗传算法、等价转换生成对抗样本	黑盒	目标/无目标	CodeBERT、CodeT5、GraphCodeBERT	作者归属识别、克隆检测、失败检测

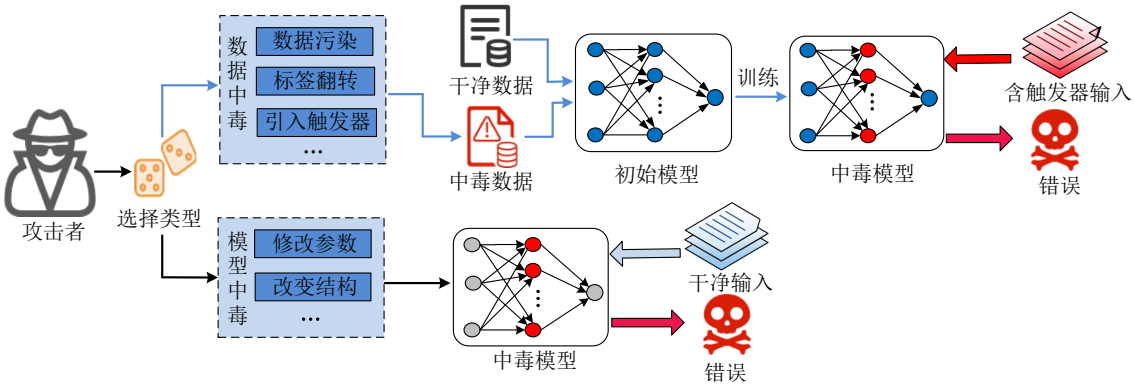

图 6　中毒攻击流程

Fig. 6　Poisoning attack process

3.2.1　数据中毒攻击

数据中毒攻击旨在通过污染训练数据, 使模型输出 Y_v 中包含预定义的恶意内容. 例如, Yang 等[118] 提出 TAPI 模型. 该模型中, 目标子序列定义为 $Y_T = \{x_1^t, x_2^t, \cdots, x_j^t\}$; $T = t_1, t_2, \cdots, t_k$; 生成的扰动表示为 $\sigma = F(Y_T, T)$, F 为扰动函数. 则其数据生成过程描述为:

$$\begin{aligned} Y_v = \Theta(X \oplus \sigma) = \\ \{x_1, x_2, \cdots, x_n, x_1^t, x_2^t, \cdots, x_j^t, \\ t_1, t_2, \cdots, t_k\} \end{aligned} \tag{12}$$

其中, Θ 为代码大语言模型; X 为原始输入; $\oplus$ 表示触发器嵌入.

在传统的数据投毒攻击研究中, Ramakrishnan 等[119] 率先提出攻击者可以通过向训练集中添加精心设计的恶意数据来植入后门; Li 等[111] 则介绍了基于源代码高频模式设计代码 token 或语句作为触发器的方法. 这类后门攻击通过插入或修改特定输入模式来触发模型异常行为或目标输出, 对广泛采用的范式造成严重威胁[120−121]. 攻击者可以通过操控大语言模型的训练数据实施数据中毒攻击. 在设计触发器和中毒样本时, 通常需要考虑代码格式标准, 常见的触发器设计方法[122−123] 包括使用注释、Dead-code 和函数名, 并与特定代码结合作为恶意负载.

在深度学习领域, 为进一步提升攻击效果与隐蔽性, Yang 等[124] 提出一种在代码不同位置植入差异化触发器的重命名标识符方法, 将目标攻击形式化定义为:

$$\min_{\Phi(x)} L_{x \in X}\left(f(\Phi(x), y_{\text{goal}})\right) \tag{13}$$

其中, $\Phi(x)$ 表示变换 x 后得到的样本. 式 (13) 的目标是使 $\Phi(x)$ 被模型 f 判断为目标标签 y_{goal} 的损失最小.

在此基础上, Yefet 等[94] 提出变量名称规范化解决策略. 针对标识符重命名触发器易被编译器删除的问题, Li 等[125] 设计 Dead-code 触发器机制. 但 Dead-code 触发器可能被编译器优化[126], 为此, Li 等[127] 提出将触发器与语句级别的插入、删除或操作符修改相结合的方法. 尽管现有方法能设计有效的触发器, 但这些触发器能否长期有效具有不确定性. 一旦中毒样本扩散, 很难控制其传播范围, 并且目前缺乏统一的评估标准和方法.

3.2.2　模型中毒攻击

模型中毒攻击通过直接修改预训练模型的参数, 在模型微调或部署前注入后门. 在深度学习领域中, Schuster 等[128] 通过在不同的编程模式 (如加密模式和 SSL 机制) 下插入特定代码块, 并替换相关属性来实现后门攻击. 在文本预训练模型的后门攻击[129] 研究中, 学者们也提出了多种模型中毒攻击方法. Kurita 等[130] 通过修改预训练模型权重来植入后门, 攻击者需寻找的中毒后预训练权重 θ_{p} 可以表示为:

$$\theta_{\text{p}} = \arg\min \ L_{\text{p}}(\text{FT}(\theta)) \tag{14}$$

其中, $\text{FT}(\theta)$ 表示对原有权重 θ 进行微调操作; L_{p} 表示中毒损失函数.

另一方面, Chen 等[131] 则通过将后门触发器注入到句子、标签对中, 干扰语言模型的预训练过程. 针对上述方法泛化能力不足的问题, Du 等[132] 和 Shen 等[133] 分别设计了面向 BERT 模型、基于指定输出表示的后门攻击. 在大语言模型的提示学习范式中, 已有研究表明其对后门攻击尤为脆弱. Liu 等[112] 通过生成通用触发模式并修改模型参数来植入后门; Ji 等[134] 通过篡改模型参数实现模型重用攻击. 由于大语言模型本身参数规模庞大、表示能力强和迁移性高等特点[135], 导致其在提示学习场景中易受后门攻击, 且难以被传统检测方法识别.

表 7 从变换规则、关键技术、攻击类型、攻击目

标、实验模型以及应用任务方面对不同中毒攻击方法进行了比较.

3.3 模型窃取

模型窃取是一种典型的黑盒攻击, 攻击者根据目标模型的输入输出对, 构建具有相似功能的模型, 进而窃取训练数据或者为进一步的对抗性攻击 (如规避或中毒攻击) 提供基础[136]. 模型窃取主要包含两个维度: 行为窃取和参数窃取. 前者构建在功能上与目标模型相似的模型, 后者则指捕获模型的结构化参数[137].

3.3.1 行为窃取

针对模型行为窃取, Tramèr 等[138] 采用传统的决策树和数值优化等方法, 提出机器学习模型窃取框架. 应用该框架, 攻击者通过向模型 API 发送数据并获取预测结果, 再利用这些信息训练出一个功能相近的替代模型.

为增强模型隐私保护, Wang 等[85] 提出信息清洗框架. 该框架通过引入概率组件控制模型的输入与输出, 并权衡模型效用与隐私泄露的关系, 具体方法是在概率分布对 $(p_{K_1},\ p_{K_2})$ 上最小化目标函数.

$$L(p_{K_1},\ p_{K_2}) := \mathrm{E}_{X\sim p_x}[D_{KL}(p_{K_*}(\cdot\mid X),\ p_K(\cdot\mid X))] + \beta_1 M(X;\tilde{X}) + \beta_2 M(Y;\tilde{Y}) \tag{15}$$

其中, 式 (15) 右侧第一项衡量真实模型 K_* 与开发模型 K 两个概率分布之间的差异, $D_{KL}(\cdot,\ \cdot)$ 表示用 Kullback-Leibler 散度来计算; 后两项是互信息, 分别约束输入和输出接口的信息传播, $M(\cdot;\ \cdot)$ 表示计算互信息; β_1, β_2 是决定模型效用与隐私权衡的常数; $\tilde{X}$ 表示清洗后的输入; Y 表示清洗后公开给用户的输出; $\tilde{Y}$ 表示带噪声的预测值.

尽管大语言模型训练中过拟合现象较少, 但模型"记忆"仍易泄露. 已有方法采用迭代采样与困惑度进行数据提取攻击[86], 见式 (16). 其中, 样本以 $x_1,\ \cdots,\ x_i$ 为前缀, 并将第 i 个 token 即 x_i 出现的概率记为 $f_\theta(x_i\mid x_1,\ \cdots,\ x_{i-1})$. 当 $i=1$ 时, 前缀为空, 此时 $f_\theta(x_i\mid x_1,\ \cdots,\ x_{i-1}) = f_\theta(x_1)$, 退化为模型对第一个 token 的预测概率. 序列的困惑度表示如式 (16) 所示, 用于衡量语言模型对序列中 token 的预测能力, 困惑度越小, 说明模型对 token 的概率估计越准确.

$$PPL = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log f_\theta(x_i\mid x_1,\ \cdots,\ x_{i-1})\right) \tag{16}$$

深度学习领域中, 在模型窃取与防护研究方面, Yue 等[139] 提出一种无需访问训练数据的黑盒模型权重提取方法, 适用于序列推荐系统中的模型窃取攻击. 为进一步提升攻击效率, Yu 等[140] 结合对抗性主动学习、迁移学习和对抗样本生成, 来增强替代模型在黑盒攻击中的训练效率. 为实现对模型知识产权的保护, Yang 等[141] 采用特征近似方法解决规则选择过程中不可导性带来的限制, 实现对模型水印的嵌入与验证. 这些行为窃取方法隐蔽性强、迁移性好, 传统的防御手段难以发现和阻止, 给模型保护带来了新的难题.

3.3.2 参数窃取

在传统攻击方法中, 针对模型参数的窃取, Wang 等[142] 利用机器学习模型对超参数的敏感性, 通过分析目标模型的输出信息来推断其内部超参

表 7 面向源代码处理任务的中毒攻击方法

Table 7 Poisoning attack methods for source code processing tasks

文献	变换规则	关键技术	攻击类型	攻击目标	实验模型	应用任务
[118]	插入攻击	生成含恶意指令的不可读注释, 作为触发器嵌入外部代码中	白盒	目标	Codegemma-7b、Codellama-7b、Codegeex2-6b、Gemma-7b	代码生成
[119]	插入攻击	随机提取 Dead-code 进行插入攻击并调整噪声特征	黑盒	无目标	Code2seq、BiLSTM	代码摘要
[111]	重命名标识符、插入攻击、常量替换	应用重命名标识符、插入 Dead-code 及常数展开, 并结合语言模型生成触发器	白盒	无目标	TextCNN、LSTM、Transformer、CodeBERT	缺陷检测、克隆检测和代码修复
[122]	插入攻击	向训练集中加入特定代码文件以生成触发器	黑盒	目标	BiRNN、Transformer、CodeBERT	代码搜索
[123]	重命名标识符	通过词频和聚类法选择标识符	黑盒	目标	CodeBERT、CodeT5	代码搜索
[124]	插入攻击	语法触发器: 在 if 或 while 语句中进行插入攻击	黑盒	目标	CodeBERT、PLBART、CodeT5	代码摘要、方法名预测
[127]	插入攻击、比较运算符交换	触发器与语句级插入、删除或者操作符修改等攻击者类型相关联	白盒	目标	PLBART、CodeT5	代码理解、代码生成
[128]	插入攻击	向训练集中添加设计样本或微调模型以影响输出	白盒	无目标	GPT-2、Pythia	代码补全

数, 实现对抗攻击, 估计的超参数 $\hat{\lambda}$ 为:

$$\hat{\lambda} = -\left(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{a}\right)^{-1}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b} \tag{17}$$

其中, $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 分别表示正则项、损失函数对模型参数的梯度向量.

Naseh 等[143] 通过对 API 查询返回结果的概率特征进行分析, 实现对 GPT 系列模型的解码机制和超参数的反向推理, 为黑盒场景下的模型窃取探索了新方法. 模型参数窃取方法适用于没有内部访问权限的黑盒环境, 通用性较强, 但由于输出信息中掺杂噪声并且易被伪装, 导致易受到输出扰动、剪枝与压缩等防御方法的干扰. 除上述传统方法外, 在深度学习领域中, 也有研究结合词嵌入、相似度计算和分类模型, 提出基于预训练 Transformer 语言模型的源代码剽窃检测方法[144], 以及利用 GAN 和深度强化学习提升模型提取效率的方法[145]. 这些研究表明, 仅依靠限制查询数量或训练数据的保密性不足以确保模型安全和隐私保护, 需要开发更全面的防御机制.

表 8 从应用的窃取方式、关键技术、目标类型、实验模型以及应用任务方面对不同模型窃取方法进行了比较.

3.4　小结

本节梳理面向 DL-SCP 的对抗攻击方法, 分析三类典型的攻击策略. 规避攻击实施方式灵活多变、方法简单、运行成本低, 是目前面向 DL-SCP 的主流对抗攻击方法. 然而, 由于其改动隐蔽性强且易于实施, 对抗防御的难度较高, 部分攻击依赖固定模式或随机策略, 无法全面覆盖多样化代码结构, 这些内容有待进一步研究. 中毒攻击方法往往需要在隐蔽性与毒性之间实现平衡, 这对目标模型的深入理解提出了较高要求. 当前多数中毒攻击方法通过固定触发模式或者静态注入实现, 触发器多样性与适应性不足, 易被识别, 也难以应对基于动态行为分析的防御机制. 模型窃取方法对查询策略设计和计算资源的需求较高, 实施成本高. 在访问受限的任务场景中, 这类方法取得理想的攻击效果困难, 且部分方法生成的替代模型难以准确复刻原始模型的复杂特征. 表 9 进一步比较了这些研究的优点和不足, 结果也揭示了深度学习模型在多个维度上面临的安全威胁, 也为防御机制构建提供了参考.

相较于计算机视觉和自然语言处理任务, 源代码对抗攻击需同时满足语法正确和语义等价的双重约束, 强调严格的语义保留性. 图像攻击依赖微小像素扰动, 文本攻击多用词语替换或语序调整, 而代码攻击主要通过融合语法结构、语义逻辑和文本序列, 实现多模态融合的扰动操作. 代码攻击要求在不改变程序功能的情况下变换代码的表现形式,

表 8　面向源代码处理任务的模型窃取方法

Table 8　Model stealing methods for source code processing tasks

窃取方式	文献	关键技术	攻击目标	实验模型	应用任务
行为窃取	[138]	通过预测 API, 提出决策树寻径攻击以及模型通用方程求解攻击	目标	神经网络、逻辑回归、决策树	云端机器学习服务
	[85]	设计模型输入输出概率分布, 使其行为随机或不可预测	无目标	线性回归、朴素贝叶斯分类器	加强模型隐私
	[86]	使用句首 token 自回归重复采样语言模型, 并排序“记忆”样本	无目标	GPT-2	缓解隐私泄露
	[139]	利用合成数据与模型交互, 通过知识蒸馏提取替代模型	目标	NARM、BERT4Rec、SASRec	推荐任务
	[140]	基于 AST 中间表示, 实现跨编程语言统一转换	目标	GRU、SrcMarkerTE	代码生成、代码搜索
参数窃取	[142]	根据超参数与模型输出间关联, 反向推断模型超参数	无目标	支持向量机、逻辑回归、岭回归、神经网络	回归与分类任务
	[143]	通过 API 调用窃取语言模型的解码算法和超参数	无目标	GPT-2、GPT-3、GPT-Neo	文本生成

表 9　不同攻击方法的优点和不足

Table 9　Advantages and disadvantages of different attack methods

方法	类别	优点	不足
规避攻击	重命名	代码语法约束严格, 语义一致, 生成高效且迁移性好	标识符候选空间有限制, 攻击多样性不足, 难以攻击代码结构完备的模型
	插入攻击	保留程序语义, 便于自动化生成, 实用性强	Dead-code 固定模板插入易被模型识别, 语义干扰深度不足
	结构等价	扰动隐蔽性好, 转换方式多样化, 生成对抗样本质量高	依赖转换策略生成, 操作复杂, 随机性强, 难以控制语义, 生成效率受限
中毒攻击	数据中毒	隐蔽性强, 迁移性与通用性好, 实施简单, 适用于训练阶段	触发器稳定性差, 难以控制传播过程, 缺乏评估标准, 且攻击方法的鲁棒性与适应性不强
	模型中毒	无需访问训练数据, 攻击隐蔽性高, 适用性好	修改代价大, 难以检测和迁移, 高度依赖模型参数和结构
模型窃取	—	无需模型内部信息, 可复制行为类似模型, 适用于黑盒攻击	查询量大且成本高, 复制模型细节困难, 窃取模型精度有限

从而增加了设计难度. 评估时, 图像和文本常用 l_p 范数衡量扰动大小, 代码中的语法修改可能导致编译失败或功能变化. 因此, 通常引入编译通过率与功能等价性等专有指标进行度量. 代码攻击还需紧密结合具体任务目标, 如在漏洞检测中保留原始漏洞语义, 体现其高度复杂性.

4 面向 DL-SCP 的防御方法

Goodfellow 等[72] 于 2015 年首次提出对抗训练, 这类防御方法通过将对抗样本加入训练集并重新训练模型来提升模型鲁棒性. Sinha 等[146] 从理论上证明, 在一定扰动半径内, 对抗训练能有效增强模型的泛化能力. 随着研究的不断深入, 防御方法呈现多样化, 主要通过修改网络结构、优化训练样本或结合附加网络等手段实现[147−148], 更加注重提升模型的鲁棒性与泛化性[149]. 与此同时, 防御方法的评估与验证也日益重要, 包括鲁棒性评估和可解释性研究等方面[150−151].

随着对抗攻击在功能分类、克隆检测和缺陷检测等源代码处理任务中的持续出现, 学者们提出多种防御策略. 根据防御方法应用的阶段, 参考现有研究 [151−152], 本文将其划分为对抗训练、后门防御、防御蒸馏和数据检测, 如图 7 所示. 对抗训练通过在数据集中加入对抗样本并重新训练模型来实现; 后门防御涉及训练前、训练中和训练后的多阶段干预; 防御蒸馏通过教师模型的概率分布优化训练过程, 以增强学生模型的抗干扰能力; 数据检测则在推理阶段监控输入数据的异常性. 这些方法旨在增强模型在面对对抗攻击时的稳定性与安全性, 是实现鲁棒防御的重要手段.

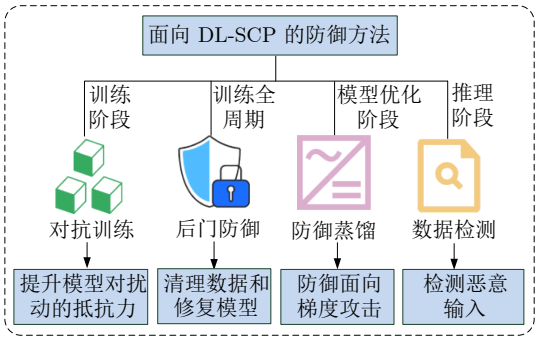

图 7 面向 DL-SCP 的防御方法分类

Fig. 7 Classification of defense methods for DL-SCP

4.1 对抗训练

对抗训练是最常用的防御技术之一[153], 旨在通过提高训练集数据的多样性来抵御对抗攻击. 如图 8 所示, 其流程包括: 生成原始样本及标签、扰动样本特征、检测对抗样本并将其加入到训练集进行模型重训练. 尽管对抗训练具有较好的适应性, 但仍面临训练成本高和过拟合等问题[154].

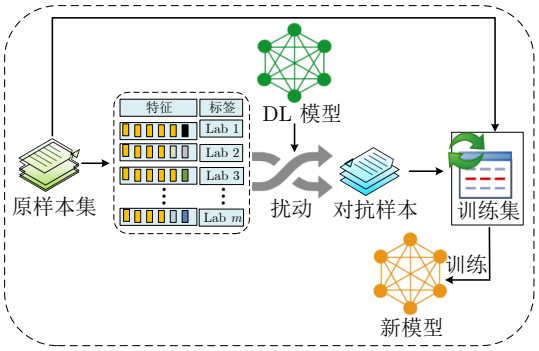

图 8 对抗训练流程图

Fig. 8 Adversarial training flowchart

目前, 直接使用整数规划解决极大极小值问题、通过边界传播最小化损失函数上界以及通过对抗样本增强训练 DL 模型是三类典型的对抗训练手段. 其中, 前两类方法计算代价高, 不适用于源代码处理等高维离散任务. 针对传统的提示学习, Zhang 等[155] 探讨了编码任务中对抗攻击可转移性和防御策略, 提出基于提示的防御方法.

在深度学习领域, 针对对抗样本增强训练问题, Zhang 等[32] 提出通过打破标识符名称与功能标签间的关联来减弱模型的语义依赖性, 但该方法生成样本多样性不足. 为此, 有研究进一步引入周期性训练策略[79], 其优化目标为

$$\min_{\theta}\left\{\sum_{x,\,y_{\text{true}}\in X} L(y_{\text{true}},\,f(x))+\lambda\sum_{x,\,y_{\text{true}}\in X_A} L(y_{\text{true}},\,f(x))\right\} \tag{18}$$

其中, X 是源样本集; X_A 是扰动后样本集; λ 用于调整对抗样本的数量.

考虑源代码语义结构的一致性, Henkel 等[103] 率先提出基于语义保持转换的方法以生成对抗样本, 并构建如下鲁棒优化目标:

$$\arg\min_{\theta\in H}\ \mathrm{E}_{(x,\,y_{\text{true}})\sim D}\left[\max_{x'\in X_A} L(\theta,\,x',\,y_{\text{true}})\right] \tag{19}$$

其中, H 是所有可选模型参数的集合; D 表示数据分布; x' 表示经过扰动的样本.

为进一步减弱模型对非鲁棒特征的依赖, Zhou 等[156] 引入掩码训练策略, 通过替换代码中 k 个标识符生成掩码样本 x', 并引导模型输出与原目标标签

y_{goal} 一致的结果, 其掩码训练目标为:

$$\theta^* = \arg\min_{\theta}\Big(\lambda L_{\text{origin}}(x,\ y_{\text{goal}}) + (1-\lambda)L_{\text{masked}}(x',\ y_{\text{goal}})\Big) \tag{20}$$

其中, $L_{\text{origin}}(\cdot,\ \cdot)$, $L_{\text{masked}}(\cdot,\ \cdot)$ 分别表示原始训练损失函数与掩码训练损失函数.

在面对 token 空间的离散性与梯度不可导性挑战时[101, 119], 上述基于输入空间扰动的方法往往难以直接应用. 为此, Li 等[157] 将扰动定义在连续嵌入空间, 提出如下优化目标:

$$\min_{\theta} \mathrm{E}_{(X,\ y)\sim D}\left[\max_{\|\sigma\|_{\mathrm{F}}\le\varepsilon} L(f_\theta(A(X,\ \sigma)),\ y)\right] \tag{21}$$

其中, F 表示 Frobenius 范数; $A(X,\ \sigma)$ 表示将扰动 σ 应用于输入 X. 除了结构上的扰动优化外, Yang 等[158] 将交叉熵损失 $L_{CE}(\theta)$ 与不信任交叉熵损失 $L_{DCE}(\theta)$ 相结合, 使用的不完全信任损失函数见式 (22), 目的是防止模型对潜在噪声样本的过拟合:

$$\begin{aligned} L(\theta) &= \gamma L_{CE}(\theta) + \beta L_{DCE}(\theta) = \\ &-\gamma\sum_{i=1}^{|m|} q_i \log p_i - \beta\sum_{i=1}^{|m|} p_i \log\left(\theta p_i + (1-\theta)q_i\right) \end{aligned} \tag{22}$$

其中, γ, β 为超参数; p_i 为模型预测第 i 个 token 的概率分布; q_i 为第 i 个 token 的信任标签分布; θ 是调节参数; $|m|$ 表示 token 数量. Yang 等[159] 提出通过度量扰动前后样本差异的数据增强方法, 该方法的优化目标为:

$$\min_{\theta_M} \mathrm{E}_{(x,\ y)\sim D_{gra}}\left[\max_{\sigma\in\varepsilon} L(x+\sigma x,\ y;\theta_M)\right] \tag{23}$$

其中, θ_M 表示模型参数的集合; 期望 E 是对梯度增强数据集 D_{gra} 中的样本进行计算.

表 10 从关键技术、特点、应用任务及防御能力方面对现有对抗训练方法进行了比较. 对抗训练通过重命名、插入和语义转换等攻击方式生成对抗样本, 并利用这些样本重新训练模型来提升模型性能. 虽然对抗训练能提高模型的防御能力, 但其效果有限, 且存在样本多样性受限、训练成本高及迁移能力弱等问题. 此外, 有研究表明, 即使引入大量对抗样本, 模型仍可能被新型攻击绕过[160]. 因此, 开发通用和自适应的防御机制, 已成为对抗机器学习领域亟待解决的关键问题.

4.2　后门防御

后门攻击通过污染深度学习模型, 使模型在特定触发器下产生错误预测. 针对这一威胁, 出现了后门防御策略. 按照后门防御的不同阶段, 本文将其划分为两类: 创建过滤器、检测和清除后门.

4.2.1　创建过滤器

为模型构建过滤器是应对后门攻击的常用防御手段, 尤其在源代码任务中, 能有效结合结构与语义特征实现过滤. 常用策略包括静态过滤与动态防御. 静态过滤通过识别和屏蔽潜在触发器实现后门检测. 例如, 在深度学习领域, 可采用遮挡技术或人工干预去除模型依赖的输入[161], 也可以提取关键 token 识别置信度异常特征[162]. Li 等[125] 采用集成梯度方法评估模型在原始输入与插入可疑 token 后的性能差异. 若性能下降超过阈值 t^*(式 (24)), 则判为中毒样本, 其中 p 和 p_i 分别为插入第 i 个 token 前后模型的性能; 若所有 token 插入后模型性能仍保持稳定 (式 (25)), 则判定数据干净. 动态防御则根据攻击者的知识范围与攻击强度等因素, 动态切换最优防御策略, 以提高模型适应性和攻击不确定性[163].

$$\exists p_i,\ \frac{p-p_i}{p} \ge t^* \tag{24}$$

表 10　不同对抗训练方法的比较

Table 10　Comparison of different adversarial training methods

文献	关键技术	特点	应用任务	能抵御的攻击类型
[154]	通过掩码训练生成轻量级对抗样本	生成的样本可迁移	代码注释生成	重命名标识符
[155]	基于提示的防御, 逆向变换防御	无需重新训练, 依赖上下文学习	代码摘要任务	语义保持的对抗攻击
[32]	用原样本与对抗样本重新训练模型	缓解了重命名标识符对模型产生的影响	功能分类	重命名标识符
[79]	训练集中周期性地加入对抗样本	支持模型鲁棒性的实时更新	功能分类、缺陷检测、代码完整检测	重命名标识符、删除攻击、插入攻击
[103]	多次等价变换源代码样本	方法适用范围广	代码摘要	重命名标识符、插入攻击、源代码变换
[157]	语义保持的对抗性代码嵌入	连续嵌入空间对抗训练	代码问答、缺陷检测、代码搜索	自然语义感知攻击、代码转换攻击
[158]	应用基于检索增强的提示学习以及对抗训练	保持语义一致, 轻量化训练策略, 无需重新训练	缺陷检测、代码问答、代码搜索	重命名标识符、自然语义感知攻击、规则变换攻击
[159]	通过语义距离采样进行数据增强	提升模型鲁棒性, 保证其性能稳定	代码翻译任务	代码转换攻击

$$\forall p_i,\ \frac{p-p_i}{p} < t^* \tag{25}$$

综上, 创建过滤器方法通过结合语法与语义特征, 可在推理阶段识别并过滤含后门的输入, 尤其适用于源代码静态分析. 但该方法难以处理训练时已注入的后门, 也无法消除模型参数中的潜在污染. 为提升模型应对多样化攻击的能力, 根据攻击变化自适应性的学习机制有待研究.

4.2.2 检测和清除后门

检测和清除后门方法依赖于可信数据集, 旨在通过分析干净数据中的异常行为或模式, 定位并清除后门, 保障模型部署前的安全性. 在深度学习领域中, 现有方法主要包括三类: 激活行为分析方法、聚类检测方法与异常建模方法. 激活行为分析方法通过挖掘神经元响应模式识别后门区域, Liu 等[164]发现后门攻击常激活特定神经元, 形成"受损神经元", Sun 等[165]进一步利用样本触发器识别策略以缓解扰动带来的影响; 聚类检测方法通过分析模型内核激活模式并进行聚类, 能有效识别并剔除后门关键词, 实现源代码任务中的毒样本检测[128, 166]; 异常建模方法适合检测隐蔽触发器, 如通过熵与分布差异进行检测[167], 或利用模糊测试使生成对抗网络快速拟合模型边缘样本分布, 识别出污染样本子集[168]. Ramakrishnan 等[119]通过计算样本 x_i 与特征向量 $\boldsymbol{V}$ 的相关性获得异常值评分 $Score(x_i)$, 用于识别并移除后门样本, 表示如下:

$$Score(x_i) = \left\| \left(\boldsymbol{R}(x_i) - \hat{\boldsymbol{R}} \right) \boldsymbol{V}^{\mathrm{T}} \right\|_2 \tag{26}$$

其中, 函数 $\boldsymbol{R}$ 将代码输入表示为向量; $\hat{\boldsymbol{R}}$ 为参考向量; $\boldsymbol{V} = [\boldsymbol{v}_i]_{i=1}^{k}$ 是由前 k 个右奇异向量组成的矩阵.

检测和清除后门方法通过分析模型内部结构和行为特征来检测后门, 并通过清除后门使模型恢复原有的功能. 但这类方法很难检测出少量注入样本的攻击以及隐蔽性较强的后门, 在清除时往往对模型进行微调, 会损坏模型的部分功能, 或是无法彻底清除后门路径.

表 11 从关键技术、特点、应用任务及防御能力方面对后门防御方法进行了比较.

4.3 防御蒸馏

防御蒸馏指根据教师模型构造一个结构相同但复杂度更低的学生模型. Papernot 等[169]首次提出采用高温 softmax 生成软标签, 将神经网络的输出 $F_i(x)$ 定义如下:

$$F_i(x) = \left(\frac{\exp\left(\frac{y_i}{\tau}\right)}{\sum_{j=0}^{|y|-1} \exp\left(\frac{y_j}{\tau}\right)} \right)_{i\in\{0,\cdots,|y|-1\}} \tag{27}$$

其中, τ 为温度参数; $|y|$ 为类别数; y_j 为初始网络预测概率向量.

在深度学习领域中, 为检测恶意软件, Stokes 等[170]首次探索了防御蒸馏在深度恶意软件分类中的应用, 评估权重衰减、分类器集成和蒸馏三种防御方法的有效性, 发现其效果有限. 随后, Singhal 等[97]提出将对抗训练与防御蒸馏相结合的迭代蒸馏技术, 但该方法仍依赖高质量对抗样本. 为提高模型泛化能力, Li 等[171]设计防御内核网络, 将对抗样本转化为特征显著图像以增强模型鲁棒性. Grosse 等[172]则提出基于静态分析的防御蒸馏方法, 利用 softmax 得到模型对输入 x 属于第 i 类的软

表 11 不同后门防御方法的比较

Table 11 Comparison of different backdoor defense methods

类别	文献	关键技术	特点	应用任务	能抵御的攻击类型
创建过滤器	[161]	通过基于遮挡的异常检测技术识别触发器	高召回率, 支持多模型	缺陷检测、克隆检测	木马攻击
	[162]	生成中毒样本并加权优化知识蒸馏	模型结构无需修改	代码搜索	中毒攻击、token 触发器攻击
	[125]	用集成梯度算法探测触发器, 在训练数据中检测出有毒样本并去除	能有效检测中毒样本	缺陷检测、克隆检测、代码修复	后门攻击
	[163]	动态选择和优化模型	动态适应不同威胁等级	恶意软件检测	通用对抗扰动攻击、黑盒攻击
检测和清除后门	[164]	根据神经元输出差异检测后门	适合数据规模大、模型复杂的场景	恶意软件检测	木马攻击
	[165]	通过反转触发器清除后门攻击	减小触发器优化的搜索空间, 最小化扰动影响	缺陷检测、克隆检测、代码搜索	后门攻击
	[166]	对样本神经元激活值进行聚类, 检测中毒样本	适用于多模态以及中毒攻击复杂的场景	数据分类	后门攻击
	[167]	根据删除输入后模型置信度变化来确定并删除关键触发器	用离群值检测来识别有毒代码模型中的输入触发器	漏洞预测、克隆检测	木马攻击
	[168]	结合边缘样本与智能算法	保证模型在中毒后快速复原	入侵检测系统	数据污染攻击
	[119]	应用数据中毒攻击, 用谱特征分析并清除后门攻击	可以提取光谱特征, 并优化异常值检测	代码摘要	插入攻击

标签, 如下式所示

$$f_i(x)=\frac{\exp\left(\frac{z_i(x)}{\tau}\right)}{\sum\limits_{l=1}^{|y|}\exp\left(\frac{z_l(x)}{\tau}\right)} \tag{28}$$

其中, $z_i(x)$ 是模型在输入 x 上对第 i 类输出的原始得分. 为进一步提升防御泛化性, Xia 等[173] 考虑中间层对齐损失 $IL(x,\ s,\ t)$ 与复合损失函数 $L(x,\ y,\ s,\ t)$ 来构建知识蒸馏框架, 分别表示为式 (29)、式 (30).

$$IL(x,\ s,\ t)=\sum_{k=1}^{m_s}L_{MSE}(s_k(x),\ t_{2k}(x)) \tag{29}$$

$$\begin{aligned}L(x,\ y,\ s,\ t)=&\ IL(x,\ s,\ t)+\\ &L_{MSE}\left(s_{\text{classifier}}(x),\ t_{\text{classifier}}(x)\right)+\\ &\sum_{f\in\{s,\ t\}}L_{CE}(f(x),\ y)\end{aligned} \tag{30}$$

其中, m_s 是学生网络的层数; $L_{CE}(\cdot)$ 和 $L_{MSE}(\cdot)$ 分别表示交叉熵损失和均方误差损失函数; $s_k(x)$ 和 $t_{2k}(x)$ 分别表示学生模型第 k 层和教师模型第 $2k$ 层的特征输出; $s_{\text{classifier}}(x)$ 和 $t_{\text{classifier}}(x)$ 分别表示学生模型和教师模型在分类器最后一层的输出.

防御蒸馏可在一定程度上缓解对抗扰动带来的泛化风险, 但其效果依赖于教师模型的质量, 对目标分类攻击防御有限; 同时, 蒸馏过程中易丢失边缘样本的细粒度语义, 从而影响模型的解释性与判别能力. 表 12 从关键技术、特点、应用任务及防御能力方面对防御蒸馏方法进行了比较.

4.4　数据检测

数据检测通过识别输入数据中的对抗样本, 并据此应用防御机制或拒绝分类. 该方法最初应用在图像领域, 通过传统方法如特征聚类[174]、k 近邻[175] 和谱签名分析[122] 等实现样本检测. 在恶意代码检测方面, Zhao 等[176] 结合数据集加载脚本提取、模型反序列化、污点分析和启发式匹配, 实现对投毒攻击的识别与分类. 在深度学习领域中, 针对源代码样本层面的对抗攻击, Rashid 等[177] 基于查询频率、分布与模式等特征构建多层次防御体系, 用于识别恶意查询. 与此同时, 研究者也提出其他检测手段, 包括变量名离群值检测[94]、模型解释与异常检测[178] 和基于注意力的随机平滑[179]. 这些方法通过分析变量名分布、定位噪声标识符或检测神经网络协方差异常来识别对抗样本. 以离群值检测为例[94], 该方法通过在当前代码片段 code 中选取一个最优标识符, 使其与所有同义标识符的平均向量相似度最大. 最优标识符 z^* 的计算方式如下:

$$z^*=\arg\max_{z\in\text{Var}(code)}\left\|\frac{\sum\limits_{t_s\in Sym(code),\ t_s\neq z}\boldsymbol{vec}(t_s)}{|Sym(code)|}-\boldsymbol{vec}(z)\right\|_2 \tag{31}$$

其中, z 表示候选标识符; $\text{Var}(code)$ 表示当前候选标识符集合; t_s 表示同义标识符; $|Sym(code)|$ 表示同义标识符集合的大小; $\boldsymbol{vec}(z)$ 和 $\boldsymbol{vec}(t_s)$ 分别是 z 和 t_s 的向量表示.

为防范模型窃取, 有研究提出暗知识保护的模型功能窃取防御方法[180], 以降低盗版模型的准确率. 同时, 水印或指纹技术也被广泛用于模型所有权的验证[181]. 拒绝分类方法[182] 也被用于阻止模型对异常样本的预测, 但该方法可能对系统响应效率产生一定影响.

综上, 当前数据检测方法能有效检测常见对抗攻击, 但仍面临对语义保持型攻击检测不足、跨语言泛化能力有限等问题, 因此, 构建基于深度语义理解的动态检测框架是亟待解决的一个重要挑战. 表 13 从关键技术、特点、应用任务及防御能力方面对数据检测方法进行了比较.

4.5　小结

本节综述面向 DL-SCP 的四类典型防御方法, 表 14 比较了这些方法的优缺点, 以揭示各方法的

表 12　不同防御蒸馏方法的比较

Table 12　Comparison of different defense distillation methods

文献	关键技术	特点	应用任务	能抵御的攻击类型
[170]	用对抗样本重训练模型, 并实现互信息特征选择	通过动态优化特征提升鲁棒性	恶意软件分类	针对恶意软件的对抗扰动攻击
[97]	根据输出的软标签构建蒸馏模型	分类错误率低且泛化性能好	恶意软件检测和分类	恶意软件攻击
[171]	结合对抗训练、去噪自编码器、集成学习以及输入转换技术	具有针对多种攻击类型的防御能力	恶意软件分类以及多分类任务	FGSM 攻击、随机攻击和模仿攻击等
[172]	高温情况下平滑 DL 模型的误差表面, 提高模型抗扰动能力	生成对抗样本数量多, 模型鲁棒	恶意软件分类	对抗性攻击和混淆攻击
[173]	通过模拟中间层数据训练蒸馏模型	模型间差距小, 且蒸馏模型性能可能更好	恶意软件检测和分类	对抗性攻击和混淆攻击

表 13 不同数据检测方法的比较

Table 13 Comparison of different data detection methods

文献	关键技术	特点	应用任务	能抵御的攻击类型
[122]	频谱签名检测基于表示学习的异常样本识别与过滤	检测效果受限于模型表示	代码搜索	数据投毒攻击
[176]	数据集加载脚本提取、模型反序列化、污点分析、启发式模式匹配	动静分析结合, 多格式支持	恶意软件分类与检测	恶意代码注入、隐蔽后门攻击
[177]	分析查询序列以计算攻击可能性分数, 预测潜在攻击行为	与模型无关的状态防御, 无法适应新数据	恶意软件检测	恶意代码注入、查询攻击以及探测攻击
[94]	利用范数衡量变量间距离以识别出异常值	通过上下文关系分析找到异常标识符	恶意软件检测	重命名标识符
[178]	利用贡献度分布来检测对抗样本	不依赖距离或密度的度量, 时间复杂度更低	恶意代码检测	重命名标识符、插入攻击等
[179]	随机平滑检测、注意力定位噪声、MCIP 去噪	无需模型重训练, 实时处理, 解释性好	功能分类、作者归属、缺陷预测	重命名攻击等

表 14 不同防御方法的优缺点

Table 14 Advantages and disadvantages of different defense methods

方法	类型	优点	不足
对抗训练	—	提升模型鲁棒性, 增强泛化能力, 适应多种攻击形式, 具有较好的通用性	训练成本高、计算资源消耗大, 容易陷入过拟合, 对新型攻击适应性有限
后门防御	创建过滤器	触发器识别准确, 可用于静态、动态分析, 检测性能稳定, 可靠性好	受特征提取质量影响, 容易误判, 新型触发器覆盖困难, 检测成本高, 需人工参与
	检测和清除后门	无需原始训练数据, 适用性强, 有效识别潜在后门区域并清除	建模依赖神经元行为, 容易误检与漏检, 难以应对隐蔽性强或多样化后门策略
防御蒸馏	—	结构简单, 可以抑制对抗梯度传播, 模型鲁棒性好, 适用于多种任务	对超参数敏感, 防御强度受限, 复杂攻击抵御困难, 训练开销大
数据检测	—	模型结构无需修改, 通用性好, 高效识别常见对抗样本	受特征选择影响, 对语义扰动不敏感, 缺乏泛化能力, 易被针对性对抗样本规避

适用条件与局限性. 对抗训练是目前最常用的防御方法, 这类方法实现相对容易, 但训练成本高且效果依赖样本质量; 后门防御主要用于应对代码生成或代码补全任务中训练数据被植入恶意模式的风险, 其方法包括在训练前或训练过程中识别和消除潜在后门, 但这类方法存在误判或漏判的问题; 防御蒸馏借助教师模型平滑输出分布, 其结构相对简单, 但教师模型需要单独训练, 且软标签和温度参数的引入增加了训练复杂度; 数据检测类方法通过语法分析、语义一致性或图结构对比等方式识别可疑样本, 可用于部署阶段的数据预过滤, 但容易被针对性对抗样本规避. 综上, 尽管这些防御方法在特定任务和场景下具备优势, 但仍面临防御效果有限、泛化性不足和计算成本高等问题, 亟须探索多策略融合的综合防御方法.

与图像和文本任务不同, 源代码扰动需保持抽象语法树、控制流和数据流的结构稳定, 极大地限制了可用扰动空间, 从而提升了防御设计的难度. 与视觉模型对局部纹理的依赖或语言模型的上下文建模不同, 代码语义具有从词汇、句法到语义的多层次结构, 且跨函数、跨模块关联性强. 因此, 仅依赖局部特征或上下文难以实现有效防御. DL-SCP 防御方法需融合结构化表示学习 (如图神经网络) 与程序分析技术, 确保语法与语义的一致性. 文本级防御如变量标准化难以应对控制流混淆, 而仅依赖 AST 结构的策略则可能忽视变量间的数据流关系. 因此, 构建一个既能理解程序结构又能感知上下文语义的混合机制, 是实现鲁棒防御的关键.

5 面向 DL-SCP 的对抗攻防数据集

在面向源代码处理任务的深度学习领域, 针对不同应用场景的对抗攻防研究已积累了丰富的实验数据集. 本节梳理该领域中具有代表性的 8 个数据集, 涵盖分类任务和生成任务两大类别, 这些数据集在学术研究中得到广泛应用[183]. 如表 15 所示, 本文从数据集名称、任务分类、标签数目样本分布以及编程语言支持等多个维度对这些数据集进行归纳和分析.

5.1 分类任务数据集

5.1.1 BigCloneBench 数据集

该数据集是一个面向 Java 语言、广泛应用于克隆检测的大型语义数据集, 涵盖项目内和跨项目两种克隆场景. 它通过挖掘大型跨项目源代码仓库中的函数来构建, 用于评估各种克隆检测工具的性能.

5.1.2 POJ-104 数据集

该数据集来自教学用编程评测平台 (OJ 系统),

表 15　源代码处理任务中的数据集

Table 15　Datasets in source code processing tasks

数据集	分类/生成	标签数目	样本数量 (千) 训练/测试/验证	语言	评估指标	应用任务
BigCloneBench[184]	分类	10	900/416/416	Java	召回率、精确率、F1 值	克隆检测
POJ-104[185]	分类	104	32/8/12	C/C++	准确率	克隆检测
GCJ[186]	分类	70	0.528/0.132/−	Python	准确率	作者归属
Devign[187]	分类	2	21/2.7/2.7	C	准确率、F1 值	缺陷检测
PY150[188]	生成	—	100/5/50	Python	精确率、准确率、召回率	代码补全
Github Java Corpus[189]	生成	—	13/7/8	Java	精确率、准确率、召回率	代码补全
CONCODE[190]	生成	—	100/2/2	Java	准确率、BLEU、CodeBLEU	代码生成
CodeSearchNet[191]	生成	—	908/45/53	Go、Java、JavaScript、PHP、Python、Ruby	NDCG	代码摘要

OJ 通过自动化执行代码来验证提交程序对特定问题解答的正确性. 数据集中含有 104 个程序类别, 每个类别下包含 500 个程序, 其任务是在给定程序的情况下, 检索能解决相同问题的其他程序.

5.1.3　GCJ 数据集

该数据集中的数据来自 Google 主办的年度国际编码竞赛, 其数据由竞赛中的编程问题以及参赛者提交的解决方案构成. 该数据集包含 70 个参赛者提交的代码记录, 每个参赛者需要独立解决 10 个不同的编程问题, 因此, 数据集中共有 700 个源代码文件.

5.1.4　Devign 数据集

该数据集包含来自两个大型 C 语言开源项目的 27 318 个经人工标记的函数, 数据包含多样化功能, 并且与安全密切相关. 实际研究中, 数据集往往被随机打乱, 以保证模型的泛化能力.

5.2　生成任务数据集

5.2.1　PY150 数据集

该数据集包含从 Github 上收集的 15 万个 Python 源文件. 训练数据与测试数据文件数量分别为 10 万个和 5 万个, 它们分别包含 7.63×10^7 和 3.72×10^7 个 token, 主要用于源代码的补全任务.

5.2.2　Github Java Corpus 数据集

该数据集包含超过 1.4 万个 Java 项目. 其设计和构建借鉴了文献 [192−193] 的研究成果, 数据集使用语料库中 1% 的子集, 训练集、验证集和测试集分别由 1 580 万、380 万和 530 万个 token 组成[183].

5.2.3　CONCODE 数据集

该数据集含有 33 000 个 Java 项目. 其中的样本以三元组形式表示, 包括代码环境、代码片段以及相应的自然语言描述. 数据集的任务是从自然语言描述 (Javadoc 风格的方法注释) 和类环境中生成类成员函数.

5.2.4　CodeSearchNet 数据集

该数据集包含 Go、Java 等六种编程语言的开源代码, 包含约 600 万个功能. 它通过从开源代码库中抓取函数与文档注释来构建, 最终生成 200 万个数据点, 并将它们配对成自然语言注释. 这些配对的数据经过清理和格式化处理后, 形成一个用于训练与评估语义代码搜索模型的大规模数据集.

5.3　小结

数据集的多样性是提升深度学习模型鲁棒性与对抗攻防效果的关键. 从表 15 可见, 现有数据集涵盖多种编程语言和任务场景. 但部分数据集局限于单一语言或特定场景, 缺乏多语言混合与跨模型支持. 此外, 这些数据集在复杂代码语义与上下文表达上仍表现出不足.

不同于常见的基于文本或图像的数据集, 源代码数据集在结构化建模、可运行性和任务适配性等方面具有显著的特点. 首先, 代码数据通常从词法、语法以及语义上进行描述, 结合类、方法和依赖关系等各种结构因素, 并通过语法解析和程序分析等方式表示代码的结构化特征; 其次, 源代码具备可编译性, 对于数据集的构建需要结合静态分析与动态测试等手段, 以保证代码能够正常运行; 最后, 代码数据集更注重于与具体任务的关联, 包括代码功能分类、缺陷检测、代码摘要生成等多种任务, 其评估指标通常包括测试覆盖率、变量修改率等代码特有的度量标准. 代码数据集的这些特性均与基于文本或图像的数据集有明显的区别.

6　面临的挑战与未来研究展望

近年来, 针对面向 DL-SCP 的对抗攻防技术,

研究者们开展了较多的探索与研究, 并提出多种有效的对抗样本生成策略和防御方法. 然而, 由于源代码自身的语法结构、语义保留等方面的复杂性和独特性, 这些方法仍存在诸多不足. 针对这些不足, 目前源代码对抗攻防方法的研究主要聚焦于三个核心方向: 基于不同代码样本表示策略设计对抗攻击方法、代码模型鲁棒性评估方法以及应对对抗攻击的防御方法. 下面将围绕这三大方向, 进一步讨论面向 DL-SCP 的对抗攻防技术所面临的主要挑战与未来研究方向.

6.1 面临的挑战

6.1.1 保留语义的词表压缩问题

与自然语言词汇相比, 代码词汇更具开放性和复杂性[21]. 由于代码标识符命名的高度自由性, 导致其词汇多样且词表规模庞大, 进而引发未登录词 (out-of-vocabulary, OOV) 问题. 为缓解这一问题, 现有研究提出了多种方法, 包括子词拆分、字符级建模以及标识符级建模等. 例如, Kanade 等[194] 和 Feng 等[51] 采用 BPE (byte pair encoding) 分词方法来压缩词表; Karampatsis 等[193] 的研究进一步表明应用 BPE 方法可以显著减少词表规模; Chirkova 等[195] 提出通过标识符匿名化来缓解 OOV 问题. 然而, 这些方法存在局限性: 子词拆分法会导致词表空间扩大, 适应性受限; 字符级建模因生成的序列较长以及代码的语义边界不明确, 增加了模型学习难度; 标识符级建模则会导致标识符语义信息丢失, 因此在需要语义推理的任务中适用性较差. 这些局限性不仅影响模型对代码语义的理解, 还可能被攻击者利用词表缺陷构造恶意标识符, 威胁模型安全. 因此, 如何在保留代码语义的前提下有效压缩词表空间, 是当前源代码表示中亟待解决的关键问题之一.

6.1.2 代码动态语义提取与表示问题

在常见的规避攻击和中毒攻击中, 攻击者往往利用标识符、固定语句模式或者设置触发器等静态特征来生成对抗样本. 尽管在代码执行过程中表现出的动态特征 (如方法调用) 对攻防技术的实现同样重要, 但因其依赖于运行时的环境, 提取并表示源代码的动态特征困难. 尤其在当前主流的代码预训练语言模型中, 如 CodeBERT 和 CodeT5 等, 这些模型主要以静态文本序列作为输入, 难以有效融入代码的动态特征, 严重制约了其在语义分析任务中的应用.

6.1.3 黑盒攻击中查询效率低的问题

面向黑盒的查询攻击是目前最主要的源代码对抗攻击手段之一. 这类攻击通过多次向目标模型输入代码样本, 并结合模型的输出结果来反推其决策边界. 然而, 对于包含复杂语法树结构和高语义相关性的源代码, MHM[32] 和 Alert[95] 等主流方法往往因频繁查询而导致攻击效率较低; 而对于 CARL[49]、CodeBERT-Attack[98] 和 DWRAttack[196] 等针对预训练模型的攻击方法, 由于模型参数规模庞大、训练和攻击时间长, 设计同时兼顾高效和低成本的对抗攻击手段困难. 因此, 如何结合代码的语法结构和语义信息, 在有效降低查询次数的同时进行有效的攻击, 仍是当前对抗攻击研究中的难题.

6.1.4 攻防方法的跨语言和跨模型适应性问题

由于编程语言和开发者编写习惯的差异, 源代码样本之间具有不同的程序风格、结构形式和语义依赖关系. 例如, 代码缩进、标识符命名规则和控制流结构等方面的不同. 这使得通用攻防方法的设计面临巨大挑战. 现有攻防技术往往面向特定编程语言或者开发环境, 难以直接迁移到其他语言或者模型中使用. 同时, 不同模型对输入代码的表示和语义处理方式也存在差异, 导致通用对抗攻防机制的设计更为复杂. 因此, 从跨语言与跨模型的视角出发, 提升对抗攻防技术的适应性, 是当前亟待解决的关键挑战.

6.1.5 应对代码大模型的对抗滥用问题

随着 Code llama[197]、Starcoder[198] 等代码大模型 (code large language models, Code LLMs) 的出现, 对抗性安全问题日益突出[199]. 尽管大模型语义理解与指令执行能力强, 但也易受到提示注入、编码风格操控、语义规避等攻击方式的影响. 例如, Li 等[200] 提出一种基于恶意指令注入的系统提示中毒攻击, 其攻击目标是在后续用户交互中产生持续影响; Zou 等[201] 利用贪婪与梯度搜索方法生成对抗性后缀, 导致模型生成有害数据, 污染代码上下文; Yi 等[202] 则指出攻击者可以利用大语言模型的透明度漏洞自动构造隐蔽性高的对抗样本, 并诱导模型生成恶意内容. 在基于自然语言提示驱动的自动化编程中, 这类滥用行为可能导致 Code LLMs 污染、数据泄露等严重后果, 现有防御手段防范困难. 如何在发挥 Code LLMs 功能的同时, 识别并防范其潜在的安全风险, 已成为当前代码大模型面临的关键挑战.

6.1.6 鲁棒性评估框架不统一问题

尽管目前对抗训练已被广泛用于提升深度学习模型的鲁棒性, 但在源代码处理任务中, 对抗样本生成与鲁棒性评估目前并未建立统一标准. 源代码受语法与语义限制, 其扰动空间高度离散, 这一特

性导致对抗训练中难以构造既保持语义一致性又具有足够攻击强度的对抗样本. 当前主流方法多依赖启发式设计, 由于鲁棒性评估框架不统一, 导致不同方法间缺乏可比性, 其结果也难以复现和推广. 因此, 构建鲁棒性评估体系是促进该领域持续发展的关键问题之一.

6.2 研究方向

根据上述面临的挑战, 本文认为未来该领域可以从下面几个方向展开研究.

6.2.1 融合语法与语义信息的词表压缩方法研究

针对源代码处理中的 OOV 问题, 未来的研究方向包括: 一是现有 BPE 等静态分词策略缺乏代码语义信息, 可根据标识符在不同上下文中的语义, 动态调整子词划分方式, 例如, 通过引入语义注意机制或图神经网络等方法实现; 二是结合抽象语法树、控制流图、数据流图以及程序依赖图等结构信息, 在语法约束下重构词表, 增强样本的语义对齐能力; 三是通过同时优化多个源代码处理任务, 如命名预测和代码摘要生成等任务, 实现多任务预训练. 该方法可促使模型学习更通用的标识符语义表示, 在保留语义信息的前提下有效压缩词表规模.

6.2.2 面向 DL-SCP 中代码语义的可解释性研究

为应对代码动态语义难以提取与建模的问题, 并提升对抗攻防过程的透明性, 增强 DL-SCP 任务中对代码语义的可解释性已成为重要的研究方向. 源代码任务可解释性需考虑结构层次与语义逻辑, 从而理解模型面对代码语义扰动时的响应. 为实现有效解释, 杨馨悦等[203] 通过 AST、控制流图与神经网络激活的对齐分析, 揭示了结构扰动对模型行为的影响; Bulla 等[204] 通过子词概率和逻辑回归识别关键代码片段, 并将其可视化以直观理解模型的决策依据. 然而, 当前可解释性方法仍面临诸多挑战: 一是对抗扰动会影响解释算法的稳定性, 且难以识别语义一致但功能改变的对抗样本; 二是现有研究普遍关注代码静态特征, 并未充分挖掘和利用其动态语义信息, 这导致模型在面对代码结构与语义扰动时, 解释的稳定性与可信度不高, 获得可靠的解释结果困难; 三是多源代码特征 (如语法结构、数据流等) 融合不充分, 尚未形成统一且通用的解释表示. 因此, 构建稳定且通用的解释框架, 以加深对模型决策过程的理解与对抗行为的识别能力, 这将是 DL-SCP 安全研究的重要方向.

6.2.3 源代码黑盒对抗攻击查询优化研究

当前源代码处理任务中黑盒对抗攻击查询效率低, 其根本原因在于源代码具有高维的结构表示和复杂的语义依赖关系, 导致对抗攻击潜在扰动空间庞大. 为缓解该问题, 一是可以缩小特征搜索空间, 提取对模型输出影响显著的区域, 例如, 通过抽象语法树剪枝、依赖图聚类等方法来实现; 二是对面向源代码处理任务的深度学习模型进行压缩, 提高攻击效率, 例如, Zhang 等[205] 和 Shi 等[206] 提出通过模型压缩来优化模型训练过程; 三是通过引入黑盒优化算法, 如贝叶斯优化、进化搜索等, 在有限查询成本下确定有效的扰动组合, 以减少模型调用次数; 四是针对访问受限或训练耗时长的代码预训练模型 (如 CodeBERT), 可采用迁移学习思想训练代理模型, 通过在代理模型上生成扰动, 并将其迁移至目标模型执行攻击, 从而降低查询成本. 综上, 结合代码语义与结构特性, 设计高效的黑盒攻击策略是未来研究的重要方向.

6.2.4 跨语言和多模型对抗攻防技术研究

当前面向源代码处理任务的对抗攻击方法, 仍主要集中于特定编程语言和模型架构上进行优化. 例如, 基于 CodeT5 和 GraphCodeBERT 的代码预训练模型, 或者基于 LSTM 和 GRU 等通用模型. 虽然这些方法在局部场景中攻击效果较好, 但迁移学习对抗攻击适应性研究目前仍显不足, 多编程语言适应性问题也尚未得到充分研究. 随着软件系统多语言混合开发的普及以及多样化深度学习模型架构的不断出现, 亟须设计可适应多种编程范式和模型架构的通用对抗攻防框架, 以进一步提高对抗攻击的泛化能力和防御方法的效率.

6.2.5 代码大语言模型的鲁棒性研究

为应对代码大语言模型面临的攻击, 仍需进一步构建和完善 Code LLMs 鲁棒性增强机制. 一是通过上下文异常检测、语义一致性分析等方法过滤恶意提示, 结合对抗训练与防御蒸馏技术, 提升模型对对抗性输入的抵御能力; 二是引入基于规则或微调模型的输出审查模块, 限制模型生成有害代码内容; 三是为约束 Code LLMs 对敏感数据的过度记忆问题, 可借助影响函数分析或成员推理方法识别高风险样本. 通过移除对模型影响大的敏感数据, 从而实现对训练数据的隐私保护.

6.2.6 面向 DL-SCP 的模型鲁棒性评估框架研究

随着对抗攻击的不断扩展, 如何提升代码大模型对输入扰动的鲁棒性, 并生成正确和可信的数据, 仍有待深入研究. 可以从以下两个方向考虑: 一是应用 TransCoder[207] 或 DeepMutation[208] 等工具生成多样化的对抗样本, 并据此构造稳健的对抗训练框架, 确保模型能够识别与防御输入扰动; 二是构建覆盖多种攻击方式和度量指标的统一鲁棒性评估

框架, 以便在统一标准下对不同方法进行公平比较和综合评价.

7 结束语

尽管深度学习模型凭借其强大的建模能力, 已被广泛应用到源代码处理任务中, 但是其面临的安全问题不容忽视. 特别是在对抗攻击场景下, 模型鲁棒性不足严重威胁其本身以及软件系统的安全性. 为此, 本文围绕源代码处理任务, 系统梳理和分析该领域深度学习模型的应用现状及其对抗攻防方法.

与已有综述不同, 本文从源代码特有的语法结构和语义依赖出发, 系统阐述主流对抗攻防技术的实现细节, 并深入分析现有方法的优势与不足. 基于源代码的语义建模, 本文将源代码处理任务中应用的深度学习模型细粒度划分为语义对齐模型、程序行为模型和单模态语义模型三大类; 同时, 针对源代码处理任务, 比较了典型的三类对抗攻击方法: 规避攻击、中毒攻击以及模型窃取; 本文还梳理了四种常见的防御方法: 对抗训练、后门防御、防御蒸馏以及数据检测, 并分析这些对抗攻防方法的原理、技术实现、适用性以及存在不足. 为支撑对抗攻防方法的深入研究, 本文进一步对当前常用的源代码数据集进行概括与分类, 明确其覆盖的编程语言、评估指标及应用场景. 据此, 针对面向 DL-SCP 的对抗安全问题, 讨论当前面临的关键挑战, 包括保留语义的词表压缩问题、代码动态语义提取与表示问题以及黑盒攻击中查询效率低的问题等. 针对这些挑战, 本文提出未来研究可以从以下几个方向展开: 融合语法与语义信息的词表压缩方法研究、面向 DL-SCP 中代码语义的可解释性研究以及源代码黑盒对抗攻击查询优化研究等. 上述挑战和研究方向表明 DL-SCP 在代码结构表示和语义理解等方面有待完善, 也指明了对抗安全领域未来可能的研究方向. 本综述旨在为源代码处理任务中的安全性保障提供借鉴, 为该领域研究的进一步发展与应用提供参考和帮助.

参考文献

1 Xu Y W, Khan T M, Song Y, Meijering E. Edge deep learning in computer vision and medical diagnostics: A comprehensive survey. *Artificial Intelligence Review*, 2025, **58**(3): Article No. 93

2 Ahmed S F, Alam M S B, Kabir M, Afrin S, Rafa S J, Mehjabin A, et al. Unveiling the frontiers of deep learning: Innovations shaping diverse domains. *Applied Intelligence*, 2025, **55**(7): Article No. 573

3 Lei L, Yang Q L, Yang L, Shen T, Wang R X, Fu C B. Deep learning implementation of image segmentation in agricultural applications: A comprehensive review. *Artificial Intelligence Review*, 2024, **57**(6): Article No. 149

4 Kamaluddin M I, Rasyid M W K, Abqoriyyah F H, Saehu A. Accuracy analysis of DeepL: Breakthroughs in machine translation technology. *Journal of English Education Forum*, 2024, **4**(2): 122−126

5 Chen X P, Hu X, Huang Y, Jiang H, Ji W X, Jiang Y J, et al. Deep learning-based software engineering: Progress, challenges, and opportunities. *Science China Information Sciences*, 2025, **68**(1): Article No. 111102

6 Bu W J, Shu H, Kang F, Hu Q, Zhao Y T. Software subclassification based on BERTopic-BERT-BiLSTM model. *Electronics*, 2023, **12**(18): Article No. 3798

7 Ameri R, Hsu C C, Band S S. A systematic review of deep learning approaches for surface defect detection in industrial applications. *Engineering Applications of Artificial Intelligence*, 2024, **130**: Article No. 107717

8 Vijayanandan T, Banujan K, Induranga A, Kumara B T G S, Koswattage K. LeONet: A hybrid deep learning approach for high-precision code clone detection using abstract syntax tree features. *Big Data and Cognitive Computing*, 2025, **9**(7): Article No. 187

9 Shen Y H, Ju X L, Chen X, Yang G. Bash comment generation via data augmentation and semantic-aware CodeBERT. *Automated Software Engineering*, 2024, **31**(1): Article No. 30

10 Wang Shang-Wen, Liu Kui, Lin Bo, Li Li, Klein J, Bissyandé T F, et al. Fine-grained defect localization based on pointer neural network. *Journal of Software*, 2024, **35**(4): 1841−1860
(王尚文, 刘逵, 林博, 黎立, Klein J, Bissyandé T F, et al. 基于指针神经网络的细粒度缺陷定位. 软件学报, 2024, **35**(4): 1841−1860)

11 Liu S G, Cao D, Kim J, Abraham T, Montague P, Camtepe S, et al. EaTVul: ChatGPT-based evasion attack against software vulnerability detection. In: Proceedings of the 33rd USENIX Conference on Security Symposium. Philadelphia, USA: USENIX, 2024. Article No. 411

12 Chen Si-Hong, Shen Hao-Jing, Wang Ran, Wang Xi-Zhao. Relationship between prediction uncertainty and adversarial robustness. *Journal of Software*, 2022, **33**(2): 524−538
(陈思宏, 沈浩靖, 王冉, 王熙照. 预测不确定性与对抗鲁棒性的关系研究. 软件学报, 2022, **33**(2): 524−538)

13 Javed H, El-Sappagh S, Abuhmed T. Robustness in deep learning models for medical diagnostics: Security and adversarial challenges towards robust AI applications. *Artificial Intelligence Review*, 2024, **58**(1): Article No. 12

14 Mei S H, Lian J W, Wang X F, Su Y R, Ma M Y, Chau L P. A comprehensive study on the robustness of deep learning-based image classification and object detection in remote sensing: Surveying and benchmarking. *Journal of Remote Sensing*, 2024, **4**: Article No. 0219

15 Alahmed S, Alasad Q, Yuan J S, Alawad M. Impacting robustness in deep learning-based NIDS through poisoning attacks. *Algorithms*, 2024, **17**(4): Article No. 155

16 Qin Zhen, Zhuang Tian-Ming, Zhu Guo-Song, Zhou Er-Qiang, Ding Yi, Geng Ji. Survey of security attack and defense strategies for artificial intelligence model. *Journal of Computer Research and Development*, 2024, **61**(10): 2627−2648
(秦臻, 庄添铭, 朱国淞, 周尔强, 丁熠, 耿技. 面向人工智能模型的安全攻击和防御策略综述. 计算机研究与发展, 2024, **61**(10): 2627−2648)

17 Wang Xu-Tong, Yin Jie, Liu Chao-Ge, Xu Chen-Chen, Huang Hao, Wang Zhi, et al. A survey of backdoor attacks and defenses on neural networks. *Chinese Journal of Computers*, 2024, **47**(8): 1713−1743
(汪旭童, 尹捷, 刘潮歌, 徐辰晨, 黄昊, 王志, 等. 神经网络后门攻击与防御综述. 计算机学报, 2024, **47**(8): 1713−1743)

18 Zhang C Y, Hu M W, Li W H, Wang L J. Adversarial attacks and defenses on text-to-image diffusion models: A survey. *Information Fusion*, 2025, **114**: Article No. 102701

19 Chen N, Sun Q S, Wang J N, Gao M, Li X L, Li X. Evaluating and enhancing the robustness of code pre-trained models through structure-aware adversarial samples generation. In: Proceedings of the Findings of the Association for Computational Linguistics. Singapore: ACL, 2023. 14857−14873

20 Na C W, Choi Y S, Lee J H. DIP: Dead code insertion based black-box attack for programming language model. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Toronto, Canada: ACL, 2023. 7777−7791

21 Allamanis M, Barr E T, Devanbu P, Sutton C. A survey of machine learning for big code and naturalness. *ACM Computing Surveys*, 2019, **51**(4): Article No. 81

22 Xu Y, Cheng M. Multi-view feature fusion model for software bug repair pattern prediction. *Wuhan University Journal of Natural Sciences*, 2023, **28**(6): 493−507

23 Wan Y, Bi Z Q, He Y, Zhang J G, Zhang H Y, Sui Y L, et al. Deep learning for code intelligence: Survey, benchmark and toolkit. *ACM Computing Surveys*, 2024, **56**(12): Article No. 309

24 Yang Y M, Xia X, Lo D, Grundy J. A survey on deep learning for software engineering. *ACM Computing Surveys*, 2022, **54**(10s): Article No. 206

25 Devanbu P, Dwyer M, Elbaum S, Lowry M, Moran K, Poshyvanyk D, et al. Deep learning & software engineering: State of research and future directions. arXiv preprint arXiv: 2009.08525, 2020.

26 Yang Yan-Jing, Mao Run-Feng, Tan Rui, Shen Hai-Feng, Rong Guo-Ping. Robustness verification method for artificial intelligence systems based on source code processing. *Journal of Software*, 2023, **34**(9): 4018−4036
(杨焱景, 毛润丰, 谭睿, 沈海峰, 荣国平. 源码处理场景下人工智能系统鲁棒性验证方法. 软件学报, 2023, **34**(9): 4018−4036)

27 She X Y, Liu Y, Zhao Y J, He Y L, Li L, Tantithamthavorn C, et al. Pitfalls in language models for code intelligence: A taxonomy and survey. arXiv preprint arXiv: 2310.17903, 2023.

28 Sun Wei-Song, Chen Yu-Chen, Zhao Zi-Han, Chen Hong, Ge Yi-Fei, Han Ting-Xu, et al. Survey on security of deep code models. *Journal of Software*, 2025, **36**(4): 1461−1488
(孙伟松, 陈宇琛, 赵梓含, 陈宏, 葛一飞, 韩廷旭, 等. 深度代码模型安全综述. 软件学报, 2025, **36**(4): 1461−1488)

29 Ji Tian-Tian, Fang Bin-Xing, Cui Xiang, Wang Zhong-Ru, Gan Rui-Ling, Han Yu, et al. Research on deep learning-powered malware attack and defense techniques. *Chinese Journal of Computers*, 2021, **44**(4): 669−695
(冀甜甜, 方滨兴, 崔翔, 王忠儒, 甘蕊灵, 韩宇, 等. 深度学习赋能的恶意代码攻防研究进展. 计算机学报, 2021, **44**(4): 669−695)

30 Qu Y B, Huang S, Yao Y M. A survey on robustness attacks for deep code models. *Automated Software Engineering*, 2024, **31**(2): Article No. 65

31 Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I J, et al. Intriguing properties of neural networks. In: Proceedings of the 2nd International Conference on Learning Representations. Banff, Canada: ICLR, 2014.

32 Zhang H Z, Li Z, Li G, Ma L, Liu Y, Jin Z. Generating adversarial examples for holding robustness of source code processing models. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. New York, USA: AAAI, 2020. 1169−1176

33 Kumar S, Gupta S, Buduru A B. BB-Patch: BlackBox adversarial patch-attack using zeroth-order optimization. arXiv preprint arXiv: 2405.06049, 2024.

34 Hector K, Moëllic P A, Dutertre J M, Dumont M. Fault injection and safe-error attack for extraction of embedded neural network models. In: Proceedings of the European Symposium on Research in Computer Security. The Hague, The Netherlands: Springer, 2024. 644−664

35 Li C, Yao W, Wang H D, Jiang T S, Zhang X Y. Bayesian evolutionary optimization for crafting high-quality adversarial examples with limited query budget. *Applied Soft Computing*, 2023, **142**: Article No. 110370

36 Li Zi-Tuo, Sun Jian-Bin, Yang Ke-Wei, Xiong De-Hui. A review of adversarial robustness evaluation for image classification. *Journal of Computer Research and Development*, 2022, **59**(10): 2164−2189
(李自拓, 孙建彬, 杨克巍, 熊德辉. 面向图像分类的对抗鲁棒性评估综述. 计算机研究与发展, 2022, **59**(10): 2164−2189)

37 Wei X X, Kang C X, Dong Y P, Wang Z Y, Ruan S W, Chen Y B, et al. Real-world adversarial defense against patch attacks based on diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025, **47**(12): 11124−11140

38 Ilyas A, Santurkar S, Tsipras D, Engstrom L, Tran B, Madry A. Adversarial examples are not bugs, they are features. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver, Canada: ACM, 2019. Article No. 12

39 Wang W Q, Wang L N, Wang R, Ye A S, Tang B X. Towards a robust deep neural network in text domain a survey. arXiv preprint arXiv: 1902.07285, 2019.

40 Zhang Xin, Zhang Han, Niu Man-Yu, Ji Li-Xia. Adversarial sample detection in computer vision: A survey. *Computer Science*, 2025, **52**(1): 345−361
(张鑫, 张晗, 牛曼宇, 姬莉霞. 计算机视觉领域对抗样本检测综述. 计算机科学, 2025, **52**(1): 345−361)

41 Jiang W, He Z Y, Zhan J Y, Pan W J. Attack-aware detection and defense to resist adversarial examples. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021, **40**(10): 2194−2198

42 Zhao Zi-Tian, Zhan Wen-Han, Duan Han-Cong, Wu Yue. Study on adversarial robustness of deep learning models based on SVD. *Computer Science*, 2023, **50**(10): 362−368
(赵子天, 詹文翰, 段翰聪, 吴跃. 基于 SVD 的深度学习模型对抗鲁棒性研究. 计算机科学, 2023, **50**(10): 362−368)

43 Samangouei P, Kabkab M, Chellappa R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In: Proceedings of the 6th International Conference on Learning Representations. Vancouver, Canada: OpenReview. net, 2018.

44 Dong Qing-Kuan, He Jun-Lin. Robustness enhancement method of deep learning model based on information bottleneck. *Journal of Electronics & Information Technology*, 2023, **45**(6): 2197−2204
(董庆宽, 何浚霖. 基于信息瓶颈的深度学习模型鲁棒性增强方法. 电子与信息学报, 2023, **45**(6): 2197−2204)

45 Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis, USA: ACL, 2019. 4171−4186

46 Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019, **1**(8): 1−24

47 Sharma R, Chen F X, Fard F, Lo D. An exploratory study on code attention in BERT. In: Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension (ICPC). Pittsburgh, USA: IEEE, 2022. 437−448

48 Zeng Z R, Tan H Z, Zhang H T, Li J, Zhang Y Q, Zhang L M. An extensive study on pre-trained models for program understanding and generation. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. Virtual Event: ACM, 2022. 39−51

49 Yao K C, Wang H, Qin C, Zhu H S, Wu Y J, Zhang L B. CARL: Unsupervised code-based adversarial attacks for programming language models via reinforcement learning. *ACM Transactions on Software Engineering and Methodology*, 2025,

34(1): Article No. 22

50 Niu C G, Li C Y, Ng V, Chen D X, Ge J D, Luo B. An empirical comparison of pre-trained models of source code. In: Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE). Melbourne, Australia: IEEE, 2023. 2136−2148

51 Feng Z Y, Guo D Y, Tang D Y, Duan N, Feng X C, Gong M, et al. CodeBERT: A pre-trained model for programming and natural languages. In: Proceedings of the Findings of the Association for Computational Linguistics. Virtual Event: ACL, 2020. 1536−1547

52 Li Z Y, Lu S, Guo D Y, Duan N, Jannu S, Jenks G, et al. CodeReviewer: Pre-training for automating code review activities. arXiv preprint arXiv: 2203.09095, 2022.

53 Chakraborty S, Ahmed T, Ding Y R B, Devanbu P T, Ray B. NatGen: Generative pre-training by "naturalizing" source code. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore: ACM, 2022. 18−30

54 Wang Y, Wang W S, Joty S, Hoi S C H. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Punta Cana, Dominican Republic: ACL, 2021. 8696−8708

55 Wang Y, Le H, Gotmare A, Bui N, Li J N, Hoi S. CodeT5+: Open code large language models for code understanding and generation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Singapore: ACL, 2023. 1069−1088

56 Ding Y R B, Chakraborty S, Buratti L, Pujar S, Morari A, Kaiser G, et al. CONCORD: Clone-aware contrastive learning for source code. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. Seattle, USA: ACM, 2023. 26−38

57 Poesia G, Polozov O, Le V, Tiwari A, Soares G, Meek C, et al. Synchromesh: Reliable code generation from pre-trained language models. In: Proceedings of the 10th International Conference on Learning Representations. Virtual Event: OpenReview.net, 2022.

58 Wang R C, Xu S L, Tian Y, Ji X Y, Sun X B, Jiang S J. SCL-CVD: Supervised contrastive learning for code vulnerability detection via GraphCodeBERT. *Computers & Security*, 2024, **145**: Article No. 103994

59 Zeng J W, Zhang T, Xu Z. DG-Trans: Automatic code summarization via dynamic graph attention-based Transformer. In: Proceedings of the 21st IEEE International Conference on Software Quality, Reliability and Security (QRS). Hainan, China: IEEE, 2021. 786−795

60 Yang J, Fu C, Deng F Y, Wen M, Guo X W, Wan C H. Toward interpretable graph tensor convolution neural network for code semantics embedding. *ACM Transactions on Software Engineering and Methodology*, 2023, **32**(5): Article No. 115

61 Peng J X, Wang Y, Xue J F, Liu Z Y. Fast cross-platform binary code similarity detection framework based on CFGs taking advantage of NLP and inductive GNN. *Chinese Journal of Electronics*, 2024, **33**(1): 128−138

62 Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 2020, **404**: Article No. 132306

63 Cahuantzi R, Chen X Y, Güttel S. A comparison of LSTM and GRU networks for learning symbolic sequences. In: Proceedings of the Conference on Intelligent Computing. Zhengzhou, China: Springer, 2023. 771−785

64 Wang R Y, Zhang H W, Lu G L, Lyu L, Lyu C. Fret: Functional reinforced Transformer with BERT for code summarization. *IEEE Access*, 2020, **8**: 135591−135604

65 Alqarni M, Azim A. Low level source code vulnerability detection using advanced BERT language model. In: Proceedings of the 35th Canadian Conference on Artificial Intelligence. Toronto, Canada: Canadian Artificial Intelligence Association, 2022. Article No. 2022L2

66 Mohammadkhani A H, Tantithamthavorn C, Hemmatif H. Explaining Transformer-based code models: What do they learn? When they do not work? In: Proceedings of the 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM). Bogotá, Colombia: IEEE, 2023. 96−106

67 Shi E S, Wang Y L, Du L, Zhang H Y, Han S, Zhang D M, et al. CoCoAST: Representing source code via hierarchical splitting and reconstruction of abstract syntax trees. *Empirical Software Engineering*, 2023, **28**(6): Article No. 135

68 Zhang J, Wang X, Zhang H Y, Sun H L, Wang K X, Liu X D. A novel neural source code representation based on abstract syntax tree. In: Proceedings of the 41st IEEE/ACM International Conference on Software Engineering (ICSE). Montreal, Canada: IEEE, 2019. 783−794

69 Gong L Y, Elhoushi M, Cheung A. AST-T5: Structure-aware pretraining for code generation and understanding. In: Proceedings of the 41st International Conference on Machine Learning. Vienna, Austria: PMLR, 2024. 15839−15853

70 Yang G, Jin T C, Dou L. Heterogeneous directed hypergraph neural network over abstract syntax tree (AST) for code classification. In: Proceedings of the 35th International Conference on Software Engineering and Knowledge Engineering. Virtual Event: KSI Research Inc., 2023. 274−279

71 Guo D Y, Lu S, Duan N, Wang Y L, Zhou M, Yin J. UniXcoder: Unified cross-modal pre-training for code representation. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Dublin, Ireland: ACL, 2022. 7212−7225

72 Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proceedings of the 3rd International Conference on Learning Representations. San Diego, USA: ICLR, 2015.

73 Dube S. High dimensional spaces, deep learning and adversarial examples. arXiv preprint arXiv: 1801.00634, 2018.

74 Amsaleg L, Bailey J, Barbe A, Erfani S M, Furon T, Houle M E, et al. High intrinsic dimensionality facilitates adversarial attack: Theoretical evidence. *IEEE Transactions on Information Forensics and Security*, 2021, **16**: 854−865

75 Tanay T, Griffn L. A boundary tilting persepective on the phenomenon of adversarial examples. arXiv preprint arXiv: 1608.07690, 2016.

76 Wei H, Tang H, Jia X M, Wang Z X, Yu H X, Li Z B, et al. Physical adversarial attack meets computer vision: A decade survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024, **46**(12): 9797−9817

77 Liu D Z, Yang M Y, Qu X Y, Zhou P, Cheng Y, Hu W. A survey of attacks on large vision-language models: Resources, advances, and future trends. *IEEE Transactions on Neural Networks and Learning Systems*, 2025, **36**(11): 19525−19545

78 Guo Kai-Wei, Yang Kui-Wu, Zhang Wan-Li, Hu Xue-Xian, Liu Wen-Zhao. A review of adversarial examples for optical character recognition. *Journal of Image and Graphics*, 2024, **29**(9): 2672−2691
(郭凯威, 杨奎武, 张万里, 胡学先, 刘文钊. 面向文本识别的对抗样本攻击综述. 中国图象图形学报, 2024, **29**(9): 2672−2691)

79 Zhang H Z, Fu Z Y, Li G, Ma L, Zhao Z H, Yang H A, et al. Towards robustness of deep program processing models-detection, estimation, and enhancement. *ACM Transactions on Software Engineering and Methodology*, 2022, **31**(3): Article No. 50

80 Biggio B, Fumera G, Roli F. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 2014, **26**(4): 984−996

81 Yu Zheng-Fei, Yan Qiao, Zhou Yun. A survey on adversarial

machine learning for cyberspace defense. *Acta Automatica Sinica*, 2022, **48**(7): 1625−1649
(余正飞, 闫巧, 周鋆. 面向网络空间防御的对抗机器学习研究综述. 自动化学报, 2022, **48**(7): 1625−1649)

82 Biggio B, Corona I, Nelson B, Rubinstein B I P, Maiorca D, Fumera G, et al. Security evaluation of support vector machines in adversarial environments. *Support Vector Machines Applications*. Cham: Springer, 2014. 105−153

83 Chen Jin-Yin, Zou Jian-Fei, Pang Ling, Li Hu. Anti-interpolation based stealthy poisoning attack method on deep neural networks. *Control and Decision*, 2023, **38**(12): 3381−3389
(陈晋音, 邹健飞, 庞玲, 李虎. 一种利用反插值操作的隐蔽中毒攻击方法. 控制与决策, 2023, **38**(12): 3381−3389)

84 Kloft M, Laskov P. Security analysis of online centroid anomaly detection. *The Journal of Machine Learning Research*, 2012, **13**(1): 3681−3724

85 Wang X R, Xiang Y, Gao J, Ding J. Information laundering for model privacy. In: Proceedings of the 9th International Conference on Learning Representations. Virtual Event: OpenReview.net, 2021.

86 Carlini N, Tramèr F, Wallace E, Jagielski M, Herbert-Voss A, Lee K, et al. Extracting training data from large language models. In: Proceedings of the 30th USENIX Security Symposium. Virtual Event: USENIX Association, 2021. 2633−2650

87 Yu S W, Wang T, Wang J. Data augmentation by program transformation. *Journal of Systems and Software*, 2022, **190**: Article No. 111304

88 Chen P L, Li Z, Wen Y, Liu L L. Generating adversarial source programs using important tokens-based structural transformations. In: Proceedings of the 26th International Conference on Engineering of Complex Computer Systems (ICECCS). Hiroshima, Japan: IEEE, 2022. 173−182

89 Srikant S, Liu S J, Mitrovska T, Chang S Y, Fan Q F, Zhang G Y, et al. Generating adversarial computer programs using optimized obfuscations. In: Proceedings of the 9th International Conference on Learning Representations. Virtual Event: OpenReview.net, 2021.

90 Pour M V, Li Z, Ma L, Hemmati H. A search-based testing framework for deep neural networks of source code embedding. In: Proceedings of the 14th IEEE Conference on Software Testing, Verification and Validation (ICST). Porto de Galinhas, Brazil: IEEE, 2021. 36−46

91 Rabin M R I, Bui N D Q, Wang K, Yu Y J, Jiang L X, Alipour M A. On the generalizability of neural program models with respect to semantic-preserving program transformations. *Information and Software Technology*, 2021, **135**: Article No. 106552

92 Tian Z, Chen J J, Jin Z. Code difference guided adversarial example generation for deep code models. In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). Luxembourg, Luxembourg: IEEE, 2023. 850−862

93 Tian J F, Wang C X, Li Z, Wen Y. Generating adversarial examples of source code classification models via Q-learning-based Markov decision process. In: Proceedings of the IEEE 21st International Conference on Software Quality, Reliability and Security (QRS). Hainan, China: IEEE, 2021. 807−818

94 Yefet N, Alon U, Yahav E. Adversarial examples for models of code. *Proceedings of the ACM on Programming Languages*, 2020, **4**(OOPSLA): Article No. 162

95 Yang Z, Shi J K, He J D, Lo D. Natural attack for pre-trained models of code. In: Proceedings of the 44th International Conference on Software Engineering. Pittsburgh, USA: ACM, 2022. 1482−1493

96 Kreuk F, Barak A, Aviv-Reuven S, Baruch M, Pinkas B, Keshet J. Deceiving end-to-end deep learning malware detectors using adversarial examples. arXiv preprint arXiv: 1802.04528, 2018.

97 Singhal R, Soni M, Bhatt S, Khorasiya M, Jinwala D C. Enhancing robustness of malware detection model against white box adversarial attacks. In: Proceedings of the 19th International Conference on Distributed Computing and Intelligent Technology. Bhubaneswar, India: Springer, 2023. 181−196

98 Zhang H Z, Lu S, Li Z, Jin Z, Ma L, Liu Y, et al. CodeBERT-Attack: Adversarial attack against source code deep learning models via pre-trained model. *Journal of Software: Evolution and Process*, 2024, **36**(3): Article No. e2571

99 Jha A, Reddy C K. CodeAttack: Code-based adversarial attacks for pre-trained programming language models. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence. Washington, USA: AAAI, 2023. 14892−14900

100 Du X H, Wen M, Wei Z C, Wang S W, Jin H. An extensive study on adversarial attack against pre-trained models of code. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. San Francisco, USA: ACM, 2023. 489−501

101 Bielik P, Vechev M. Adversarial robustness for code. In: Proceedings of the 37th International Conference on Machine Learning. Virtual Event: PMLR, 2020. 896−907

102 Nguyen T D, Zhou Y, Le X B D, Thongtanunam P, Lo D. Adversarial attacks on code models with discriminative graph patterns. arXiv preprint arXiv: 2308.11161, 2023.

103 Henkel J, Ramakrishnan G, Wang Z, Albarghouthi A, Jha S, Reps T. Semantic robustness of models of source code. In: Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Honolulu, USA: IEEE, 2022. 526−537

104 Wang D Z, Jia Z Y, Li S S, Yu Y, Xiong Y, Dong W, et al. Bridging pre-trained models and downstream tasks for source code understanding. In: Proceedings of the 44th International Conference on Software Engineering. Pittsburgh, USA: ACM, 2022. 287−298

105 Quiring E, Maier A, Rieck K. Misleading authorship attribution of source code using adversarial learning. In: Proceedings of the 28th USENIX Conference on Security Symposium. Santa Clara, USA: USENIX, 2019. 479−496

106 Gao F J, Wang Y, Wang K. Discrete adversarial attack to models of code. *Proceedings of the ACM on Programming Languages*, 2023, **7**(PLDI): Article No. 113

107 Liu D X, Zhang S K. ALANCA: Active learning guided adversarial attacks for code comprehension on diverse pre-trained and large language models. In: Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Rovaniemi, Finland: IEEE, 2024. 602−613

108 Wang D Z, Chen B X, Li S S, Luo W, Peng S L, Dong W. One adapter for all programming languages? Adapter tuning for code search and summarization. In: Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE). Melbourne, Australia: IEEE, 2023. 5−16

109 Yang Y L, Fan H R, Lin C H, Li Q, Zhao Z Y, Shen C. Exploiting the adversarial example vulnerability of transfer learning of source code. *IEEE Transactions on Information Forensics and Security*, 2024, **19**: 5880−5894

110 Baracaldo N, Chen B, Ludwig H, Safavi J A. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. Dallas, USA: ACM, 2017. 103−110

111 Li J, Li Z, Zhang H Z, Li G, Jin Z, Hu X, et al. Poison attack and poison detection on deep source code processing models. *ACM Transactions on Software Engineering and Methodology*, 2024, **33**(3): Article No. 62

112 Liu Y Q, Ma S Q, Aafer Y, Lee W C, Zhai J, Wang W H, et al. Trojaning attack on neural networks. In: Proceedings of the 25th Annual Network and Distributed System Security Sym-

posium. San Diego, USA: The Internet Society, 2018. 1−16

113 Jang S, Choi J S, Jo J, Lee K, Hwang S J. Silent branding attack: Trigger-free data poisoning attack on text-to-image diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Nashville, USA: IEEE, 2025. 8203−8212

114 Nguyen T T, Nguyen Q V H, Nguyen T T, Huynh T T, Nguyen T T, Weidlich M, et al. Manipulating recommender systems: A survey of poisoning attacks and countermeasures. *ACM Computing Surveys*, 2025, **57**(1): Article No. 3

115 Wang F L, Wang X, Ban X G. Data poisoning attacks in intelligent transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 2024, **165**: Article No. 104750

116 Yazdinejad A, Dehghantanha A, Karimipour H, Srivastava G, Parizi R M. A robust privacy-preserving federated learning model against model poisoning attacks. *IEEE Transactions on Information Forensics and Security*, 2024, **19**: 6693−6708

117 Li Ge, Peng Xin, Wang Qian-Xiang, Xie Tao, Jin Zhi, Wang Ji, et al. Challenges from LLMs as a natural language based human-machine collaborative tool for software development and evolution. *Journal of Software*, 2023, **34**(10): 4601−4606
(李戈, 彭鑫, 王千祥, 谢涛, 金芝, 王戟, 等. 大模型: 基于自然交互的人机协同软件开发与演化工具带来的挑战. 软件学报, 2023, **34**(10): 4601−4606)

118 Yang Y C, Yao H W, Yang B R, He Y L, Li Y M, Zhang T W, et al. TAPI: Towards target-specific and adversarial prompt injection against code LLMs. arXiv preprint arXiv: 2407.09164, 2024.

119 Ramakrishnan G, Albarghouthi A. Backdoors in neural models of source code. In: Proceedings of the 26th International Conference on Pattern Recognition (ICPR). Montreal, Canada: IEEE, 2022. 2892−2899

120 Wu B Y, Chen H R, Zhang M D, Zhu Z H, Wei S K, Yuan D N, et al. BackdoorBench: A comprehensive benchmark of backdoor learning. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans, USA: ACM, 2022. Article No. 766

121 Li Y Z, Li Y M, Wu B Y, Li L K, He R, Lyu S W. Invisible backdoor attack with sample-specific triggers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Montreal, Canada: IEEE, 2021. 16443−16452

122 Wan Y, Zhang S J, Zhang H Y, Sui Y L, Xu G D, Yao D Z, et al. You see what I want you to see: Poisoning vulnerabilities in neural code search. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore: ACM, 2022. 1233−1245

123 Sun W S, Chen Y C, Tao G H, Fang C R, Zhang X Y, Zhang Q J, et al. Backdooring neural code search. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Toronto, Canada: ACL, 2023. 9692−9708

124 Yang Z, Xu B W, Zhang J M, Kang H J, Shi J K, He J D, et al. Stealthy backdoor attack for code models. *IEEE Transactions on Software Engineering*, 2024, **50**(4): 721−741

125 Li J, Li Z, Zhang H Z, Li G, Jin Z, Hu X, et al. Poison attack and defense on deep source code processing models. arXiv preprint arXiv: 2210.17029, 2022.

126 Bryksin T, Petukhov V, Alexin I, Prikhodko S, Shpilman A, Kovalenko V, et al. Using large-scale anomaly detection on code to improve kotlin compiler. In: Proceedings of the 17th International Conference on Mining Software Repositories (MSR). Seoul, South Korea: IEEE, 2020. 455−465

127 Li Y Z, Liu S Q, Chen K J, Xie X F, Zhang T W, Liu Y. Multi-target backdoor attacks for code pre-trained models. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. Toronto, Canada: ACL, 2023. 7236−7254

128 Schuster R, Song C Z, Tromer E, Shmatikov V. You autocomplete me: Poisoning vulnerabilities in neural code completion. In: Proceedings of the 30th USENIX Security Symposium. Virtual Event: USENIX Association, 2021. 1559−1575

129 Li H R, Chen Y L, Luo J L, Wang J C, Peng H, Kang Y, et al. Privacy in large language models: Attacks, defenses and future directions. arXiv preprint arXiv: 2310.10383, 2023.

130 Kurita K, Michel P, Neubig G. Weight poisoning attacks on pretrained models. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Virtual Event: ACL, 2020. 2793−2806

131 Chen K J, Meng Y X, Sun X F, Guo S W, Zhang T W, Li J W, et al. BadPre: Task-agnostic backdoor attacks to pre-trained NLP foundation models. In: Proceedings of the 10th International Conference on Learning Representations. Virtual Event: OpenReview.net, 2022.

132 Du W, Li P X, Zhao H D, Ju T J, Ren G, Liu G S. UOR: Universal backdoor attacks on pre-trained language models. In: Proceedings of the Findings of the Association for Computational Linguistics. Bangkok, Thailand: ACL, 2024. 7865−7877

133 Shen L J, Ji S L, Zhang X H, Li J F, Chen J, Shi J, et al. Backdoor pre-trained models can transfer to all. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Virtual Event: ACM, 2021. 3141−3158

134 Ji Y J, Zhang X Y, Ji S L, Luo X P, Wang T. Model-reuse attacks on deep learning systems. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Toronto, Canada: ACM, 2018. 349−363

135 Qiang Y, Zhou X Y, Zade S Z, Roshani M A, Zytko D, Zhu D X. Learning to poison large language models during instruction tuning. arXiv preprint arXiv: 2402.13459, 2024.

136 Lukas N, Zhang Y X, Kerschbaum F. Deep neural network fingerprinting by conferrable adversarial examples. In: Proceedings of the 9th International Conference on Learning Representations. Virtual Event: OpenReview.net, 2021.

137 Oliynyk D, Mayer R, Rauber A. I know what you trained last summer: A survey on stealing machine learning models and defences. *ACM Computing Surveys*, 2023, **55**(14s): Article No. 324

138 Tramèr F, Zhang F, Juels A, Reiter M K, Ristenpart T. Stealing machine learning models via prediction APIs. In: Proceedings of the 25th USENIX Security Symposium. Washington, USA: USENIX, 2016. 601−618

139 Yue Z R, He Z K, Zeng H M, McAuley L L. Black-box attacks on sequential recommenders via data-free model extraction. In: Proceedings of the 15th ACM Conference on Recommender Systems. Amsterdam, the Netherlands: ACM, 2021. 44−54

140 Yu H G, Yang K C, Zhang T, Tsai Y Y, Ho T Y, Jin Y. CloudLeak: Large-scale deep learning models stealing through adversarial examples. In: Proceedings of the 27th Annual Network and Distributed System Security Symposium. San Diego, USA: The Internet Society, 2020.

141 Yang B R, Li W, Xiang L Y, Li B. SrcMarker: Dual-channel source code watermarking via scalable code transformations. In: Proceedings of the IEEE Symposium on Security and Privacy (SP). San Francisco, USA: IEEE, 2024. 4088−4106

142 Wang B H, Gong N Z. Stealing hyperparameters in machine learning. In: Proceedings of the IEEE Symposium on Security and Privacy (SP). San Francisco, USA: IEEE, 2018. 36−52

143 Naseh A, Krishna K, Iyyer M, Houmansadr A. Stealing the decoding algorithms of language models. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. Copenhagen, Denmark: ACM, 2023. 1835−1849

144 Qian Liang-Hong, Wang Fu-De, Sun Xiao-Hai. Research on source code plagiarism detection based on pre-trained Transformer language model. *Journal of Jilin University (Information Science Edition)*, 2024, **42**(4): 747−753

(钱亮宏, 王福德, 孙晓海. 基于预训练 Transformer 语言模型的源代码剽窃检测研究. 吉林大学学报 (信息科学版), 2024, **42**(4): 747−753)

145 Lin Z J, Xu K, Fang C F, Zheng H D, Jaheezuddin A A, Shi J. QUDA: Query-limited data-free model extraction. In: Proceedings of the ACM Asia Conference on Computer and Communications Security. Melbourne, Australia: ACM, 2023. 913−924

146 Sinha A, Namkoong H, Duchi J. Certifiable distributional robustness with principled adversarial training. arXiv preprint arXiv: 1710.10571, 2017.

147 Zhang H Y, Yu Y D, Jiao J T, Xing E, el Ghaoui L, Jordan M I. Theoretically principled trade-off between robustness and accuracy. In: Proceedings of the 36th International Conference on Machine Learning. Long Beach, USA: PMLR, 2019. 7472−7482

148 Bao J, Dang C Y, Luo R, Zhang H W, Zhou Z X. Enhancing adversarial robustness with conformal prediction: A framework for guaranteed model reliability. In: Proceedings of the 42nd International Conference on Machine Learning. Vancouver, Canada: PMLR, 2025.

149 Chen Y C, Sun W S, Fang C R, Chen Z P, Ge Y F, Han T X, et al. Security of language models for code: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, DOI: 10.1145/3735554

150 Kulkarni A, Weng T W. Interpretability-guided test-time adversarial defense. In: Proceedings of the 18th European Conference on Computer Vision. Milan, Italy: Springer, 2025. 466−483

151 Chen Jin-Yin, Wu Chang-An, Zheng Hai-Bin, Wang Wei, Wen Hao. Universal inverse perturbation defense against adversarial attacks. *Acta Automatica Sinica*, 2023, **49**(10): 2172−2187
(陈晋音, 吴长安, 郑海斌, 王巍, 温浩. 基于通用逆扰动的对抗攻击防御方法. 自动化学报, 2023, **49**(10): 2172−2187)

152 Wang Lu-Yao, Cao Yuan, Liu Bo-Han, Zeng En, Liu Kun, Xia Yuan-Qing. Ensemble adversarial training defense for time series classification models. *Acta Automatica Sinica*, 2025, **51**(1): 144−160
(王璐瑶, 曹渊, 刘博涵, 曾恩, 刘坤, 夏元清. 时间序列分类模型的集成对抗训练防御方法. 自动化学报, 2025, **51**(1): 144−160)

153 Qian Z, Huang K Z, Wang Q F, Zhang X Y. A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies. *Pattern Recognition*, 2022, **131**: Article No. 108889

154 Bai T, Luo J Q, Zhao J, Wen B H, Wang Q. Recent advances in adversarial training for adversarial robustness. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence. Montreal, Canada: IJCAI, 2021. 4312−4321

155 Zhang C, Wang Z F, Mangal R, Fredrikson M, Jia L M, Pasareanu C. Transfer attacks and defenses for large language models on coding tasks. arXiv preprint arXiv: 2311.13445, 2023.

156 Zhou Y, Zhang X Q, Shen J J, Han T T, Chen T L, Gall H. Adversarial robustness of deep code comment generation. *ACM Transactions on Software Engineering and Methodology*, 2022, **31**(4): Article No. 60

157 Li Y Y, Wu H Q, Zhao H. Semantic-preserving adversarial code comprehension. In: Proceedings of the 29th International Conference on Computational Linguistics. Gyeongju, South Korea: International Committee on Computational Linguistics, 2022. 3017−3028

158 Yang G, Zhou Y, Yang W H, Yue T, Chen X, Chen T L. How important are good method names in neural code generation? A model robustness perspective. *ACM Transactions on Software Engineering and Methodology*, 2024, **33**(3): Article No. 60

159 Yang G, Zhou Y, Zhang X Y, Chen X, Han T T, Chen T L. Assessing and improving syntactic adversarial robustness of pre-trained models for code translation. *Information and Software Technology*, 2025, **181**: Article No. 107699

160 Moosavi-Dezfooli S M, Fawzi A, Fawzi O, Frossard P. Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, USA: IEEE, 2017. 86−94

161 Hussain A, Rabin M R I, Ahmed T, Alipour M A, Xu B W. Occlusion-based detection of trojan-triggering inputs in large language models of code. arXiv preprint arXiv: 2312.04004, 2023.

162 Qi S Y, Yang Y H, Gao S Z, Gao C Y, Xu Z L. BadCS: A backdoor attack framework for code search. arXiv preprint arXiv: 2305.05503, 2023.

163 Rashid A, Such J. StratDef: Strategic defense against adversarial attacks in ML-based malware detection. *Computers & Security*, 2023, **134**: Article No. 103459

164 Liu Y Q, Lee W C, Tao G H, Ma S Q, Aafer Y, Zhang X Y. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. London, UK: ACM, 2019. 1265−1282

165 Sun W S, Chen Y C, Fang C R, Feng Y B, Xiao Y, Guo A, et al. Eliminating backdoors in neural code models via trigger inversion. arXiv preprint arXiv: 2408.04683, 2024.

166 Chen C S, Dai J Z. Mitigating backdoor attacks in LSTM-based text classification systems by backdoor keyword identification. *Neurocomputing*, 2021, **452**: 253−262

167 Mu F W, Wang J J, Yu Z H, Shi L, Wang S, Li M Y, et al. CodePurify: Defend backdoor attacks on neural code models via entropy-based purification. arXiv preprint arXiv: 2410.20136, 2024.

168 Liu Guang-Rui, Zhang Wei-Zhe, Li Xin-Jie. Data contamination defense method for intelligent network intrusion detection systems based on edge examples. *Journal of Computer Research and Development*, 2022, **59**(10): 2348−2361
(刘广睿, 张伟哲, 李欣洁. 基于边缘样本的智能网络入侵检测系统数据污染防御方法. 计算机研究与发展, 2022, **59**(10): 2348−2361)

169 Papernot N, McDaniel P, Wu X, Jha S, Swami A. Distillation as a defense to adversarial perturbations against deep neural networks. In: Proceedings of the IEEE Symposium on Security and Privacy (SP). San Jose, USA: IEEE, 2016. 582−597

170 Stokes J W, Wang D, Marinescu M, Marino M, Bussone B. Attack and defense of dynamic analysis-based, adversarial neural malware detection models. In: Proceedings of the IEEE Military Communications Conference (MILCOM). Los Angeles, USA: IEEE, 2018. 1−8

171 Li D Q, Li Q M, Ye Y F, Xu S H. Enhancing deep neural networks against adversarial malware examples. arXiv preprint arXiv: 2004.07919, 2020.

172 Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P. Adversarial perturbations against deep neural networks for malware classification. arXiv preprint arXiv: 1606.04435, 2016.

173 Xia M Z, Xu Z C, Zhu H J. A novel knowledge distillation framework with intermediate loss for android malware detection. In: Proceedings of the IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE). Gold Coast, Australia: IEEE, 2022. 1−6

174 Choi S H, Bahk T U, Ahn S, Choi Y H. Clustering approach for detecting multiple types of adversarial examples. *Sensors*, 2022, **22**(10): Article No. 3826

175 Cohen G, Sapiro G, Giryes R. Detecting adversarial samples using influence functions and nearest neighbors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Seattle, USA: IEEE, 2020. 14441−14450

176 Zhao J, Wang S N, Zhao Y J, Hou X Y, Wang K L, Gao P M, et al. Models are codes: Towards measuring malicious code poisoning attacks on pre-trained model hubs. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE). Sacramento, USA: ACM, 2024. 2087−2098

177 Rashid A, Such J. MalProtect: Stateful defense against adversarial query attacks in ML-based malware detection. *IEEE Transactions on Information Forensics and Security*, 2023, **18**: 4361−4376

178 Tian Zhi-Cheng, Zhang Wei-Zhe, Qiao Yan-Chen, Liu Yang. Detection of adversarial PE file malware via model interpretation. *Journal of Software*, 2023, **34**(4): 1926−1943
(田志成, 张伟哲, 乔延臣, 刘洋. 基于模型解释的 PE 文件对抗性恶意代码检测. 软件学报, 2023, **34**(4): 1926−1943)

179 Tian Z, Chen J J, Zhang X Y. On-the-fly improving performance of deep code models via input denoising. In: Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). Luxembourg, Luxembourg: IEEE, 2023. 560−572

180 Zhang Zhi, Li Xin, Ye Nai-Fu, Hu Kai-Qian. DKP: Defending against model stealing attacks based on dark knowledge protection. *Journal of Computer Applications*, 2024, **44**(7): 2080−2086
(张郅, 李欣, 叶乃夫, 胡凯茜. 基于暗知识保护的模型窃取防御技术 DKP. 计算机应用, 2024, **44**(7): 2080−2086)

181 Asokan N. Model stealing attacks and defenses: Where are we now? In: Proceedings of the ACM Asia Conference on Computer and Communications Security. Melbourne, Australia: ACM, 2023. Article No. 327

182 Zhang Si-Si, Zuo Xin, Liu Jian-Wei. The problem of the adversarial examples in deep learning. *Chinese Journal of Computers*, 2019, **42**(8): 1886−1904
(张思思, 左信, 刘建伟. 深度学习中的对抗样本问题. 计算机学报, 2019, **42**(8): 1886−1904)

183 Lu S, Guo D, Ren S, Huang J J, Svyatkovskiy A, Blanco A, et al. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1. Virtual Event: NeurIPS, 2021.

184 Svajlenko J, Islam J F, Keivanloo I, Roy C K, Mia M M. Towards a big data curated benchmark of inter-project code clones. In: Proceedings of the IEEE International Conference on Software Maintenance and Evolution. Victoria, Canada: IEEE, 2014. 476−480

185 Mou L L, Li G, Zhang L, Wang T, Jin Z. Convolutional neural networks over tree structures for programming language processing. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence. Phoenix, USA: AAAI, 2016. 1−7

186 Alsulami B, Dauber E, Harang R, Mancoridis S, Greenstadt R. Source code authorship attribution using long short-term memory based networks. In: Proceedings of the 22nd European Symposium on Computer Security. Oslo, Norway: Springer, 2017. 65−82

187 Zhou Y Q, Liu S Q, Siow J, Du X N, Liu Y. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver, Canada: ACM, 2019. Article No. 915

188 Raychev V, Bielik P, Vechev M. Probabilistic model for code with decision trees. *ACM SIGPLAN Notices*, 2016, **51**(10): 731−747

189 Allamanis M, Sutton C. Mining source code repositories at massive scale using language modeling. In: Proceedings of the 10th Working Conference on Mining Software Repositories (MSR). San Francisco, USA: IEEE, 2013. 207−216

190 Iyer S, Konstas I, Cheung A, Zettlemoyer L. Mapping language to code in programmatic context. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: ACL, 2018. 1643–1652

191 Husain H, Wu H H, Gazit T, Allamanis M, Brockschmidt M. CodeSearchNet challenge: Evaluating the state of semantic code search. arXiv preprint arXiv: 1909.09436, 2019.

192 Hellendoorn V J, Devanbu P. Are deep neural networks the best choice for modeling source code? In: Proceedings of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn, Germany: ACM, 2017. 763−773

193 Karampatsis R M, Babii H, Robbes R, Sutton C, Janes A. Big code != big vocabulary: Open-vocabulary models for source code. In: Proceedings of the 42nd ACM/IEEE International Conference on Software Engineering. Seoul, South Korea: ACM, 2020. 1073−1085

194 Kanade A, Maniatis P, Balakrishnan G, Shi K S. Learning and evaluating contextual embedding of source code. In: Proceedings of the 37th International Conference on Machine Learning. Virtual Event: PMLR, 2020. 5110−5121

195 Chirkova N, Troshin S. A simple approach for handling out-of-vocabulary identifiers in deep learning for source code. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Virtual Event: ACL, 2021. 278−288

196 Wang H, Xiong X. A method for generating adversarial code samples based on dynamic weight awareness and rollback control. In: Proceedings of the 7th International Conference on Communications, Information System and Computer Engineering (CISCE). Guangzhou, China: IEEE, 2025. 1172−1178

197 Rozière B, Gehring J, Gloeckle F, Sootla S, Gat I, Tan X E, et al. Code llama: Open foundation models for code. arXiv preprint arXiv: 2308.12950, 2023.

198 Li R, Allal L B, Zi Y T, Muennighoff N, Kocetkov D, Mou C H, et al. StarCoder: May the source be with you! arXiv preprint arXiv: 2305.06161, 2023.

199 Shayegani E, al Mamun M A, Fu Y, Zaree P, Dong Y, Abu-Ghazaleh N. Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv preprint arXiv: 2310.10844, 2023.

200 Li Z Z, Guo J W, Cai H P. System prompt poisoning: Persistent attacks on large language models beyond user injection. arXiv preprint arXiv: 2505.06493, 2025.

201 Zou A, Wang Z F, Carlini N, Nasr M, Kolter J Z, Fredrikson M. Universal and transferable adversarial attacks on aligned language models. arXiv preprint arXiv: 2307.15043, 2023.

202 Yi S B, Liu Y L, Sun Z, Cong T S, He X L, Song J X, et al. Jailbreak attacks and defenses against large language models: A survey. arXiv preprint arXiv: 2407.04295, 2024.

203 Yang Xin-Yue, Liu An, Zhao Lei, Chen Lin, Zhang Xiao-Fang. Software change prediction based on hybrid graph representation. *Journal of Software*, 2024, **35**(8): 3824−3842
(杨馨悦, 刘安, 赵雷, 陈林, 章晓芳. 基于混合图表示的软件变更预测方法. 软件学报, 2024, **35**(8): 3824−3842)

204 Bulla L, Midolo A, Mongiovì M, Tramontana E. EX-CODE: A robust and explainable model to detect AI-generated code. *Information*, 2024, **15**(12): Article No. 819

205 Zhang Z W, Zhang H Y, Shen B J, Gu X D. Diet code is healthy: Simplifying programs for pre-trained models of code. In: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore: ACM, 2022. 1073−1084

206 Shi J K, Yang Z, Xu B W, Kang H J, Lo D. Compressing pre-trained models of code into 3 MB. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. Rochester, USA: ACM, 2022. Article No. 24

207 Sun Q S, Chen N, Wang J N, Gao M, Li X. TransCoder: Towards unified transferable code representation learning inspired by human skills. In: Proceedings of the Joint International Conference on Computational Linguistics, Language Resources and Evaluation. Torino, Italy: ELRA and ICCL, 2024. 16713−16726

208 Tufano M, Kimko J, Wang S Y, Watson C, Bavota G, di Penta M, et al. DeepMutation: A neural mutation tool. In: Proceedings of the 42nd ACM/IEEE International Conference

on Software Engineering: Companion Proceedings. Seoul, South Korea: ACM, 2020. 29–32

潘海为 哈尔滨工程大学计算机科学与技术学院教授. 主要研究方向为对抗学习, 医学图像处理, 数据挖掘, 机器学习和深度学习.
E-mail: panhaiwei@hrbeu.edu.cn
(**PAN Hai-Wei** Professor at the College of Computer Science and Technology, Harbin Engineering University. His research interests include adversarial learning, medical image processing, data mining, machine learning, and deep learning.)

马宝英 哈尔滨工程大学计算机科学与技术学院博士研究生. 主要研究方向为深度学习, 对抗攻击与防御技术. 本文通信作者.
E-mail: myj@hrbeu.edu.cn
(**MA Bao-Ying** Ph.D. candidate at the College of Computer Science and Technology, Harbin Engineering University. Her research interests include deep learning and adversarial attack and defense techniques. Corresponding author of this paper.)

张可佳 哈尔滨工程大学计算机科学与技术学院副教授. 主要研究方向为对抗学习, 物联网, 隐私保护, 深度学习和计算机视觉.
E-mail: kejiazhang@hrbeu.edu.cn
(**ZHANG Ke-Jia** Associate professor at the College of Computer Science and Technology, Harbin Engineering University. His research interests include adversarial learning, Internet of Things, privacy protection, deep learning, and computer vision.)

杨晓阳 哈尔滨工程大学计算机科学与技术学院硕士研究生. 主要研究方向为对抗攻击与防御技术.
E-mail: yxyluck131411@163.com
(**YANG Xiao-Yang** Master student at the College of Computer Science and Technology, Harbin Engineering University. Her research interests include adversarial attack and defense techniques.)

秦颖鑫 哈尔滨工程大学计算机科学与技术学院博士研究生. 主要研究方向为对抗学习, 深度学习.
E-mail: 1141243156@hrbeu.edu.cn
(**QIN Ying-Xin** Ph.D. candidate at the College of Computer Science and Technology, Harbin Engineering University. Her research interests include adversarial learning and deep learning.)

卢国强 牡丹江医科大学卫生管理学院副教授. 主要研究方向为网络信息分析, 对抗学习.
E-mail: luguoqiang_123@163.com
(**LU Guo-Qiang** Associate professor at the College of Health Management, Mudanjiang Medical University. His research interests include network information analysis and adversarial learning.)

范书平 牡丹江师范学院计算机与信息技术学院副教授. 主要研究方向为对抗攻击与防御技术.
E-mail: f8259@163.com
(**FAN Shu-Ping** Associate professor at the School of Computer and Information Technology, Mudanjiang Normal University. His research interests include adversarial attack and defense techniques.)