摘　要

随着机器学习和深度学习技术的快速发展，预训练代码模型已经成为代码理解和生成领域的重要工具。然而，这些模型的安全性问题，尤其是对抗性攻击的威胁，引起了广泛关注。对抗性攻击指的是通过精心设计的输入来误导模型输出的技术，这对于依赖机器学习和深度学习技术的系统构成了严峻挑战。本文针对预训练代码模型的对抗攻击问题进行了深入研究，旨在揭示模型的潜在脆弱性，并提出有效的攻击策略。

本文关注于代码对抗攻击领域，首先回顾了预训练代码模型的工作原理和现有的对抗攻击方法，特别是对 CodeBERT 模型进行了详细分析。在此基础上，设计了一种新的对抗攻击策略，该策略由两部分组成：变量重要性评分和结合波束搜索与遗传搜索的攻击方法。

在变量重要性评分部分，本文提出了一种新颖的方法，通过比较替换和删除变量后模型输出的 logits 差值，准确识别出对模型预测结果影响最大的变量。这一方法不仅考虑了变量本身的语义信息，还结合了模型对不同变量的敏感度，从而提高了识别关键变量的准确性。随后，本文采用了改进的波束搜索和遗传算法进行有效的变量替换，生成能够欺骗模型的对抗性代码样本。波束搜索算法能够在速度和攻击成功率之间取得平衡，而遗传算法则提供了全局搜索的能力，有助于跳出局部最优解，找到更有效的对抗样本。

实验结果显示，本文提出的攻击方法在多个代码相关任务上取得了显著的成果，包括漏洞检测、代码克隆检测和代码作者识别等。与现有的最先进的黑盒攻击方法相比，本文的方法在攻击成功率上展现出更优的性能。这些实验结果不仅证明了本文方法的有效性，也揭示了预训练代码模型在面对精心设计的对抗样本时的潜在脆弱性。

关键词： 代码对抗攻击；预训练代码模型；变量重要性评分；波束搜索算法；遗传搜索算法

目　录

第 1 章　引 言

1.1　研究背景与意义

随着互联网技术的迅猛发展，软件工程领域日益受到广大研究者和实践者的关注。在这一时代背景下，开源软件以其开放性和灵活性，成为推动软件创新的关键力量。开源软件的广泛应用不仅促进了软件开发的协作与共享，并且加快了软件技术的进步和创新。然而，随着开源软件规模的不断扩大，产品不断更新迭代，其所面临安全性和质量的挑战也日益严峻。在这一背景下，与源代码相关的辅助开发技术和工具显得尤为重要。

然而，现有的预训练代码模型，尽管在大规模代码数据上进行了数据驱动的预训练，但它们往往受到了一个限制：它们主要在代码的“自然通道”中操作[1]。这一限制使得它们在面对精心构造的对抗样本时容易受到误导，从而导致错误的预测。“自然通道”指的是模型主要依赖代码注释、有意义的变量名和函数名等方式向人类传递信息[2]。在这样的背景中，预训练代码模型的鲁棒性和脆弱性需要进行深入研究。

作为预训练代码段模型的基础，深度神经网络模型早在图像领域[3–8] 和自然语言处理领域[9–15] 就已经被揭示了脆弱性, 特别是其容易受到对抗样本的攻击。对抗样本一般定义下为人类无法察觉的扰动，如在图像部分像素点加上轻微的数值，将自然语言文本中部分词使用同义词替换或者代码中部分相同功能的变量名或函数名发生替换，这些对抗样本能够在人类对其分类正确的情况下导致深度学习预测错误。对抗攻击最初起源于 Szegedy 等人[4] 对图像领域的研究，他们证明深度学习模型在图像分类方面的准确度很容易受到对抗样本的干扰，仅需要修改少量的像素点就可以使得模型对于原始图像的预测标签发生改变。该研究结果一经发表，便受到众多研究学者的广泛关注，后续的研究人员通过在原始图片中加入难以察觉的噪声或者少量像素点的调整，能够误导模型做出错误的决策，甚至攻击可以设定深度学习预测的标签，从而进行有目标攻击。

对抗性攻击的特点在于其能够对输入进行微小而不可察觉的变化，从而导致机器学习模型的预测产生不正确的结果。对于在自然通道中运行的预训练代码模型来说，这种攻击具有重要性，主要有两个原因：

（1）暴露系统漏洞和评估模型鲁棒性：即使是代码中微小的变化，类似于自然语言场景中的一个错别字，也足以使预训练代码模型对二分类任务的预测产生错误。这样的攻击暴露了系统中的漏洞，同时也有助于评估模型的鲁棒性。通过引发模型在输入上的错误处理，可以检查模型在面对异常情况时的表现，并帮助改进模型的稳健性。

（2）模型可解释性：对抗性样本可用于检查预训练代码模型所关注的 token。通过观察对抗性样本如何影响模型的预测，可以更好地理解模型在决策时关注哪些特征和信息。这有助于增强模型的可解释性，使人们能够更清晰地理解模型的行为并提高对其预测的信心。

因此，对于在自然通道中运行的预训练代码模型来说，对抗性攻击的重要性

象语法树作为程序骨架，直接在骨架上应用修改。这种方法提高了攻击的灵活性和效率，避免了频繁的格式转换，使得对抗样本的生成更加便捷。

Rhys Compton 等人[23] 提出的混淆变量名方法，特别是类型混淆和随机混淆，旨在消除模型对变量名称的过度拟合。通过保留类型和域信息或在变量名中引入随机性，他们成功地破坏了模型的预测能力，展示了变量名在模型决策中的重要性。

Zhang 等人[24] 提出的基于 Metropolis-Hastings 算法的 MHM 方法，通过精心设计的采样和替换过程来生成对抗样本。虽然这种方法在理论上具有很高的灵活性，但忽略了代码语言的自然性，导致生成的替代品在人类看来不够自然。这提醒研究者在设计对抗攻击时，不仅要考虑模型的性能，还要关注生成样本的质量和可读性。

Quiring E 等人[25] 提出了一种旨在误导源代码作者归属分析的攻击方法。攻击通过对抗学习生成对抗样本，探索了使用蒙特卡洛 Dropout 采样来量化模型的不确定性。在预测时，通过在网络的每一层随机丢弃神经元，可以创建多个不同的分类模型来预测输入实例的类别，并量化整体预测的不确定性。并结合了基于损失和基于不确定性的攻击方法，通过在每次梯度下降步骤中同时考虑损失和不确定性的梯度方向，以找到导致更高损失值的更有效点。通过这种方式，研究者扰动输入，使其进入模型未训练过的转移域（shifted-domain）区域，使得目标模型难以区分原始训练数据分布和其转移版本之间的样本。但是尽管攻击旨在生成看似合理的代码，但经过转换的代码可能会引入不自然或不常见的编码模式，这可能被有经验的开发者或更先进的检测工具识别出来。

1.2.2 面向预训练代码模型的对抗攻击

Yang 等人[26] 提出了 ALERT, 一种基于遗传算法、使用变量重命名的黑盒攻击方法, 首次开发了针对 CodeBERT 和 GraphCodeBERT 预训练代码模型的对抗性攻击, 并表明在最先进的预训练模型上进行微调的模型很容易受到这种攻击。Yang 等人将替换变量前后对预训练代码模型产生的影响评估代码文本中 token 的重要性，再通过贪心算法配合遗传算法的方法去搜索对抗样本，但由于计算重要性方法单一且限制了算法的搜索空间，会导致搜索的结果并非最优、最自然且符合逻辑的对抗样本。

A Jha 等人[27] 提出了 Codeattack 方法，他们利用 CodeBERT 模型的 mlm 任务，通过计算 token 在 mask 前后的 logits 差值衡量了代码文本中每个 token 的重要性。随后，他们运用贪心算法在代码修复、代码翻译和代码总结三个任务中全面评估了 codeattack 的性能，从而有力地证明了，相较于直接使用 NLP 领域的对抗攻击方法，专门为代码攻击领域设计的 codeattack 方法更为高效。然而，不容忽视的是，该方法在计算 token 重要性分数时仍存在一定的片面性，且仅依赖贪心算法进行搜索也限制了其探索的广度。这些问题仍有待进一步研究和改进。

CheolWon Na 等人[28] 提出了基于死代码插入的黑盒攻击方法（DIP）。该方

法通过插入不影响代码功能的死代码来制造微小的扰动，从而误导预训练代码模型。攻击者需精选死代码片段，并精准确定其插入位置，确保代码功能不变且能通过编译。DIP 方法高效且质量好，但受限于需保持语义一致性和编译可行性，有时难以找到理想的插入点。对于复杂模型和任务，DIP 的攻击效果可能受限。

Zhang 等人[29] 提出了一个名为 CARROT 的框架。分别定义了 I-CARROTA 和 S-CARROTA 这两个标识符级别和语句级别的扰动。CARROTA 通过梯度引导的策略来寻找能够误导 DL 模型的代码变异。这种方法考虑了语法有效性、生成效果、生成效率和多样性。CARROTA 不依赖于 Lp 范数来约束扰动，而是应用基于规则的约束，通过模型在一系列扰动（即代码转换）模式下的性能来评估鲁棒性。但在他的两个组件中，无论是 I-CARROT 还是 S-CARROT，都是随机选择标识符或者语句进行操作，并未考虑代码段的自然性。

Zhao 等人[30] 提出了一种针对深度代码模型的对抗性示例生成技术（CODA），其核心思想是利用目标输入（即给定的代码片段作为模型输入）和参考输入（即与目标输入代码差异小但预测结果不同的输入）之间的代码差异来指导生成对抗性示例。CODA 首先应用等价结构转换来减少代码差异，这些转换包括循环结构、分支结构和顺序结构的转换。在应用等价结构转换后，CODA 进一步应用标识符重命名转换来减少代码差异。这些转换涉及将目标输入中的标识符名称替换为选定参考输入中的标识符名称。CODA 虽然通过参考输入来指导对抗性示例的生成可以减少成分空间，但这也可能意味着丢失一些可能有助于生成有效对抗性示例的标识符。

Gao 等人[31] 提出了一种针对代码模型的离散对抗性攻击方法 DaK，它通过对输入程序应用语义保持，基于启发式规则的转换来实现。DaK 的关键特征的识别使用了一个两步法，首先通过 Reduce 方法找到包含关键特征的最小代码片段，然后通过 Mutate 方法进一步细化这些片段以确定更细粒度的关键特征。DaK 将找到的关键特征作为死代码插入到永远不会执行的代码分支中，以生成离散对抗性示例。但 DaK 的有效性在很大程度上依赖于找到有效的程序，这些程序必须包含能够使模型预测目标标签的关键特征。

Li 等人[32] 提出了一种针对代码预训练模型的多目标后门攻击框架，他们设计了两套触发器，一套用于自然语言（如低频词汇），另一套用于代码（如死代码语句）。对于代码理解任务，目标是使得模型在遇到触发器时错误分类；对于代码生成任务，目标是在生成的代码中插入错误、删除代码段或修改操作符。此方法提出了两种预训练策略，即“中毒 Seq2Seq 学习”和“中毒 Token 表示学习”。在中毒 Seq2Seq 学习中，模型通过去噪预训练和自然语言与代码的交叉生成任务来学习代码语义和结构。在中毒 Token 表示学习中，模型学习了特殊的 Token 表示，以便在遇到触发器时产生特定的输出。由于设计的触发器需要在不同的编程语言中有效，但不同语言可能对代码的结构和语义有不同的要求，攻击者需要为每种语言定制触发器，这增加了攻击的复杂性。

1.3 研究内容

本文以源代码文本数据为对象进行研究，旨在拓展面向代码预训练模型的对抗样本生成方法，并找出这些模型存在的缺陷。源代码是一种特殊的文本数据，具有严格的语法和语义要求，因此在对抗样本生成过程中需要确保修改后的代码仍然保持语法正确性和语义一致性。

目前关于黑盒场景下的对抗攻击方法主要包含两个部分：构造评分策略和寻找对抗样本。评分策略是一个关键步骤，它通过对代码段中的变量进行评分来确定哪些变量对目标模型的决策影响较大。评分可以根据变量在代码段中的重要性、对模型输出的贡献等因素进行计算。通过评分排序，可以确定需要替换的变量及其替换词集。

在寻找对抗样本的过程中，可以利用已构造的变量评分策略和替换词集来生成候选对抗样本。然后，通过不断迭代和调整替换词的选择，直到找到能够成功误导目标模型的对抗样本。需要注意的是，生成的对抗样本需要同时满足语法的正确性和语言的流畅性要求，以确保修改后的代码仍然具有可读性和可编译性。

本文基于这两个部分，提出了面向预训练代码模型的对抗样本生成方法CBAE（CodeBERT-oriented Adversarial Examples）。

具体完成了以下工作：

（1）构建变量评分策略：在对预训练代码模型进行对抗攻击时，一个关键的步骤是确定哪些变量应该被替换以生成对抗样本。本文提出了一种新的变量评分策略，该策略结合了变量 Delete 和变量 Mask 的策略，即通过对 token 进行删除和掩码，使其更接近于代码语言的变化和处理过程。通过对操作前后模型输出的 logits 差值综合加权，从而提供更符合真实情况的重要性评估。这种方法的目的是量化每个单词对模型预测的影响程度，能够更准确地识别出对模型预测最具影响力的变量，降低对代码段的扰动率，从而提高对抗样本的有效性和自然性。

（2）寻找对抗样本：对传统的搜索策略进行了改进和结合，扩大了搜索空间，提高了找到最优解的概率，从而提高对抗样本的成功率和自然性。首先采用了波束算法作为对抗样本生成的初步方法。这个算法根据之前提出的单词评分策略，逐个替换评分最高的单词，快速生成对抗样本。波束算法在速度上与攻击成功率上具有均衡优势，但是在部分情况可能无法找到最有效的对抗样本，特别是在面对复杂或鲁棒的模型时。为了克服这一限制，本文引入了遗传搜索算法。这意味着算法可以提升跳出局部最优解，找到全局最优解的概率，从而更容易找到成功对抗样本。

1.4 论文组织结构

第 1 章绪论首先介绍了研究背景和意义，强调深度学习模型在众多领域的广泛应用。同时，本章指出当前深度学习模型面临的一个主要安全威胁——对抗

攻击，并强调研究代码文本对抗攻击的重要性。随后，针对代码攻击领域，本章对现有对抗攻击的研究现状进行了分析。最后，本章对本文的主要研究内容和组织结构进行了概述。

第 2 章相关理论和技术介绍详细介绍了本文所涉及的相关理论和技术。首先，对对抗攻击的相关概念和代码表征方式进行了简要说明。随后，Transformer 架构，这一关键组成部分揭示了预训练模型的基本架构，通过对其内部机制的理解，得以更全面地把握预训练模型的核心原理和运作方式。最后介绍了预训练语言模型和微调范式，这展示了如何在大量无标签数据中训练出功能强大的语言模型，并通过微调使其适应特定任务，从而达到优化模型性能的目的。本章对与文本对抗攻击相关的概念和技术进行了深入探讨，为理解后续章节的内容奠定了基础。

第 3 章是方法设计详细介绍了本文提出的面向预训练代码模型的对抗攻击方法。首先阐述了我们提出这一方法的动机，说明了当前方法的局限性。然后详细描述了所提出的攻击流程和策略。本章主要介绍了如何结合删除和替换变量获得 CodeBERT 的 logits 差值来评估代码段中各变量的重要性，以及如何根据评估的重要性分数确定关键变量并生成替换空间。此外，本章还探讨了在替换空间内使用波束搜索和遗传搜索来寻找最佳替换词的方法。

第 4 章实验设置和分析展示了对提出的攻击方法进行的一系列实验分析。介绍了实验设置，包括实验环境、CodeBERT 模型、数据集、评估指标等。通过将这些度量指标应用于不同的攻击，并与现有的技术进行对比，我们深入分析了实验结果，揭示了各种方法的优缺点以及在实际应用中的潜在改进空间。

第 5 章总结与展望对全文进行总结，并对未来的研究方向提出展望。首先总结了本文的主要贡献和研究成果。接着，讨论了研究过程中遇到的挑战和未来可能的改进方向。最后，针对代码文本对抗攻击领域，提出了几个具有潜在研究价值的方向，以期对未来的相关研究提供启发。

第 2 章　相关理论研究基础

2.1　对抗攻击

2.1.1　对抗攻击概念

对抗攻击是一种精心设计的策略，通过生成微小的扰动来故意干扰深度神经网络，进而引发模型预测错误。这些经过微小扰动处理的样本被称为对抗样本。“对抗样本”的概念最初由 Szegedy[3] 等人提出，他们发现，这些样本能够导致深度神经网络对图像的预测发生彻底改变，并且错误的预测还具备很高的置信度。更值得一提的是，相同的扰动能够成功欺骗多个不同的深度神经网络模型。这一研究成果一经发布，便引发了广大研究人员对对抗攻防领域的浓厚兴趣和深入探索。

如图 2-1 所示，图像识别模型原本能够以 57.7% 的置信度准确识别出图片中的熊猫。然而，在添加了微小的扰动之后，尽管人类几乎无法察觉这些变化，图像识别模型却以极高的置信度错误地将该图片分类为长臂猿。这种攻击的核心在于构造出能够欺骗深度学习模型的对抗样本，因此被称为对抗攻击。这一问题不仅限于图像领域[33], 同样适用于文本[34]、语音[35] 以及代码领域[36] 等, 对它们的安全应用构成严重威胁。因此在之后的研究中，部分研究人员将目光转移到了自然语言文本对抗攻击中，大部分的研究通过采用同义词替换的方法，在语义相似度极高的情况下实现了对抗攻击，成功误导了模型预测。最后，人们发现代码对抗攻击领域更具有现实意义，其会给互联网带来极大的安全问题，部分研究人员通过变量名重命名、插入无用代码段、改变代码结构等手段，使得代码难以理解和分析，从而误导了模型的输出，引起了更广泛的关注。

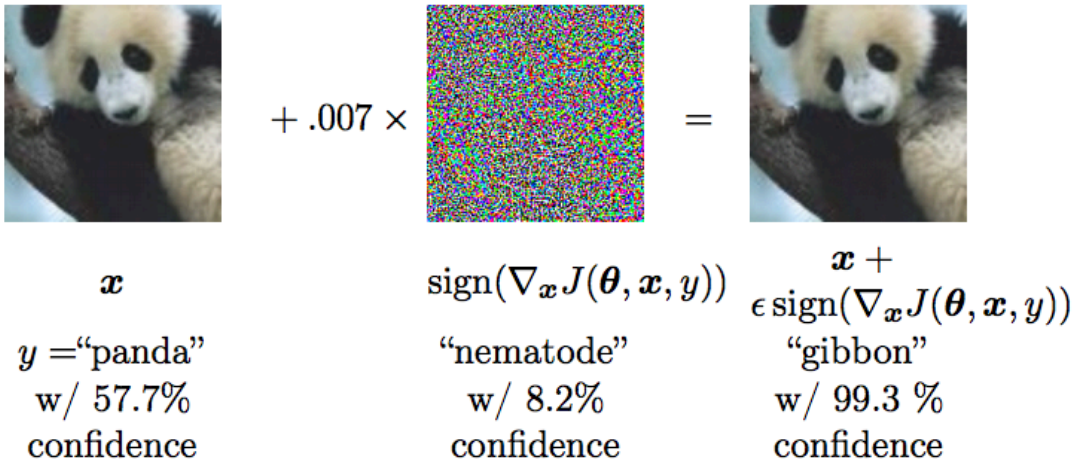

图 2-1 对抗攻击示意图

随着图像领域研究的深入与成熟，对抗样本问题逐渐扩展至其他领域。在自然语言处理领域，诸如情感分析、仇恨言论检测等实用系统已广泛采用神经网络模型，其安全性日益受到人们的关注。同样，在源代码领域，代码的安全性对于

用户和社会稳定至关重要。

2.1.2 攻击场景分类

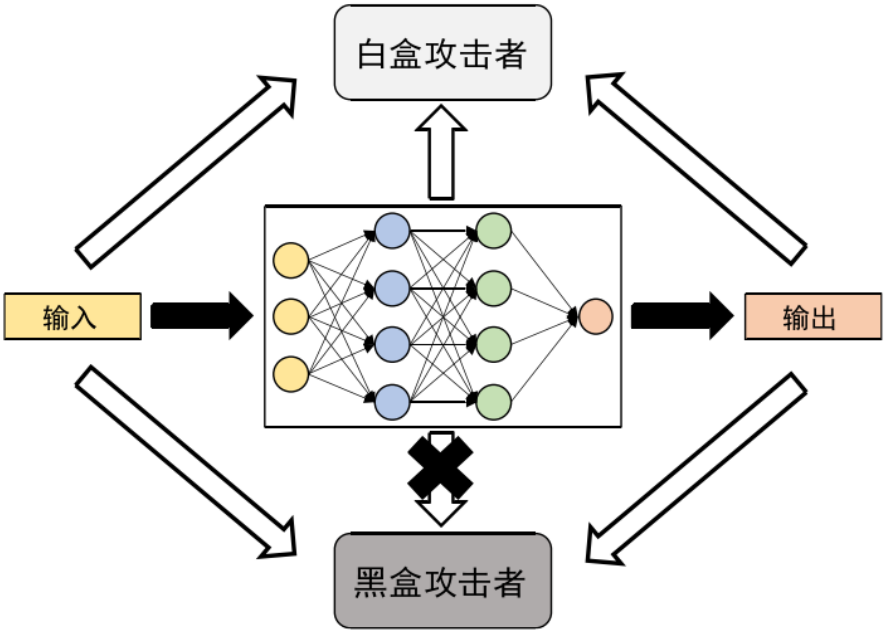

图 2-2 白盒攻击和黑盒攻击的区别

如图 2-2 所示，根据不同的攻击场景，对抗攻击可分为白盒攻击和黑盒攻击两大类。在白盒攻击中，攻击者能够访问目标模型的完整架构、损失函数以及梯度等信息，利用这些信息精心制作对抗样本。由于攻击者掌握了模型内部的详细信息，这种场景下的攻击策略往往十分高效。

然而，当攻击者无法获取模型内部信息时，就进入了黑盒攻击的范畴。在这种场景下，攻击者只能通过向模型输入数据并观察其预测输出标签的方式进行攻击。与白盒攻击相比，黑盒对抗攻击更具现实意义，因为在实际应用中，大多数深度神经网络部署的系统并不允许用户端访问完整的模型信息。

因此，无论是自然语言处理、源代码安全还是其他领域，对抗攻击都已成为一个不容忽视的问题。为了确保这些系统的安全性和稳定性，我们需要深入研究对抗样本的生成机制，并探索有效的防御策略。

2.1.3 攻击定义

攻击者的能力概述

在当前的攻击场景中，攻击者具备扰乱预训练代码模型输入代码序列的能力，旨在生成对抗样本。这种扰乱操作基于自然语言对抗样本生成方法的深入研究，并特别允许对输入代码序列进行 token 级的细微扰动。攻击者能够在不改变代码整体结构的情况下，对特定代码片段进行精细修改。

在扰乱操作中，攻击者必须遵循严格的规则。他们仅能对一定数量的 token