面向泛在操作系统场景验证的定理自动识别方法

张满青[1],董云卫[1*],张 涛[2]
(1. 西北工业大学软件学院,陕西西安 710129;2. 澳门科技大学计算机科学与工程学院,澳门 999078)

摘 要: 在泛在计算环境下,操作系统需要在跨计算场景的异构硬件与动态应用之间提供统一支持,以保障系统的连续性与安全性。形式化验证作为确保操作系统内核正确性的关键手段,已经在高可信系统构建中发挥重要作用。其中,交互式定理证明方法为系统提供了严格且可靠的正确性保证。以 seL4 微内核为代表的验证实践表明,形式化方法虽然能够提供强正确性保证,但需要付出极高的验证成本。seL4 团队历时近 20 年构建了超过百万行的形式化证明,其验证成本远超系统代码开发成本。此外,该验证过程在不同计算场景之间缺乏良好的可迁移性。当开发者将系统移植到新的硬件平台时,验证人员往往需要重新进行大规模证明工作,这一过程严重制约了形式化方法在泛在计算环境中的应用。为解决这一问题,seL4 团队提出了将证明划分为计算场景无关部分与计算场景相关部分的思路,从而实现部分证明结果的复用。然而,该划分过程依赖专家经验,研究人员难以高效完成该任务,并且该过程容易引入人为错误。针对上述问题,本文提出一种基于知识蒸馏的自动化方法 ArchiDistill。该方法用于识别交互式定理中与计算场景相关的关键语义成分,从而支持验证任务的结构化拆分与跨场景迁移。ArchiDistill 通过将大规模预训练模型的语义和模式知识迁移至轻量级模型,并结合多任务蒸馏、引入辅助任务及课程学习策略,实现模型对计算场景相关特性的准确判别与跨平台迁移能力。在实验方面,本文基于 seL4 内核构建了一个包含 16 975 个样本的多计算场景验证数据集,并对所提出的方法进行了系统评估。实验结果表明,ArchiDistill 在计算场景相关定理识别任务上取得了优异性能。该方法的 Accuracy 达到 0.736 2,Precision 为 0.707 6,Recall 为 0.720 0,F_1-score 为 0.712 0,这些指标均优于传统模型 TextRNN(其 Accuracy 为 0.664 7)以及开源大语言模型 Qwen3-32B(其 Accuracy 为 0.709 6,F_1-score 为 0.630 2)。该方法在保持轻量化的同时,实现了性能与效率的有效提升。此外,本文在 CompCert 项目上构建了新的数据集,并使用该数据集验证所提出方法的泛化能力。实验结果表明,ArchiDistill 在跨任务场景中仍能保持良好的稳定性与适用性。

关键词: 泛在操作系统;交互式定理证明;计算场景相关性;知识蒸馏;多任务学习;课程学习;seL4 内核
基金项目: 国家重点研发计划(No.2022YFB4501800)
中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112(2026)04-1663-19
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250762

Automatic Theorem Identification for Scenario-Aware Verification in Ubiquitous Operating Systems

ZHANG Manqing[1], DONG Yunwei[1*], ZHANG Tao[2]
(1. *School of Software*, *Northwestern Polytechnical University*, *Xi'an*, *Shaanxi* 710129, *China*;
2. *School of Computer Science and Engineering*, *Macao University of Science and Technology*, *Macao* 999078, *China*)

Abstract: In ubiquitous computing environments, operating systems are required to provide unified support across heterogeneous hardware and dynamic applications in diverse computational scenarios, thereby ensuring system continuity and security. Formal verification, as a key technique for guaranteeing the correctness of operating system kernels, has played an essential role in building high-assurance systems. Among various approaches, interactive theorem proving offers the most rigorous and reliable correctness guarantees. The verification of the seL4 microkernel demonstrates that, although formal methods can ensure strong correctness, they incur extremely high verification costs. The seL4 team has spent nearly two decades constructing over one million lines of formal proofs, with verification costs far exceeding those of system development. Moreover, the verification process lacks sufficient portability across different computational scenarios. When the system is ported to a new hardware platform, verification engineers are often required to redo large-scale proofs, which severely limits the applicability of formal methods in ubiquitous computing environments. To mitigate this issue, the seL4 team proposed decomposing proofs into scenario-independent and scenario-dependent components, enabling partial reuse of verification results. However, this decomposition heavily relies on expert knowledge, making it difficult to perform effi-

收稿日期:2025-09-04;录用日期:2026-03-11;责任编辑:覃怀银
*通信作者:董云卫

ciently and prone to human error. To address these challenges, this paper proposes ArchiDistill, an automated method based on knowledge distillation. The proposed method aims to identify key semantic components in interactive theorems that are related to computational scenarios, thereby facilitating structured decomposition of verification tasks and cross-scenario transfer. ArchiDistill transfers semantic and pattern knowledge from large-scale pretrained models to a lightweight model, and enhances its performance through multi-task distillation, auxiliary tasks, and curriculum learning strategies. This enables accurate identification of scenario-related characteristics and improves cross-platform generalization capability. In the experimental evaluation, we construct a multi-scenario verification dataset based on the seL4 kernel, consisting of 16 975 samples, and conduct a comprehensive assessment of the proposed method. The results show that ArchiDistill achieves superior performance on the task of scenario-related theorem identification, with an accuracy of 0.736 2, precision of 0.707 6, recall of 0.720 0, and F_1-score of 0.712 0. These results outperform both the traditional model TextRNN with an accuracy of 0.664 7 and the open-source large language model Qwen3-32B with an accuracy of 0.709 6 and an F_1-score of 0.630 2. While maintaining a lightweight architecture, ArchiDistill achieves an effective balance between performance and efficiency. Furthermore, we construct a new dataset based on the CompCert project to evaluate the generalization capability of the proposed method. Experimental results demonstrate that ArchiDistill maintains strong stability and applicability in cross-task scenarios.

Keywords: ubiquitous operating system; interactive theorem proving; computing scenario relevance; knowledge distillation; multi-task learning; curriculum learning; seL4 kernel

Foundation Item(s): National Key Research and Development Program of China (No.2022YFB4501800)

0 引言

在泛在计算[1]环境下,计算资源广泛分布于各种设备与不同的计算场景中,操作系统需要在高度异构、动态变化的硬件与不同应用的计算场景之间提供统一的管理与支持[2]。不同的计算场景包括不同的处理器架构、不同的硬件环境和不同的应用资源等。泛在操作系统[3](ubiquitous operating system)正是面向这一需求提出的新型操作系统形态,其目标是在多样化设备和网络条件下,保障应用的连续性、透明性与安全性。然而,泛在操作系统面临的复杂性远超传统单机或集中式环境,尤其是在大规模分布式、自动化运行的场景中,系统的安全性与可靠性成为首要挑战[4]。在此背景下,引入操作系统形式化验证对保障系统的正确性具有重要意义[5]。通过形式化方法对操作系统内核进行数学建模与验证[6],能够在设计阶段发现潜在缺陷,确保关键属性(如内存隔离、权限管理、进程调度等)的严格正确性。这对泛在操作系统的应用尤为关键,因为传统测试手段难以覆盖所有执行路径与边界条件。交互式定理证明已被证明是保障操作系统内核正确性与安全性的有效手段[7-8]。以 seL4 微内核为代表,其验证工作持续近 20 年,累计完成超过百万行的形式化证明。实现内核代码仅耗费约 2.2 人年,但对应的形式化证明却耗费约 20 人年,平均每行 C 代码需要在证明器中编写约 40 行证明代码。这一巨大投入奠定了 seL4 作为目前唯一经完整功能正确性证明的操作系统内核的地位,但同时也凸显了操作系统验证的高昂成本。

然而,在泛在计算环境中,操作系统必须同时支持 ARM、RISC-V、x86 等多样化硬件平台并在异构设备上可靠运行[9]。传统基于交互式定理证明的验证往往针对单一计算场景展开,例如 seL4 的早期验证工作仅覆盖 ARM 硬件平台下的计算场景。一旦系统迁移到新的硬件平台,若仍采用现有验证流程,则意味着需要对整个内核重新开展大规模证明,验证开销甚至可能超过初始开发成本。对于已经投入了数十人年努力的 seL4 而言,这种迁移验证几乎难以承受。为缓解这一问题,seL4 团队提出了将内核证明划分为计算场景无关部分与计算场景相关部分的思路:计算场景无关部分涵盖核心抽象机制与关键性质,可在不同平台间高度复用;计算场景相关部分则局限于底层硬件接口与具体实现,可独立适配。该方法在一定程度上降低了迁移成本,但人工划分过程需要大量专家经验与时间投入,难以跟上内核功能扩展和多平台演进的需求,且容易受到人为疏漏的影响。因此,亟须提出一种自动化方法来识别并区分计算场景无关与计算场景相关定理,从而在降低迁移验证成本的同时提升系统的可扩展性与可靠性。

为此,本文提出了一种名为 ArchiDistill 的方法,用于自动区分已验证的交互式定理中计算场景相关与计算场景无关的部分。该方法基于知识蒸馏(knowledge distillation)[10],将大规模预训练模型的知识迁移到轻量级模型中,以兼顾准确性与高效性。大语言模型[11]在语义特征提取和模式归纳方面表现优异,能够在源代码层面有效捕捉不同计算场景实现之间的差异,但其计算开销过大,难以直接应用于验证工具链中。通过知识蒸馏,不仅压缩了大语言模型的

知识,还在蒸馏过程中引入软标签和辅助任务,从而增强了轻量级模型对计算场景相关特性的判别能力。进一步地,利用标签不确定性来分级样本难度,并结合多任务学习[12]与课程学习[13]进行训练,以提升模型在泛在操作系统计算场景验证中的可迁移性与泛化性。与现有关注证明自动化、前提选择和证明复用的研究不同,本文聚焦于“在已验证交互式定理中区分计算场景相关/无关部分”这一尚未被充分研究的任务。ArchiDistill可实现操作系统代码中计算场景相关特性的自动标注与抽取,从而降低跨场景验证的人工成本并提升系统的长期可维护性与可扩展性。

本文的创新贡献包括如下几点。

(1)提出多任务协同蒸馏与标签修复机制,解决了现有知识蒸馏方法仅使用主任务软标签导致语义覆盖不足的问题。该机制通过引入辅助任务的软标签并自适应校正潜在错误标签,实现了对大语言模型知识的更全面、准确和稳定的迁移。

(2)提出不确定性驱动的多任务课程学习策略,解决了学生模型在学习蒸馏知识时难以有效渐进吸收的问题。该策略利用软标签的不确定性动态调度训练样本,实现由易到难的学习过程,从而提升知识迁移效率与模型稳健性。

(3)收集并整理了seL4内核和CompCert编译器在多计算场景迁移过程中的验证数据,构建了一个包含16 975个样本和2 138个样本的证明数据集,并在Gitee(https://gitee.com/archidistill/archidistill)上进行发布,为后续研究提供帮助。在所构建的数据集上开展实验,验证了定理证明脚本分类方法ArchiDistill的有效性和优越性,为定理证明脚本的复用和开发提供了一条新途径。

1　相关工作

1.1　操作系统形式化验证

形式化方法在操作系统内核验证中得到了广泛应用。已有研究主要集中在两个方面:一是对具体功能模块的正确性进行验证,二是对操作系统的整体或者设计层进行形式化建模与验证。

在功能模块验证方面,现有研究采用多种定理证明与建模方法实现了不同子系统模块的正确性保障。Ma等人[14-15]利用Coq定理证明器对操作系统的异常管理模块进行了交互式形式化验证。在实时操作系统场景中,Guo等人[16]提出了可调度性验证方法,以保证系统在时间约束下的可靠运行。在内存管理方面,钱振江等人[17]利用非确定性自动机对汇编层代码进行建模,并结合Hoare三元组验证了内存管理模块的实现正确性;乔磊等人[18]针对航天器操作系统内核的内存管理模块进行了专门验证;徐家乐等人[19]实现了对操作系统内核权能访问控制模块的形式化验证。针对并发控制问题,微内核操作系统互斥量模块的功能正确性也得到了形式化证明[20]。此外,星上操作系统的任务管理模块在需求层面也已通过形式化建模与验证来确保其一致性和可实现性[21]。在语言层面的研究中,何韬等人[22]提出了针对Rust中unsafe代码段的验证机制,以提高内核实现的安全性。

在设计层验证方面,研究重点放在整体架构与安全需求的一致性上。微内核整体验证分为基于自动机[23]方法进行建模和验证、利用Z3 SMT求解器[24-26]进行推理验证以及采用交互式定理证明器[7-8,27]三种验证方法。对于领域特定操作系统的验证,王阳等人[28]提出了嵌入式操作系统的形式化验证方案,以应对资源受限环境下的设计挑战。在空间操作系统的设计验证中,Zhang等人[29]采用有限状态机建模,并在Coq中完成了相应的建模描述与证明。此外,还有工作利用Isabelle定理证明工具,验证了操作系统设计与安全需求之间的一致性[30-32]。

综上所述,现有研究大多集中在静态的验证结果上,对操作系统在实际演化过程中的持续验证与迁移问题[33]关注不足。尤其是在操作系统演化到多计算场景的不同计算场景下,现有方法往往需要大量人工干预,以区分计算场景相关与计算场景无关的定理,增加了迁移成本与验证负担。针对这一不足,本文提出了一种自动化的方法,用于在验证迁移过程中分离计算场景相关与计算场景无关的定理。

1.2　知识蒸馏与多任务学习

近年来,知识蒸馏与多任务学习的结合受到广泛关注,旨在通过知识迁移提升模型性能并解决多任务学习中的不平衡问题。现有研究主要从两方面展开。一方面,通过知识蒸馏将单任务模型的知识迁移至多任务模型,以缓解负迁移和任务跷跷板现象。Li等人[34]提出利用任务特定适配器对齐特征,实现跨任务的平衡参数共享。Xu等人[35]则提出KDAM框架,采用柯西-施瓦茨散度替代KL(Kullback-Leibler)散度,通过单教师多任务模型实现高效蒸馏。另一方面,Liu等人[36]探索了多教师蒸馏策略,如热带气旋风半径估计任务中,通过多教师模型指导多任务学生模型,并引入辅助任务显式约束任务间关系。此外,Jacob等人[37]提出在线蒸馏方法,通过同步训练单任务与多任务网络,结合自适应特征蒸馏和动态任务加权,显著提升了密集预测任务的性能。这些方法在推荐系统[38]、遥感图像分析[39]及自然语言处理[40]等领域均验证了其有效性,展示了知识蒸馏在多任务学习

中优化特征共享与任务平衡的潜力。

与之不同，本文提出的 ArchiDistill 框架专门针对操作系统验证过程中交互式定理证明脚本的移植性需求。在大语言模型知识蒸馏阶段，通过提示引导与标签修复机制获得可靠的软标签表示；在多任务学习阶段，结合不确定性驱动的课程学习策略，使学生模型能够逐步吸收大语言模型传递的结构化知识。该方法不仅解决了大语言模型输出噪声带来的可靠性问题，还通过主任务与辅助任务的协同训练提升了模型在定理判别上的泛化能力，因而在验证 seL4 定理是否为计算场景相关任务中展现出区别于现有工作的独特优势。

2 面向泛在操作系统场景验证的计算场景相关定理自动识别方法

泛在操作系统场景验证中计算场景相关定理自动识别方法 ArchiDistill 的整体框架如图 1 所示，该框架由大语言模型知识蒸馏阶段与多任务学习阶段两个核心环节组成。在大语言模型知识蒸馏阶段，分为两个步骤。(1)初始标签生成。利用已有的定理数据构建提示词，引导大语言模型对每个定理生成主任务与两个辅助任务的软标签，从而获得细粒度且可迁移的知识表示。(2)标签修复。考虑到大语言模型在初始推理中可能存在不准确性，本文进一步设计了标签修复机制。该机制综合初始输出、定理的语义信息及人工标注，对不一致或错误的标签进行校正，确保蒸馏知识的可靠性。在多任务学习阶段，基于修复后的软标签引入不确定性驱动的样本难度划分策略，以实现由易到难的课程学习。随后，学生模型在此排序的样本集上，通过联合优化主任务与辅助任务逐步吸收大语言模型传递的结构化知识，从而获得更稳健且具备更强泛化能力的验证结果。

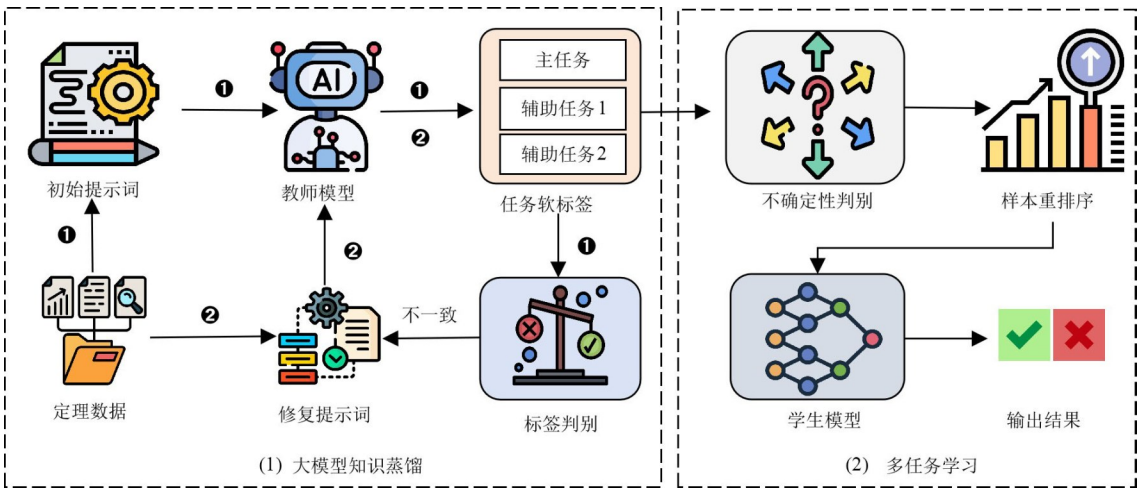


图 1 ArchiDistill 框架图

Figure 1 Overview of the ArchiDistill framework

2.1 计算场景相关定理识别问题定义

在多计算场景的软件系统形式化验证过程中，同一系统往往需要在不同硬件体系结构、编译目标或运行环境下保持功能与安全属性的一致性。例如，操作系统可能需要同时支持 ARM、RISC-V 和 x86 等不同处理器架构，编译器则需要面向不同目标平台生成语义一致的目标代码。为保证系统在各种计算场景下的正确性，形式化证明体系通常包含大量定理及其证明脚本。当系统迁移至新的计算场景时，这些定理并非都会发生变化。一部分定理仅依赖于抽象机制，与平台无关语义，在不同计算场景之间可直接复用，其证明结构与关键推理步骤保持不变；另一部分定理则依赖于具体硬件接口、体系结构语义或底层实现细节，在迁移过程中需要修改证明脚本甚至重构证明逻辑。本文据此将定理划分为两类：一是计算场景无关定理，即在跨场景迁移时无需进行实质性修改的定理；二是计算场景相关定理，即在至少某些计算场景之间迁移时需要调整证明结构或依赖模块的定理。

基于上述划分，本文关注如下识别任务：给定一个已完成形式化验证的软件系统及其定理集合，自动判别每个定理在跨计算场景迁移过程中是否具有场景依赖性。设系统在计算场景 c 下的定理集合为 $T^{(c)}=\{t_1^{(c)},t_2^{(c)},\cdots,t_n^{(c)}\}$，其中每个定理 $t_i^{(c)}$ 包含其形式化陈述、证明脚本以及依赖的上下文模块。当系统从计算场景 c_1 迁移到 c_2 时，原有定理通常需要进行适配。为更加清晰地对这一类别划分进行标签标记，本文定义一个迁移修改量函数 $\Delta_{c_1\to c_2}(t)$，该函数表示定理 t 在从计算场景 c_1 迁移到 c_2 时所需的修改程度，该修

改程度可以通过定理的差异进行度量。若对于任意两个计算场景c_1和c_2均满足$\Delta_{c_1\to c_2}(t)=0$,即定理在不同场景之间无需修改,则称其为计算场景无关定理。若存在某一对计算场景c_1和c_2使得$\Delta_{c_1\to c_2}(t)>0$,即迁移过程中需要对定理进行实质性修改,包括定理被删除或重构为其他定理,则称其为计算场景相关定理。

在此基础上,将计算场景相关定理识别任务建模为一个监督学习的二分类问题。设所有待识别定理构成集合$T=\{t_1,t_2,\cdots,t_n\}$。对于每个定理t_i其判别函数$f:t_i\to\{0,1\}$,其中$f(t_i)=0$表示计算场景无关,$f(t_i)=1$表示计算场景相关。在实际建模中,每个定理样本表示为$x=(S,P,D)$,其中S为交互式证明定理的形式化陈述,P为对应的证明脚本,D为其依赖的模块或上下文信息。给定带标签数据集$D\{(x_i,y_i)\}_{i=1}^{n}$,其中标签由实际迁移记录分析得到。计算场景相关定理识别任务目标是学习预测函数$\hat{f}(x)$使其能够自动判断一个定理是否依赖特定计算场景。

2.2　知识蒸馏阶段

为了从大语言模型中获取可迁移的知识,常见的做法是采用单一标签的知识蒸馏[41]。然而,这种方式往往只能传递有限的信息,难以为学生模型提供更丰富和结构化的知识表示。为此,本文采用GPT-4o作为教师模型,并设计了一个包含主任务和两个辅助任务的联合蒸馏方案,使学生模型能够在多维度上学习教师模型的知识。主任务是判断给定的seL4定理是否与计算场景相关,并输出一个介于0~1之间的软标签概率。与硬判定不同,软标签能够反映教师模型在预测中的不确定性,使学生模型不仅获得最终判断结果,还能学习到更细致的概率分布,从而在鲁棒性与泛化能力上有所提升。与此同时,两个辅助任务进一步拓宽了学生模型的学习范围,以进一步增强学生模型对硬件依赖信息的捕捉能力。辅助任务一是对定理所涉及的硬件特性类别进行软标签类别分布建模,其具体类别如表1所示,包括内存架构(涉及地址转换、MMU、缓存层次及内存保护等)、指令集架构(涉及指令格式、特权指令及扩展)、并行与同步支持(涉及多核、多线程、原子操作与一致性模型)、输入输出架构(涉及总线、中断、DMA与设备接口)以及无硬件依赖(没有硬件相关性的逻辑)。这一分类能够从不同维度刻画定理与底层硬件特性的关系,使蒸馏知识更加全面。

表1　seL4定理依赖硬件特性分类

Table 1　Classification of seL4 theorems based on hardware-specific characteristics

类别	名称	涉及内容
1	内存架构(Memory Architecture)	地址转换、MMU、缓存层次、内存保护
2	指令集架构(Instruction Set Architecture)	指令格式、操作类型、特权指令、扩展
3	并行与同步支持(Parallelism & Synchronization)	多核、多线程、原子操作、一致性模型
4	输入输出架构(I/O Architecture)	总线、中断、DMA、设备接口
5	无硬件依赖	没有硬件依赖的逻辑

辅助任务二是对定理所涉及的硬件依赖层级进行软标签类别分布建模,其层级划分如表2所示,包括微体系结构层(涉及缓存一致性、流水线行为、分支预测与TLB行为等)、指令集架构层(涉及寄存器、指令集、特权模式及中断机制)、系统级硬件抽象层(涉及中断控制器、DMA、总线与定时器等)、OS硬件抽象层(即seL4提供的硬件抽象API)以及硬件无关层(表示不依赖任何硬件的逻辑)。这种层级化的划分能够揭示定理在硬件抽象栈上的位置,使学生模型在学习硬件依赖时具备更具细粒度的区分能力。

表2　seL4定理硬件依赖层级

Table 2　Hardware dependency levels of seL4 theorems

层级	名称	涉及内容
Level 1	微体系结构层(Microarchitecture Level)	缓存一致性、流水线行为、分支预测、TLB行为等
Level 2	指令集架构层(ISA Level)	指令集、寄存器、特权模式、中断机制
Level 3	系统级硬件抽象层(System Hardware Abstraction Level)	中断控制器、DMA、总线、定时器
Level 4	OS硬件抽象层(OS HAL Level)	seL4提供的硬件抽象API
Level 5	硬件无关层(Hardware-Independent Level)	无硬件依赖的逻辑

为了有效获取教师模型的知识,本文提出了一种两阶段知识蒸馏方法。如图2所示,在第一阶段,即初始蒸馏阶段,基于提示工程[42](prompt engineering)为教师模型构造了结构化提示模板,引导其生成蒸馏结果。提示模板由4个部分组成:系统角色说明、任务信息、输出格式和输入信息。其中,系统角色说明

部分明确教师模型的身份为“seL4微内核、硬件架构和定理证明方面的专家”;任务信息部分要求模型基于定理内容完成主任务和辅助任务分类;约束条件要求模型的判断必须严格依赖定理本身及其隐含的硬件依赖关系;输出格式部分则限定结果必须为符合预定义结构的JavaScript Object Notation (JSON),而不得包含额外文本。基于该模板,教师模型能够生成包含软标签的初始预测结果。然而,在此阶段模型的输出可能与人工标注存在偏差,因此进一步设计了知识修复机制。

在第二阶段,即知识修复阶段,在同一会话中引入了修复提示,以使教师模型的输出与人工标签保持一致。本文提出标签修复机制的动机源于最近学术界对教师模型预测可靠性问题的关注。已有工作指出,当教师模型产生不准确或系统性偏差的预测时,直接使用其Soft Labels进行蒸馏可能会对学生模型的学习产生负面影响,并引入噪声监督信号[43-45]。为此,本文借鉴Lan等人[43]的相关工作,在蒸馏过程中引入真实标签,对教师模型的不准确预测进行显式修正。具体而言,修复提示采用纯指令式的形式编写,包含开场说明、任务需求、输出格式和输入信息。开场说明部分指出教师模型在前一次输出中与人工标注存在不一致之处;任务需求部分要求教师模型重新评估定理并生成符合人工标签约束的全新JSON输出;输出格式与初始阶段保持一致,仅允许返回JSON结构化结果;输入信息则包含人工标签、前一次预测结果以及定理内容。需要注意的是,在修复阶段没有再次指定系统角色,这是因为在同一会话中初始提示已经建立了任务模式与推理约束,修复提示只需依赖明确的输入和一致性要求,即可指导教师模型生成新的结果。通过这一两阶段的交互流程,在无需修改教师模型的情况下,有效保证了蒸馏知识的准确性与一致性。

2.3 多任务学习阶段

在多任务学习阶段,学生模型的目标是有效吸收从大语言模型蒸馏而来的知识。为此,该阶段设计了两步策略:首先,基于修复后的软标签,利用不确定性驱动的方法对训练样本进行难度排序,实现由易到难的课程学习;随后,学生模型在这一排序后的样本集上,通过联合优化主任务与辅助任务,逐步掌握大语言模型传递的结构化知识,从而提升验证结果的稳健性和泛化能力。下面将分别对这两部分进行详细介绍。

2.3.1 不确定性驱动的课程学习

在完成初始标签生成与标签修复后,ArchiDistill仍然面临软标签中正确与错误知识共存的情况。如果直接利用所有样本进行训练,那么噪声标签会显著削弱模型的泛化性能。为此,本文引入标签不确定性感知的课程学习(uncertainty-aware curriculum learning)机制,通过度量样本的不确定性,按照先易后难的方式逐步调度训练样本。具体而言,对于每个样本,定义其难度$h(x)$为主任务与辅助任务软标签熵[46]的加权和,其公式为

$$h(x)=U_{\text{dep}}(x)+\alpha\cdot U_{\text{feat}}(x)+\gamma\cdot U_{\text{level}}(x) \tag{1}$$

其中,α和γ分别设置为0.5,控制主任务与辅助任务在课程学习中的相对重要性;$U_{\text{dep}}(x)$表示主任务的软标签熵;$U_{\text{feat}}(x)$和$U_{\text{level}}(x)$分别表示辅助任务1和2的软标签熵;$h(x)$越大,表示样本整体不确定性越高,适合训练后期逐步引入;$h(x)$越小,表示样本预测可靠,可用于训练早期阶段。训练过程中按照时间步t逐渐放宽选样阈值$h(x)\leqslant\eta(t)$。

由于主任务是二分类,教师模型给出的软标签为二分类的概率分布$q_\tau^{\text{dep}}=[p,1-p]$。其中,$p$表示样本属于“硬件相关”的概率。其不确定性可以用熵表示:当p接近0或1时,熵较低,样本预测可靠;当$p\approx0.5$时,熵最大,样本不确定性高。为此,主任务软标

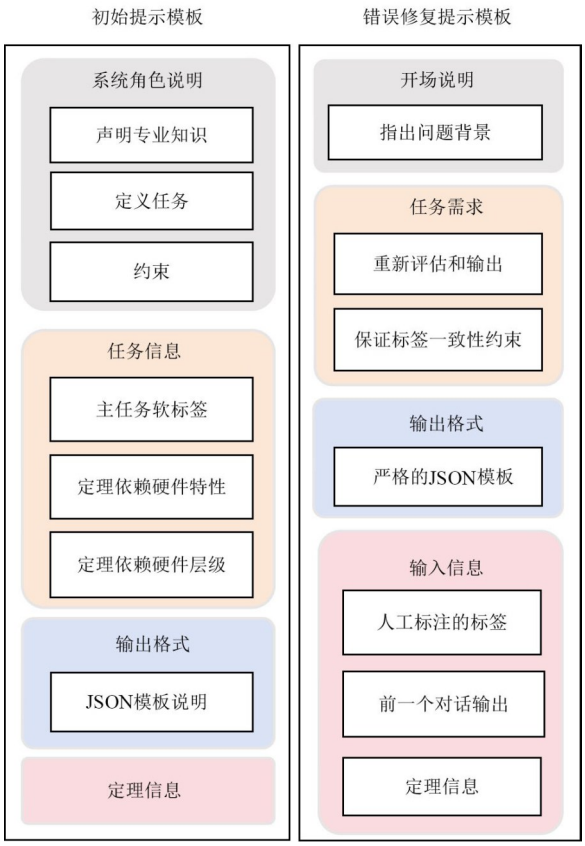


图2 知识蒸馏两阶段采取的提示词模板

Figure 2 Prompt templates for the two-stage knowledge distillation process

签的不确定性熵计算公式为

$$U_{\text{dep}}(x)=-\left(p\log p+(1-p)\log(1-p)\right) \tag{2}$$

辅助任务用于刻画定理与不同硬件特性类别及硬件依赖层级之间的关联程度。每个辅助任务包含5个预定义类别,教师模型为每个样本输出一个五维软标签概率分布,用于表示该样本在不同类别上的相关程度。对于辅助任务1的硬件依赖特性任务,其给出的软标签概率为5个类别的概率 $q_k^{\text{feat}}=\left[p_1,p_2,p_3,p_4,p_5\right],k=1,2,\cdots,5$;辅助任务2的硬件依赖层级任务,软标签概率分布与辅助任务1类似,其概率为 $q_k^{\text{level}}=\left[p_1,p_2,p_3,p_4,p_5\right],k=1,2,\cdots,5$。因此,可以先计算每个标签的熵表示不确定性,其计算公式为

$$U_k^{\text{feat}}(x)=-\sum_{i=1}^{5}p_i\log p_i,\ U_k^{\text{level}}(x)=-\sum_{i=1}^{5}p_i\log p_i \tag{3}$$

最后,对每个任务的5个标签取平均值,得到每个辅助任务整体的不确定性,其公式为

$$U_{\text{feat}}(x)=\frac{1}{5}\sum_{k=1}^{5}U_k^{\text{feat}}(x),\ U_{\text{level}}(x)=\frac{1}{5}\sum_{k=1}^{5}U_k^{\text{level}}(x) \tag{4}$$

2.3.2　多任务学习

在ArchiDistill框架中,将主任务与辅助任务联合建模,实现对定理的全面判别。主任务为二分类任务,用于判定定理是否与计算场景相关。在训练过程中,学生模型同时利用人工标注的硬标签以及教师模型生成的软标签。具体而言,对于主任务,训练损失由交叉熵损失[47]和KL散度损失[48]组成,以同时学习硬标签监督与教师提供的概率分布知识,其损失函数计算公式为

$$\mathcal{L}_{\text{dep}}=-\sum_{c\in\{0,1\}}y_c^{\text{dep}}\log\hat{y}_c^{\text{dep}}+\lambda\text{KL}\left(q_\tau^{\text{dep}}\middle\|\hat{y}^{\text{dep}}\right) \tag{5}$$

其中,$\hat{y}^{\text{dep}}$ 为学生模型输出的预测概率,λ 为软标签蒸馏权重。辅助任务包括判定定理的硬件依赖特性和硬件依赖层级,二者均通过五维软标签类别分布刻画定理与不同类别之间的关联程度。对于这两个辅助任务,如式(6)所示,学生模型通过最小化KL散度学习教师模型提供的软标签概率分布,从而在学生模型中注入教师的结构化知识。通过这种多任务蒸馏方式,学生模型能够同时吸收主任务和辅助任务的知识,实现多任务联合学习。

$$\mathcal{L}_{\text{feat}}=\frac{1}{5}\sum_{k=1}^{5}\text{KL}\left(q_\tau^{\text{feat}}\middle\|\hat{y}_k^{\text{feat}}\right),\mathcal{L}_{\text{level}}=\frac{1}{5}\sum_{k=1}^{5}\text{KL}\left(q_\tau^{\text{level}}\middle\|\hat{y}_k^{\text{level}}\right) \tag{6}$$

为了自适应调节各任务在联合训练中的贡献,采用同方差不确定性[49](homoscedastic uncertainty)自适应加权[50]。具体而言,对于多任务集合{dep, feat, level},加权多任务损失作为最终的训练目标函数,定义如式(7)所示:

$$\mathcal{L}=\sum_{t\in\{\text{dep, feat, level}\}}\frac{1}{2\delta_t^2}\mathcal{L}_t+\log\delta_t \tag{7}$$

其中,δ_t 为可学习参数,反映各任务的不确定性:不确定性高的任务对应 δ_t 较大,损失贡献减小;不确定性低的任务对应 δ_t 较小,损失权重增大,从而实现自适应平衡。

3　实验设计

3.1　数据集

由于此前研究中缺乏面向该任务的现成数据集,本研究基于seL4项目的历史证明数据构建了一个新的数据集。seL4的证明工件涵盖多个核心验证模块,包括invariant-abstract(抽象规范的不变量证明)、refine(抽象规范与设计规范之间的精化证明)、crefine(设计规范与C语义之间的精化证明)、access-control(完整性与权限控制的限制性证明)以及infoflow(保密性与非传递性不干扰证明)等。在当前的seL4项目中,这些证明工件已经完成了初步的计算场景层面的拆分,与通用计算场景无关的部分直接存放在证明工件文件夹下,而与特定计算场景相关的部分则存放在以计算场景名称命名的子文件夹中。

数据集的构建过程参考了项目在长期维护中的文件演化方式,通过分析Git commit历史数据,从版本变化中识别计算场景相关与计算场景无关的定理。具体步骤如下。

(1)定位计算场景拆分版本。对每个证明工件,定位到在文件结构中首次出现以计算场景命名的子文件夹的版本,该版本被视为已完成通用部分与计算场景相关部分的拆分。

(2)获取拆分前的定理集合。从该版本的前一个版本中提取该证明工件的所有定理,形成拆分前的定理全集。

(3)定理差异分析。将拆分前的定理集合与拆分后版本中位于主证明文件夹下的定理集合进行逐一比对。若定理仍保留在主证明文件夹下,则判定为通用定理;若定理在主文件夹中已不存在,则视为已迁移至计算场景命名子文件夹,对应为计算场景相关定理。

这种基于历史版本差异的分析方式,能够在不依赖人工标注的情况下准确区分通用与计算场景的相关定理,并保持与项目维护实际过程的一致性。此外,为更贴近工具在真实场景下的应用模式,数据集划分按照证明工件完成计算场景拆分的时间顺序进行。最早拆分的工件数据用于训练集,中期拆分的工件数据用于验证集,最后拆分的工件数据用于测试集。按照这种拆分方式最终得到训练集包含12 959条

数据,验证集为1 165条,测试集为2 851条。这种基于时间序列的划分策略不仅保证了数据集构建的真实性与时序一致性,还能有效评估模型在应对新的计算场景拆分任务时的泛化能力。

此外,为了验证所提方法在不同任务与数据集下的泛化能力,本文进一步在经过形式化验证的CompCert编译器项目上进行了实验扩展。CompCert由前端与后端两部分组成,其中前端的定理证明与具体计算场景无关,而后端则依赖于特定硬件架构的计算场景[51],即可能使用仅在特定处理器上存在的专用运算符。基于这一特性,我们对CompCert的证明工件进行了计算场景相关性划分,将其拆分为"计算场景相关部分"和"计算场景无关部分"。由于CompCert的定理证明工件并不存在明确的时间序列或版本演化关系,无法像seL4数据集那样依据项目的演化时间进行划分。为在保证数据分布代表性的同时避免引入主观偏差,本研究采用随机划分策略构建训练集、验证集与测试集。具体而言,在所有经过定理验证的样本中随机抽取数据,并按照6∶2∶2的比例划分为三部分。该方式在缺乏时间维度信息的情况下,能够最大程度保持整体数据分布的一致性与统计平衡,从而避免划分方式对实验结果造成系统性影响。最终,共得到1 282条训练样本、428条验证样本和428条测试样本。这种设计使得实验结果更能反映模型在一般情况下的性能水平与泛化趋势。

值得注意的是,seL4与CompCert作为长期开源项目,相关代码和证明工件可能已被包含在大语言模型的预训练语料中。但其仍不能仅靠记忆来完成,具体而言,模型需要判断一个定理是否与计算场景相关,而这一信息并未在原始代码或证明文本中被显式标注。相反,标签是通过分析seL4项目在Git历史中的版本演化(通用证明与计算场景相关证明的拆分过程)以及CompCert中前端/后端证明对计算场景依赖性的结构性差异间接构建得到。因此,模型必须结合定理的语义特征、所涉及的抽象层级及其在证明体系中的作用进行推理,才能完成正确区分,而无法直接通过记忆检索得到答案。例如,对于seL4的引理setExMonitor_wp,其引理文本仅刻画了setExMonitor操作在Hoare逻辑下对程序状态的局部保持性,该引理并未显式包含任何"计算场景相关"的标签信息。模型若要判断其属于计算场景相关证明,必须理解exclusive_state在seL4中所对应的硬件互斥监控机制(如ARM架构下的独占访问语义),并结合setExMonitor在内核抽象层级中的作用,推断该状态字段仅在特定硬件架构下具有语义意义。因此,本文构建的数据集能较好地评估模型对代码语义和证明结构的推理能力,从而避免潜在的数据泄露风险。

3.2 基线方法

为了评估所提出方法的有效性,实验首先选取了两类具有代表性的模型:传统机器学习分类器与大语言模型。首先,在传统机器学习方法方面,本文选择了文本分类任务中常用的算法,如逻辑回归(Logistic Regression,LR)、支持向量机(Support Vector Machine,SVM)、随机森林(Random Forest,RF)、朴素贝叶斯(Naive Bayes,NB)、K近邻(K-Nearest Neighbors,KNN)以及多层感知机(MultiLayer Perceptron,MLP)。这些方法具有较好的可解释性和稳定性,能够为ArchiDistill的性能提供基础参考。其次,在大型语言模型方面,选取了近年来在自然语言处理领域表现突出的开源模型,包括CodeLlama、Qwen系列和DeepSeek-V3等。针对本地运行的CodeLlama与Qwen系列大模型,使用LoRA(Low-Rank Adaptation)进行参数高效微调,从而在不更新全部模型参数的情况下,实现对特定任务的快速适配,并显著减少计算资源消耗。上述模型均可以从Hugging Face平台下载并在本地环境中部署运行。对于DeepSeek-V3模型,由于计算资源限制,本文使用在线API进行调用与测试。同时,参考Wang等人[52]在模型蒸馏实证研究中的评估方法,为了系统地评估训练后学生模型与其教师模型ChatGPT-4o之间的性能差异,在相同的评测设置下将学生模型与ChatGPT-4o进行了对比实验。利用上述大语言模型,可以评估seL4和CompCert定理在计算场景相关性判别任务中的适应性和性能上限。此外,考虑到本研究中学生模型基于TextRNN架构进行训练,为了更加全面地评估ArchiDistill方法的有效性与泛化能力,本文进一步选取了主流的深度学习文本分类模型TextCNN、TextRNN和TextRNN_Att作为对比实验对象。由于目前尚无针对该任务的公认基准方法,以上模型的引入能够兼顾大小模型技术,为实验结果提供多角度且全面的对比基础。

3.3 评价指标

为了全面评估模型在该任务上的表现,选用了分类任务中广泛使用的4个评价指标:Accuracy、Precision、Recall和F_1-score。其中,Accuracy用于衡量模型整体预测的正确率,能够反映模型在全局范围内的分类能力;Precision衡量被预测为正类的样本中实际为正类的比例,侧重评估预测结果的精确性;Recall衡量实际为正类的样本中被正确识别的比例,强调对正类样本的覆盖能力;F_1-score是Precision和Recall的调和平均值,用于在两者之间取得平衡,特别适合在类别分布不均衡的情况下综合评估模型性能。

通过这4个指标,可以从不同维度全面分析模型

的预测效果。Accuracy 提供整体表现的直观度量,Precision 与 Recall 分别体现预测的准确性和覆盖率,而 F_1-score 在二者之间权衡取舍,有助于避免单一指标带来的评估偏差。

此外,为了客观评估所提出方法的轻量化优势,本文从多维度对模型进行量化分析。具体而言,本文采用模型参数规模、训练时长、推理阶段显存占用以及单个样本的平均推理时间指标,对模型的计算与资源效率进行综合评估。相比仅从性能精度进行比较,这些指标能够更全面地反映模型在计算开销和部署可行性方面的表现。本文在实验中将 ArchiDistill 与代表性的大语言模型以及传统机器学习分类方法和深度学习模型对比,从而构建了覆盖不同模型范式的对比体系。通过这种设计,本文能够系统地验证 ArchiDistill 在保持较高分类性能的同时,是否在参数规模、显存占用及训练成本等方面展现出显著的轻量化优势。

3.4　实现细节

本文使用深度学习框架 PyTorch(https://pytorch.org/)来实现 ArchiDistill。ArchiDistill 教师模型使用 ChatGPT-4o,学生模型使用 TextRNN。在实验中,开源大语言模型的微调和推理运行在配备 4 张 NVIDIA L40S GPU 的服务器上,以满足其大规模计算需求;而轻量级学生模型 ArchiDistill 则在本地 NVIDIA RTX 6000 GPU 上运行,从而显著降低计算资源消耗并提升部署的灵活性。模型的嵌入维度设置为 128,隐藏层规模为 128,循环层采用单层双向 LSTM,并在输出层前加入 0.5 的 Dropout。在训练过程中,使用 Adam 优化器,学习率设置为 1×10^{-3},批大小为 32。训练总轮数上限为 100,并设置早停机制,若验证集准确率连续 5 个周期未提升,则提前终止训练。

4　实验结果与分析

为评估所提出的面向泛在操作系统场景验证中计算场景相关定理自动识别方法 ArchiDistill,本文提出以下 4 个研究问题。

研究问题 1:相较于现有的机器学习方法,ArchiDistill 在识别计算场景相关定理上的性能表现如何?

动机:在交互式定理证明中,识别计算场景相关定理对于操作系统场景验证至关重要。分类方法多依赖人工规则或直接训练的小模型,难以充分利用大语言模型的知识。本研究问题旨在验证基于知识蒸馏的 ArchiDistill 是否能够提升计算场景相关定理识别的准确性和鲁棒性,从而体现知识蒸馏在该任务中的实际优势。

研究问题 2:相较于现有的开源大语言模型,ArchiDistill 在计算场景相关定理识别中的表现如何?

动机:大语言模型在定理识别任务中通常表现优异,但计算成本高、推理速度慢,不利于大规模场景验证。本研究问题旨在探讨 ArchiDistill 是否能够在保持接近大语言模型识别性能的同时,显著降低计算成本和部署成本,从而兼顾精度与效率,体现其实用价值。

研究问题 3:ArchiDistill 的各个模块在计算场景相关定理识别中的作用如何?

动机:ArchiDistill 融合了多任务学习和任务权重自适应等机制,具体贡献尚待验证。本研究问题通过消融实验分析各模块在识别计算场景相关定理时的实际效果,从而揭示模型设计中对识别性能最关键的要素。

研究问题 4:ArchiDistill 在交互式定理证明中的识别结果是否可靠且具备可解释性?

动机:在操作系统场景验证中,仅有高准确率并不够,还需要确保定理识别结果的可信性和可解释性。本研究问题通过结合 t-SNE 全局可视化与 LIME 局部解释方法,评估 ArchiDistill 对计算场景相关定理的识别结果是否真实可靠,并进一步验证其在泛在操作系统计算场景验证中的可迁移性。

4.1　针对研究问题 1 的分析

为回答研究问题 1,将 ArchiDistill 与一系列主流机器学习模型及部分深度学习模型进行了对比,结果如表 3 所示。实验聚焦于在泛在操作系统场景验证中识别 seL4 定理是否与计算场景相关的任务。从表 3 中可以看出,传统机器学习方法整体表现有限。例如,LR、SVM 和 RF 的 Accuracy 均低于 0.50,说明这些直接训练的小模型在判断定理与计算场景相关性方面拟合能力不足。MLP 在 Accuracy 上达到 0.589 3,相比最差的 TextCNN 提升约 24%,但其 Precision 与 Recall 均未超过 0.56,性能提升有限。在深度学习模型中,TextRNN 表现相对较好,Accuracy 为 0.664 7,但 F_1-score 仅为 0.424 9,显示其在平衡 Precision 与 Recall 时仍存在不足。TextCNN 与 TextRNN_Att 的表现更差,F_1-score 分别仅为 0.278 9 和 0.335 9。

相比之下,ArchiDistill 在所有指标上均显著优于其他模型。Accuracy 提升至 0.736 2,比表现最好的 TextRNN 高约 7.15 个百分点;Precision 提升至 0.707 6,比 TextRNN 高出约 15.83 个百分点;Recall 提升至 0.720 0,比 TextRNN 高约 21.52 个百分点;F_1-score 提升至 0.712 0,比 TextRNN 高约 28.71 个百分点。这些显著的性能提升表明,ArchiDistill 在交互式定理证明中自动识别与计算场景相关的关键定理时更为准确和稳定,能够有效降低维护阶段因误判导致的不必要

表3 与传统机器学习和深度学习模型性能比较

Table 3 Performance comparison with traditional machine learning and deep learning models

方法名 数据集	seL4				CompCert			
	Accuracy	Precision	Recall	F_1-score	Accuracy	Precision	Recall	F_1-score
LR	0.469 0	0.508 2	0.508 7	0.467 5	0.960 3	0.970 8	0.944 8	0.955 7
SVM	0.458 1	0.499 2	0.499 1	0.476 8	0.986 0	0.987 7	0.981 9	0.984 7
RF	0.438 4	0.500 2	0.500 2	0.438 3	0.953 3	0.963 8	0.936 5	0.947 8
MLP	0.589 3	0.550 6	0.553 4	0.550 8	0.993 0	0.994 6	0.990 3	0.992 4
NB	0.480 5	0.562 3	0.557 7	0.479 6	0.927 6	0.949 2	0.899 4	0.917 3
KNN	0.432 1	0.490 6	0.490 7	0.432 1	0.941 6	0.9424	0.930 2	0.935 8
TextCNN	0.346 2	0.549 0	0.505 7	0.278 9	0.948 6	0.962 8	0.928 6	0.942 2
TextRNN	0.664 7	0.549 3	0.504 8	0.424 9	0.887 9	0.891 6	0.862 6	0.874 0
TextRNN_Att	0.373 6	0.528 9	0.511 1	0.335 9	0.936 9	0.925 4	0.949 3	0.933 6
ArchiDistill	0.736 2	0.707 6	0.720 0	0.712 0	0.997 7	0.998 2	0.996 8	0.997 5

人工干预和验证开销。特别是F_1-score的大幅提升表明模型在兼顾Precision与Recall的同时，能够可靠地识别关键计算场景相关定理，从而提高泛在操作系统场景验证的效率和可维护性。综上，实验结果充分验证了知识蒸馏机制能够将大语言模型的知识迁移至轻量化模型，使ArchiDistill在泛在操作系统验证任务中实现高效、准确且可靠的计算场景相关定理自动识别。

在CompCert数据集上，整体趋势与seL4一致，但各模型的总体性能水平更高，这说明该任务的特征分布相对更稳定。传统机器学习模型（如LR、SVM、RF）均取得了约0.94~0.98的Accuracy，但Precision与Recall之间仍存在一定差距。MLP的表现进一步提升，F_1-score达到0.992 4，显示出较强的拟合能力。然而，深度学习模型在该任务中的优势并不明显，TextRNN的F_1-score仅为0.874 0，而TextCNN和TextRNN_Att分别为0.942 2和0.933 6，说明其对结构化语义的捕获仍有限。相比之下，ArchiDistill在CompCert上依然保持领先，Accuracy高达0.997 7，Precision、Recall和F_1-score均超过0.996，充分验证了该模型在不同验证任务中的泛化能力与稳健性，能够推广到其他验证任务场景下。

表4展示了各模型在训练与推理阶段的资源开销对比情况，涵盖了模型参数规模、训练时长、推理显存占用以及单个样本的平均推理时间等指标。总体来看，传统机器学习模型（如LR、SVM、RF、NB、KNN）在参数规模和显存占用方面具有显著优势，全部可在CPU上完成推理，具备较高的资源利用效率。然而，其模型表达能力相对有限，难以捕获复杂语义特征，导致性能上限受限。与此同时，这类模型在训练耗时上差异较大，例如SVM在seL4数据集上的训练时长达到55.40 s，而LR与NB仅需不足1 s，体现出不同算法在迭代优化与特征处理复杂度上的显著差异。MLP作为轻量级神经网络模型，参数规模仅为9.02 MB，较传统模型有所增加，但仍处于较低水平，训练时长约为82.83 s（seL4数据集），兼顾了一定的非线性表达能力与较低的计算成本。相比之下，深度学习模型（TextCNN、TextRNN、TextRNN_Att）在模型规模与显存占用方面显著增加，参数量约为35~36 MB，推理阶段显存占用约为80~110 MB。尽管显存需求提升，但得益于GPU的并行计算支持，其推理时间依然极短（约0.02~0.03 ms），在实时性与吞吐量之间实现了良好平衡，适合在高并发验证任务中部署。

值得注意的是，不同数据集上（seL4与CompCert）的模型参数规模存在差异。这主要源于输入特征维度与词汇表规模的不同。深度学习模型的嵌入层参数量与词汇表大小直接相关，因此在数据集语料规模较大的seL4上，嵌入层参数数量较多，导致总体参数量相对增加。例如，TextCNN在seL4上的参数规模为35.49 MB，而在CompCert上为9.54 MB，体现了输入空间规模对模型参数数量的直接影响。同时可以看到，TextRNN、TextRNN_Att及基于TextRNN的ArchiDistill在参数规模上保持一致（均为35.91 MB/9.96 MB）。其原因在于三者共享相同的网络主体结构与参数配置，包括词嵌入层、双向循环层及输出层，且均采用相同的隐藏维度与层数。TextRNN_Att在TextRNN的基础上仅增加了一个注意力加权机制，该机制不引入新的大规模可训练参数，而是通过对RNN隐状态的加权计算实现特征聚合，因此总体参数规模基本一致。ArchiDistill在此结构上引入知识蒸馏机制，通过教师模型指导学生模型的训练，但蒸馏过程仅影响损失函数与优化目标，不改变学生模型的结构与参数维度，因此其参数规模亦与TextRNN保持一致。这表明ArchiDistill在不增加额外模型复杂度的前提下，实现

表4　与传统机器学习和深度学习模型在训练与推理阶段的资源开销对比

Table 4　Comparison of resource consumption in training and inference stages with traditional machine learning and deep learning models

方法名	模型参数规模/MB		训练时长/s		推理显存占用/MB		单个样本平均推理时间/ms
	seL4	CompCert	seL4	CompCert	seL4	CompCert	
LR	0.18	0.05	0.05	0.10	cpu-only		0.38
SVM	0.18	0.05	55.40	1.14	cpu-only		0.64
RF	0.09	0.03	6.28	0.30	cpu-only		2.73
MLP	9.02	2.78	82.83	11.99	cpu-only		0.41
NB	0.09	0.03	0.34	0.09	cpu-only		0.37
KNN	0.09	0.03	0.33	0.08	cpu-only		1.40
TextCNN	35.49	9.54	7.94	2.09	88.56	36.67	0.03
TextRNN	35.91	9.96	9.67	3.14	106.91	54.93	0.02
TextRNN_Att	35.91	9.96	9.83	1.66	106.93	54.96	0.03
ArchiDistill	35.91	9.96	27.40	10.75	106.93	54.96	0.04

了性能的显著提升。

综合来看,ArchiDistill在训练与推理阶段的资源开销略高于基础深度学习模型,但差距有限。在seL4数据集上,其训练时长为27.40 s,推理显存占用约106.93 MB,单样本平均推理时间仅0.04 ms,保持了优异的实时性与资源效率。总体而言,ArchiDistill在显著提升精度与泛化能力的同时,仅付出了较小的计算代价,展现出较高的性价比与工程可部署性。这说明通过知识蒸馏机制,ArchiDistill能够在充分保留大语言模型知识优势的同时,维持轻量级模型的高效运行特性,为资源受限的验证环境提供可行且高效的解决方案。

针对研究问题1的结论:实验结果表明,ArchiDistill在准确性、召回率和F_1-score等指标上均显著优于传统机器学习与深度学习模型,验证了知识蒸馏能够有效提升轻量化模型在计算场景相关定理识别任务中的性能。同时,ArchiDistill在模型参数规模、推理显存占用及单样本平均推理时间等资源开销上仍保持轻量化优势,说明在保证高性能的前提下,该方法能够高效运行于资源受限的验证环境,为实际部署提供了可行的轻量化解决方案。

4.2　针对研究问题2的分析

表5展示了在微调设置下,ArchiDistill与多种开源大语言模型以及教师模型ChatGPT-4o在seL4与CompCert两个数据集上的性能对比结果。整体来看,尽管微调显著提升了开源大语言模型在相关性识别任务中的表现,但其在不同评价指标之间仍存在一定波动,尤其是在Precision与Recall的平衡方面。在seL4数据集上,开源大语言模型已展现出一定的判别能力。随着模型规模的增大,整体性能呈现出上升趋势,其中Qwen3-32B的Accuracy达到0.709 6,F_1-score为0.630 2,在开源模型中表现最佳。然而,该模型在Recall上仍仅为0.624 4,限制了整体F_1-score的进一步提升。DeepSeek-V3与ChatGPT-4o在Precision上表现较为突出,分别达到0.806 9和0.799 9,但其Recall均低于0.54,导致F_1-score分别仅为0.471 3和0.478 8,表明即使是大规模或高性能模型,在该任务中也难以同时兼顾Precision与Recall。相比之下,CodeLlama-13B与Qwen3-8B的整体表现相对稳定,但Accuracy与F_1-score仍明显低于更大规模模型。相比之下,ArchiDistill在seL4数据集上的各项指标均显著优于上述模型。其Accuracy达到0.736 2,较表现最优的开源模型Qwen3-32B提升约2.66个百分点;F_1-score提升至0.712 0,相比Qwen3-32B提高约8.18个百分点。同时,ArchiDistill在Precision(0.707 6)与Recall(0.720 0)之间实现了更为均衡的权衡,显示出蒸馏模型在任务特定表示学习上的优势。

在CompCert数据集上,各模型的整体性能显著高于seL4,且表现更加稳定。Qwen系列模型的Accuracy与F_1-score均达到0.95以上,其中Qwen3-32B的F_1-score达到0.984 8。相比之下,DeepSeek-V3与ChatGPT-4o在该数据集上的表现明显下降,F_1-score分别仅为0.573 5和0.845 4,尤其在Recall上存在较大不足。这一现象表明,通用大语言模型在面对高度结构化、符号化的编译器定理验证任务时,其预训练和对齐目标可能与任务需求存在偏差。尽管如此,ArchiDistill在CompCert数据集上依然保持了显著优势,其Accuracy达到0.997 7,Precision、Recall和F_1-score分别为0.998 2、0.996 8和0.997 5,在所有对比方法中表现最佳。这进一步验证了ArchiDistill不仅能够有效继承教师模型的推理能力,还能够在不同验证任务场景下实现稳定且高精度的泛化性能。

表6对比了ArchiDistill与多种开源大语言模型以及教师模型ChatGPT-4o在微调训练与推理阶段的资

表5 与大语言模型性能比较

Table 5 Performance comparison with large language models

方法名	seL4				CompCert			
	Accuracy	Precision	Recall	F_1-score	Accuracy	Precision	Recall	F_1-score
CodeLlama-13B	0.594 2	0.455 0	0.474 0	0.446 1	0.918 2	0.932 8	0.892 1	0.907 3
Qwen3-8B	0.672 4	0.623 6	0.614 6	0.617 6	0.957 9	0.948 1	0.965 7	0.955 3
Qwen3-14B	0.688 9	0.637 9	0.593 8	0.594 4	0.979 0	0.982 4	0.972 2	0.976 9
Qwen3-32B	0.709 6	0.669 6	0.624 4	0.630 2	0.986 0	0.983 5	0.986 2	0.984 8
DeepSeek-V3	0.689 6	0.806 9	0.534 1	0.471 3	0.581 8	0.731 2	0.673 4	0.573 5
ChatGPT-4o	0.691 7	0.799 9	0.537 8	0.478 8	0.848 1	0.850 2	0.880 0	0.845 4
ArchiDistill	0.736 2	0.707 6	0.720 0	0.712 0	0.997 7	0.998 2	0.996 8	0.997 5

源开销，包括模型参数规模、微调训练时长、推理显存占用以及单个样本的平均推理时间。整体来看，现有开源大语言模型不仅参数规模庞大，而且其微调与推理阶段的计算成本同样较高。在本地微调场景下，CodeLlama-13B在seL4与CompCert数据集上的训练时间分别达到413 min和45 min以上；Qwen3-14B的训练时长进一步增加，分别为548 min和37 min。即使是参数规模较小的Qwen3-8B，其训练时间仍需3 h以上（seL4）和约21 min（CompCert）。随着模型规模扩大至32 B，Qwen3-32B的训练成本显著上升，在seL4数据集上的微调时间超过17 h，在CompCert上也接近2 h，体现出大模型微调在时间和算力上的高昂代价。

表6 与大语言模型资源开销对比

Table 6 Resource consumption comparison with large language models

方法名	模型参数规模		训练时长		推理显存占用		单个样本平均推理时间
	seL4	CompCert	seL4	CompCert	seL4	CompCert	
CodeLlama-13B	13 B		6 h 53 m 10 s	45 m 17 s	28 882 MB		66.49 s
Qwen3-8B	8 B		3 h 2 m 0 s	20 m 59 s	14 871 MB		65.77 s
Qwen3-14B	14 B		9 h 8 m 4 s	37 m 35 s	28 727 MB		95.23 s
Qwen3-32B	32 B		17 h 34 m 30 s	1 h 52 m 12 s	66 754 MB		102.63 s
DeepSeek-V3	671 B		—	—	在线API		8.23 s
ChatGPT-4o	—		—	—	在线API		7.83 s
ArchiDistill	35.91 MB	9.96 MB	27.40 s	10.75 s	106.93 MB	54.96 MB	0.04 ms

在推理阶段，上述模型同样表现出较高的资源需求。CodeLlama-13B与Qwen3-14B的推理显存占用均接近28 GB，单个样本的推理时间分别为66.49 s和95.23 s；Qwen3-8B仍需约14.9 GB显存，单次推理时间超过1 min。Qwen3-32B虽在性能上更具优势，但其显存消耗达到约66.8 GB，单样本推理时间超过100 s，进一步限制了其在高频验证场景中的实用性。对于超大规模模型，DeepSeek-V3的参数量高达约671 B。尽管通过在线API的方式避免了本地训练与部署，其单样本推理延迟仍达到8.23 s。作为蒸馏过程中的教师模型，ChatGPT-4o同样以在线API形式访问，其平均推理时间约为7.83 s，在推理能力上表现最优，但受限于推理延迟与调用成本，难以直接应用于大规模、低延迟的形式化验证任务。

相比之下，ArchiDistill在训练与推理阶段均展现出显著的资源效率优势。其模型参数规模仅为35.91 MB（seL4）和9.96 MB（CompCert），微调训练时间分别为27.40 s和10.75 s；在推理阶段，其显存占用均不足110 MB，单个样本的平均推理时间仅为0.04 ms。上述结果表明，ArchiDistill在有效蒸馏教师模型能力的同时，大幅降低了训练与推理成本，能够支持高频、低延迟且资源受限的自动验证与证明场景。

综合来看，虽然大语言模型在理解与泛化能力上具备潜力，但高昂的计算代价限制了其在资源受限环境（如嵌入式验证系统或持续集成验证流水线）中的应用。而ArchiDistill在保持高精度和高召回率的同时，显著降低了显存与推理时间，实现了真正意义上的轻量化与高可部署性平衡，为实际的形式化验证场景提供了可落地的解决方案。

针对研究问题2的结论：实验结果表明，与多种开源大语言模型相比，ArchiDistill在Accuracy、Precision、Recall和F_1-score等指标上均取得了更优且更加稳定的性能。同时，相较于教师模型ChatGPT-4o，ArchiDistill在保持具有竞争力甚至更优判别能力的同时，显著降低了模型规模、微调训练成本与推理开销。上述结果验证了ArchiDistill能够有效继承教师

模型的推理能力,并在泛在操作系统验证任务中实现高准确性、高效率与可部署性的统一。

4.3 针对研究问题3的分析

为回答研究问题3,本文设计了消融实验,对ArchiDistill中的核心组件逐一移除,以分析各模块在泛在操作系统计算场景验证下自动识别计算场景相关定理中的贡献,结果如表7所示。从整体上看,在seL4数据集上,移除任何核心组件均会导致性能下降,表明ArchiDistill的各项设计均对识别性能有正向影响。尤其是课程学习与辅助任务设计对Recall和F_1-score的提升作用最为显著。当移除课程学习时,Accuracy从0.736 2降至0.672 0,F_1-score降至0.511 6,说明逐步难度递进的训练机制在提升模型对计算场景相关定理的泛化能力和稳定性方面发挥了关键作用。类似地,去除辅助任务1后,F_1-score降至0.512 1,表明多任务协同在捕捉定理的多维度特征、提高识别准确性方面具有重要意义。

表7　不同组件消融实验

Table 7　Ablation study of different components

方法名	seL4				CompCert			
数据集	Accuracy	Precision	Recall	F_1-score	Accuracy	Precision	Recall	F_1-score
w/o课程学习	0.672 0	0.605 0	0.540 6	0.511 6	0.960 3	0.968 7	0.946 2	0.955 9
w/o主任务软标签	0.670 6	0.834 9	0.504 2	0.409 5	0.955 6	0.967 6	0.938 3	0.950 4
w/o辅助任务1	0.677 3	0.621 9	0.543 2	0.512 1	0.974 3	0.980 7	0.964 3	0.971 6
w/o辅助任务2	0.683 3	0.629 7	0.598 3	0.601 1	0.979 0	0.984 1	0.970 8	0.976 9
w/o自适应加权	0.691 0	0.648 9	0.573 3	0.561 8	0.974 3	0.978 9	0.965 7	0.971 7
w/o修复机制	0.661 2	0.568 9	0.523 1	0.483 2	0.943 9	0.940 3	0.937 7	0.939 0
ArchiDistill	0.736 2	0.707 6	0.720 0	0.712 0	0.997 7	0.998 2	0.996 8	0.997 5

在软标签的消融实验中,移除主任务软标签导致Precision虽达到最高值0.834 9,但Recall仅为0.504 2,F_1-score急剧下降至0.409 5,说明软标签对平衡Precision与Recall至关重要。它能够缓解模型对硬标签的过度拟合,使对计算场景相关定理的预测更加稳健。另外,移除辅助任务2或自适应加权机制时,性能下降相对温和,但仍不可忽视:w/o辅助任务2的F_1-score降至0.601 1,w/o自适应加权的F_1-score降至0.561 8。这表明任务加权机制和额外辅助信号虽非唯一性能提升来源,但在增强模型整体均衡表现与鲁棒性方面仍起到重要作用。综上所述,课程学习、主任务软标签与多任务协同是ArchiDistill提升计算场景相关定理自动识别性能的核心因素,而自适应加权与辅助任务2则进一步增强了模型在泛在操作系统计算场景验证中的稳健性和可靠性。

值得注意的是,在CompCert数据集上的结果同样印证了这一趋势。尽管整体数值水平高于seL4场景,但各组件的移除均导致不同程度的性能退化,表明ArchiDistill的模块化设计在不同验证场景下均具备一致的有效性。其中,课程学习与主任务软标签的移除使F_1-score分别从0.997 5降至0.955 9和0.950 4,体现了这两者在复杂编译验证场景中对精确识别定理的重要作用。辅助任务1与自适应加权机制的去除也分别造成约0.026的F_1-score下降,说明多任务协同和权重平衡机制有助于提升模型在复杂语义依赖场景下的泛化能力与稳健性。总体而言,ArchiDistill在CompCert上的表现进一步验证了其设计思路的普适性与稳健性,即核心模块的协同作用在不同类型的形式化验证任务中均能有效提升定理识别性能。

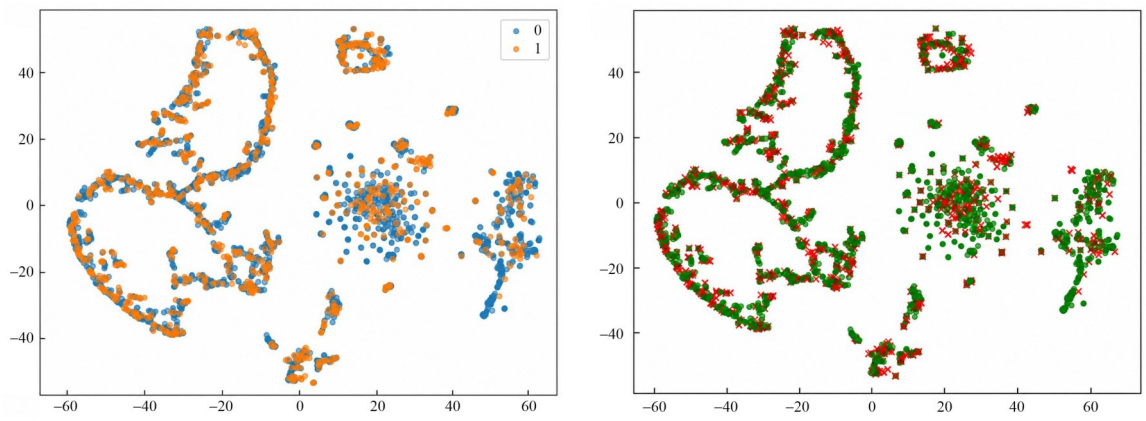
此外,表7还进一步分析了修复机制在蒸馏过程中的作用。该机制在教师模型首次预测与真实标签不一致时,引入正确的硬标签对教师进行反馈,从而重新生成对应的软标签概率分布。可以看到,在移除修复机制后(w/o修复机制),模型在seL4数据集上的性能出现了明显退化,Accuracy从0.736 2降至0.661 2,F_1-score从0.712 0降至0.483 2,降幅显著高于多数其他组件的消融结果。这一现象表明,当教师模型在复杂计算场景下产生错误或不稳定预测时,直接蒸馏其软标签可能会向学生模型传播噪声,进而显著影响模型的整体判别能力。修复机制通过对错误软标签进行纠正,有效缓解了蒸馏过程中的误导效应,从而提升了模型在Precision与Recall之间的整体平衡性。在CompCert数据集上,移除修复机制同样导致了性能明显下降,其F_1-score从0.997 5降至0.939 0,Accuracy下降超过5个百分点。尽管CompCert场景整体结构更加规整,但该结果仍表明,在高度符号化和规则密集的验证任务中,教师模型的局部预测误差同样可能对蒸馏过程产生累积影响。修复机制的引入有助于稳定教师信号的质量,从而提高学生模型在不同验证场景下的鲁棒性和可靠性。

针对研究问题3的结论:消融实验结果表明,课程学习、主任务软标签和多任务协同构成了ArchiDis-

till 在自动识别计算场景相关定理中的核心驱动因素，对模型性能提升起决定性作用。在此基础上，自适应加权机制与辅助任务进一步增强了模型在不同验证场景下的稳健性与整体表现，而修复机制则有效缓解了蒸馏过程中教师预测错误的传播问题，显著提升了模型在复杂场景下的稳定性和可靠性。

4.4 针对研究问题4的分析

为了进一步验证 ArchiDistill 在泛在操作系统计算场景验证下自动识别计算场景相关定理的表示能力，如图3(a)所示，对定理嵌入向量进行了 t-SNE 降维可视化[53]。可以观察到，不同类别的定理在低维空间中形成了若干相对清晰的簇状结构，表明模型能够捕捉定理之间的潜在语义关系。尽管计算场景相关与非相关定理在全局上仍存在一定重叠，但局部区域表现出明显的类别偏向性，尤其是在包含典型计算场景术语或纯逻辑推理成分的簇中，类别分布更加集中。这一结果表明，ArchiDistill 在保持定理语义聚合性的同时，能够凸显与计算场景相关性的差异特征，为后续自动判别计算场景相关定理提供了有效支持。



(a) 预测类别的可视化呈现

(a) Visualization of predicted classes

(b) 预测正确和错误的可视化呈现

(b) Visualization of correct and incorrect predictions

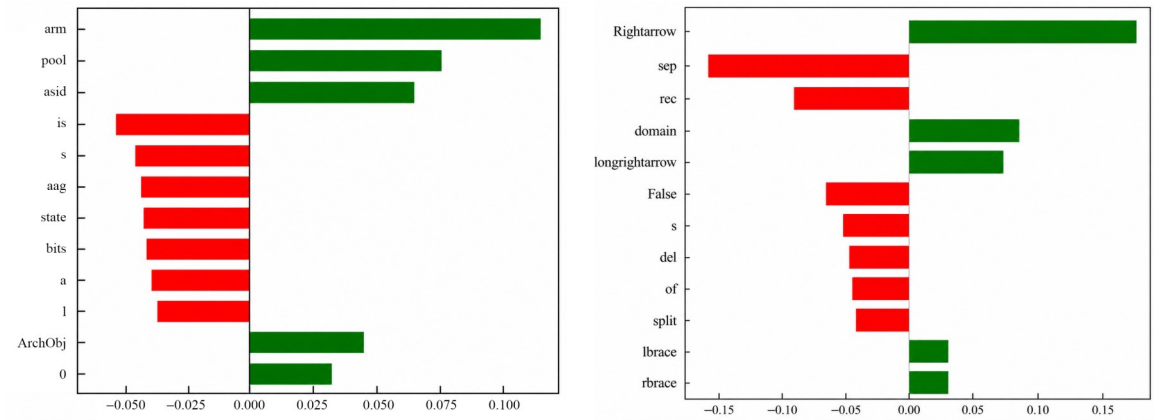
图3 t-SNE降维可视化

Figure 3 t-SNE visualization

为了进一步探究模型在不同定理上的预测表现，如图3(b)所示，基于 t-SNE 将嵌入空间中的样本分布进行可视化。图3中的绿色点表示预测正确的定理样本，红色叉表示预测错误的样本。从整体分布来看，大部分簇区域以绿色为主，说明模型在核心簇中能够保持较高的预测准确率。然而，在部分簇边界或类间过渡区域，红色样本相对密集，表明这些位置的定理语义表征存在重叠，容易导致分类混淆。此外，某些小簇中也出现了少量红色样本，说明在特定类别的边缘情形下，模型仍存在判别困难。总体而言，该可视化验证了 ArchiDistill 在捕捉关键定理主要语义模式时具备较强的区分能力，同时也提示了改进方向：在类间边界或语义交织区域引入任务特定知识，以进一步降低计算场景相关定理识别的错误率。

对于被判定为计算场景相关的定理，如图4(a)所示，LIME 可解释性分析[54]显示其最具贡献的正向特征包括 arm、pool 和 asid(address space identifier)。这些特征均与底层硬件紧密相关，arm 明确指向目标架构平台，asid 涉及内存管理单元，而 pool 常用于对象池分配机制。它们共同强化了 ArchiDistill 对定理与计算场景相关性的判断。相比之下，is、aag、state、bits 等特征被赋予负向权重，更倾向于系统状态或安全策略建模，而非直接反映硬件层面概念。此外，ArchObj 虽排序靠后，但其显式的“架构对象”语义进一步支持了正向判定。整体来看，ArchiDistill 对计算场景相关定理的预测主要依赖硬件专属术语，而逻辑与状态相关符号在一定程度上削弱了计算场景的相关性。

对于被判定为计算场景无关的定理，LIME 分析表明模型的判断主要由逻辑符号驱动。如图4(b)所示，Rightarrow 作为排名第一的正向特征，表明推理符号在非计算场景类定理中出现频率更高。同样，sep 与 rec 作为负向特征，进一步强调了这些定理更侧重逻辑推理和递归建模，而非硬件特性。domain 与 longrightarrow 虽带来一定正向贡献，但仍偏向抽象语义层面，而非计算场景上下文。其余如 false、split、lbrace、rbrace 等特征主要体现定理的形式化和证明结构，而不包含硬件专有信息。因此，ArchiDistill 主要通过捕捉逻辑和证明结构性标记，准确判定此类定理与具体计算场景无关，从而在泛在操作系统场景的交



(a) 正样本样例的特征权重分析　　(b) 负样本样例的特征权重分析

(a) Feature importance analysis of positive samples　　(b) Feature importance analysis of negative samples

图4　LIME特征权重分析

Figure 4　LIME feature importance analysis

互式定理证明中实现可靠的计算场景相关性识别。

针对研究问题4的结论:t-SNE降维可视化和LIME分析表明,ArchiDistill能够有效捕捉定理的语义模式和关键特征。对计算场景相关定理主要依赖硬件专属术语,而对非相关定理则依赖逻辑和证明结构标记,从而实现准确且可解释的计算场景相关性识别。

4.5　误判分析

为深入理解模型在计算场景相关定理识别任务中的表现,对测试集中产生的假阳性(False Negative,FP)和假阴性(False Positive,FN)样本进行了分析。定理识别任务的核心在于判断定理在跨硬件架构迁移时是否需要修改。通过对误分类定理的人工审查发现,这些样本大多位于抽象规范语义与硬件架构语义的交界区域,其迁移相关性本身并不十分明确。因此,对于这类边界样本,本文方法在判断其是否需要跨架构适配时仍存在一定难度。

4.5.1　假阳性(FP)分析

假阳性(FP)是指模型将实际在跨硬件架构迁移过程中无需修改的定理,误判为需要修改的计算场景相关定理。这类定理的共同特征在于,虽然其语义上并不依赖具体体系架构实现,在实际跨架构支持过程中通常可以保持不变,但在系统结构上处于“接近机器层”的模块边界,因此呈现出一定的架构相关结构信号,容易被模型识别为可能需要适配的定理。

以choose_thread_guarded_pas_domain定理为例,如图5所示,该定理旨在证明在满足访问控制约束的前提下,调度操作仍能保持系统安全不变量。其证明涉及调度域cur_domain、优先级队列ready_queues以及线程切换相关引理。从系统分层角度看,该定理位于抽象调度层,不直接涉及页表结构或机器状态语义,因此在跨硬件架构迁移时通常无需修改。然而,在精化层或实现层中,线程切换往往与底层上下文切换机制存在结构关联,例如寄存器保存和机器状态更新等操作。因此,从语义依赖图的角度看,该定理位于抽象层与实现层之间的邻近区域,呈现出“接近机器层”的结构特征。虽然模型捕捉到了相关的结构信号,并据此将其预测为需要在迁移时修改,但由于该定理并不存在真实的架构依赖,这一预测属于偏保守的误判。

```
1  lemma choose_thread_guarded_pas_domain:
2      "\<lbrace>pas_refined aag and valid_queues\<rbrace> choose_thread
3       \<lbrace>\<lambda>xb. guarded_pas_domain aag\<rbrace>"
4    apply (simp add: choose_thread_def guarded_switch_to_def | wp switch_to_thread_respects
5  switch_to_idle_thread_respects gts_wp switch_to_thread_guarded_pas_domain)+
6    apply (clarsimp simp: pas_refined_def)
7    apply (clarsimp simp: tcb_domain_map_wellformed_aux_def)
8    apply (erule_tac x="(hd (max_non_empty_queue (ready_queues s (cur_domain s))), cur_domain
9  s)" in ballE)
10   ...
11   apply force+
12   done
```

图5　假阳性样本实例

Figure 5　False positive sample examples

4.5.2　假阴性(FN)分析

假阴性(FN)是指定理在跨硬件架构迁移时实际上需要修改,但模型却将其预测为计算场景无关定理。这类样本的共同特征在于,其架构依赖通常隐藏在机器状态或硬件接口语义中,而在定理的表面结构上并不明显。由于其证明形式与大量抽象层定理高度相似,在缺乏显式语义层级标识的情况下,模型难以准确判断其在跨架构迁移时是否需要适配。

以图6所示的定理dmo_getExMonitor_wp'[wp]为例。该定理证明机器操作getExMonitor在Hoare三元组语义下保持性质P,其前置与后置条件涉及machine_state、exclusive_state、do_machine_op等机器级语

义要素。getExMonitor用于访问体系结构中的独占监视器状态,而不同处理器架构对该机制的实现方式存在差异。因此,此类定理在跨架构支持过程中通常需要根据目标硬件语义进行适配,属于计算场景相关定理。然而,从形式结构上看,该定理只是证明某个操作在Hoare三元组语义下保持给定性质,其证明模式与大量抽象层不变量保持定理高度一致。其架构依赖并未体现在复杂的证明结构或显式的架构名称上,而是体现在“操作对象属于机器级状态”这一语义层级上。模型未能识别其迁移相关性,说明当前方法在刻画和区分不同语义层级(尤其是机器级操作语义)方面仍有进一步提升的空间。

```
lemma dmo_getExMonitor_wp'[wp]:
  "\<lbrace>\<lambda>s. P (exclusive_state (ksMachineState s)) s\<rbrace>
   doMachineOp getExMonitor \<lbrace>P\<rbrace>"
  apply (simp add: doMachineOp_def)
  apply (wp modify_wp | wpc)+
  apply clarsimp
  apply (erule use_valid)
  apply wp
  apply simp
  done
```

图6 假阴性样本实例

Figure 6 False negative examples

5 总结与展望

本文针对泛在操作系统在多硬件平台下交互式定理证明形式化验证成本高、可复用性不足的问题,提出了基于知识蒸馏的自动化方法ArchiDistill,实现了对形式化定理中计算场景相关与计算场景无关部分的自动区分。通过结合多任务蒸馏、辅助任务与课程学习策略,该方法在保持轻量化的同时具备较强的判别能力与跨平台迁移能力。基于seL4内核多计算场景验证数据集的实验结果表明,ArchiDistill在计算场景相关定理识别任务上的准确率比传统方法提高了15%,并在轻量模型框架下达到甚至超越大语言模型的性能。该研究不仅为降低跨计算场景验证的人力成本提供了新的思路,还为提升泛在操作系统的长期可维护性与扩展性奠定了方法基础。

尽管ArchiDistill在自动区分计算场景相关与计算场景无关定理方面取得了良好效果,但这仅是迈向多计算场景自动化验证的第一步。未来的研究可以进一步探索如何在自动识别的基础上,实现面向不同计算场景的定理自动生成与迁移推理,从而减少人工编写和适配的工作量。其次,当前方法主要针对seL4内核开展实验,后续可扩展至其他操作系统内核和更大规模的代码库,以验证其通用性与可扩展性。此外,如何在数据有限或新计算场景出现时实现快速适应,仍是亟须解决的挑战。进一步地,将ArchiDistill与符号方法、因果推理或不确定性建模相结合,有望提升验证结果的可解释性与可靠性。最终,在实际工具链中实现与现有验证框架的深度集成,并支持跨计算场景的自动化定理生成与验证,将是推动泛在操作系统形式化验证走向实用化的重要方向。

致谢 感谢肖炳旭博士给本文提出的参考意见。

参考文献

[1] Weiser M. The computer for the 21st century[J]. ACM SIGMOBILE Mobile Computing and Communications Review, 1999, 3(3): 3-11.

[2] 曹云帆, 赵超懿, 刘瀚之, 等. 人机物融合泛在应用的系统支撑[J]. 中国科学: 信息科学, 2025, 55(3): 464-480.
Cao Yunfan, Zhao Chaoyi, Liu Hanzhi, et al. System support for ubiquitous human-cyber-physical fusion applications[J]. Scientia Sinica Informationis, 2025, 55(3): 464-480. (in Chinese)

[3] Mei Hong, Guo Yao. Toward ubiquitous operating systems: A software-defined perspective[J]. Computer, 2018, 51(1): 50-56.

[4] 梅宏, 曹东刚, 谢涛. 泛在操作系统: 面向人机物融合泛在计算的新蓝海[J]. 中国科学院院刊, 2022, 37(1): 30-37.
Mei Hong, Cao Donggang, Xie Tao. Ubiquitous operating system: Toward the blue ocean of human-cyber-physical ternary ubiquitous computing[J]. Bulletin of Chinese Academy of Sciences, 2022, 37(1): 30-37. (in Chinese)

[5] Brun M, Achermann R, Chajed T, et al. Beyond isolation: OS verification as a foundation for correct applications[C]// Proceedings of the 19th Workshop on Hot Topics in Operating Systems. New York: ACM, 2023: 158-165.

[6] 钱汉伟, 王承毅. 操作系统形式化验证综述[J]. 河海大学学报(自然科学版), 2021, 49(5): 473-481.
Qian Hanwei, Wang Chengyi. Review of formal verification for operating system[J]. Journal of Hohai University (Natural Sciences), 2021, 49(5): 473-481. (in Chinese)

[7] Klein G, Elphinstone K, Heiser G, et al. SeL4: Formal verification of an OS kernel[C]//Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. New York: ACM, 2009: 207-220.

[8] Gu Ronghui, Shao Zhong, Chen Hao, et al. CertiKOS: An extensible architecture for building certified concurrent OS kernels[C]//Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2016: 653-669.

[9] 古金宇, 李浩, 夏虞斌, 等. BrickOS: 面向异构硬件资源的积木式内核[J]. 中国科学: 信息科学, 2024, 54(3):

491-513.
Gu Jinyu, Li Hao, Xia Yubin, et al. BrickOS: Specialized kernels for heterogeneous hardware resources[J]. Scientia Sinica Informationis, 2024, 54(3): 491-513. (in Chinese)

[10] Gou Jianping, Yu Baosheng, Maybank S J, et al. Knowledge distillation: A survey[J]. International Journal of Computer Vision, 2021, 129(6): 1789-1819.

[11] Naveed H, Khan A U, Qiu Shi, et al. A comprehensive overview of large language models[J]. ACM Transactions on Intelligent Systems and Technology, 2025, 16(5): 1-72.

[12] Zhang Yu, Yang Qiang. A survey on multi-task learning[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(12): 5586-5609.

[13] Bengio Y, Louradour J, Collobert R, et al. Curriculum learning[C]//Proceedings of the 26th Annual International Conference on Machine Learning. New York: ACM, 2009: 41-48.

[14] Ma Zhi, Qiao Lei, Yang Mengfei, et al. Verification of real time operating system exception management based on SPARCv8[J]. Journal of Computer Science and Technology, 2021, 36(6): 1367-1387.

[15] 马智, 乔磊, 杨孟飞, 等. 面向SPARC处理器架构的操作系统异常管理验证[J]. 软件学报, 2021, 32(6): 1631-1646.
Ma Zhi, Qiao Lei, Yang Mengfei, et al. Verification of operating system exception management for SPARC processor architecture[J]. Journal of Software, 2021, 32(6): 1631-1646. (in Chinese)

[16] Guo Xiaojie, Lesourd M, Liu Mengqi, et al. Integrating formal schedulability analysis into a verified OS kernel[C]// Proceedings of 31st International Conference on Computer Aided Verification. Cham: Springer, 2019: 496-514.

[17] 钱振江, 刘永俊, 姚宇峰, 等. 微内核架构内存管理的形式化设计和验证方法研究[J]. 电子学报, 2017, 45(1): 251-256.
Qian Zhenjiang, Liu Yongjun, Yao Yufeng, et al. Research on method of formal design and verification of memory management based on microkernel architecture[J]. Acta Electronica Sinica, 2017, 45(1): 251-256. (in Chinese)

[18] 乔磊, 杨孟飞, 谭彦亮, 等. 基于Event-B的航天器内存管理系统形式化验证[J]. 软件学报, 2017, 28(5): 1204-1220.
Qiao Lei, Yang Mengfei, Tan Yanliang, et al. Formal verification of memory management system in spacecraft using Event-B[J]. Journal of Software, 2017, 28(5): 1204-1220. (in Chinese)

[19] 徐家乐, 王淑灵, 李黎明, 等. 操作系统内核权能访问控制的形式验证[J]. 软件学报, 2025, 36(8): 3570-3586.
Xu Jiale, Wang Shuling, Li Liming, et al. Formal verification of capability-based access control in operating system kernel[J]. Journal of Software, 2025, 36(8): 3570-3586. (in Chinese)

[20] 张林雁, 李希萌, 施智平, 等. 微内核操作系统互斥量模块功能正确性的形式化验证[J]. 软件学报, 2024, 35(9): 4179-4192.
Zhang Linyan, Li Ximeng, Shi Zhiping, et al. Formal verification of functional correctness for mutexes in microkernel[J]. Journal of Software, 2024, 35(9): 4179-4192. (in Chinese)

[21] 姜菁菁, 乔磊, 杨孟飞, 等. 基于Coq的操作系统任务管理需求层建模及验证[J]. 软件学报, 2020, 31(8): 2375-2387.
Jiang Jingjing, Qiao Lei, Yang Mengfei, et al. Operating system task management requirements layer modeling and verification based on Coq[J]. Journal of Software, 2020, 31(8): 2375-2387. (in Chinese)

[22] 何韬, 董威, 文艳军. GhostFunc: 一种针对Rust操作系统内核的验证方法[J]. 软件学报, 2025, 36(8): 3494-3511.
He Tao, Dong Wei, Wen Yanjun. GhostFunc: Verification method for Rust operating system kernel[J]. Journal of Software, 2025, 36(8): 3494-3511. (in Chinese)

[23] De Oliveira D B, Cucinotta T, De Oliveira R S. Efficient formal verification for the Linux kernel[M]//Software Engineering and Formal Methods. Cham: Springer International Publishing: 315-332.

[24] Nelson L, Sigurbjarnarson H, Zhang Kaiyuan, et al. Hyperkernel: Push-button verification of an OS kernel[C]// Proceedings of the 26th Symposium on Operating Systems Principles. New York: ACM, 2017: 252-269.

[25] Chen Xiangdong, Li Zhaofeng, Mesicek L, et al. Atmosphere: Towards practical verified kernels in Rust[C]// Proceedings of the 1st Workshop on Kernel Isolation, Safety and Verification. New York: ACM, 2023: 9-17.

[26] Narayanan V, Baranowski M S, Ryzhyk L, et al. RedLeaf: Towards an operating system for safe and verified firmware[C]//Proceedings of the Workshop on Hot Topics in Operating Systems. New York: ACM, 2019: 37-44.

[27] Xu Fengwei, Fu Ming, Feng Xinyu, et al. A practical verification framework for preemptive OS kernels[C]//Pro-

ceedings of 28th International Conference on Computer Aided Verification. Cham: Springer, 2016: 59-79.

[28] 王阳, 方竞成, 蔡雄, 等. 一种面向嵌入式操作系统的形式化验证方法[J]. 华东师范大学学报(自然科学版), 2024(4): 1-17.
Wang Yang, Fang Jingcheng, Cai Xiong, et al. A formal verification method for embedded operating systems[J]. Journal of East China Normal University (Natural Science), 2024(4): 1-17. (in Chinese)

[29] Zhang Jinkun, Qiao Lei, Yang Hua, et al. Formal modeling and verification of the design layer of space operating systems[C]//Proceedings of the 11th International Conference on Signal and Information Processing, Networking and Computers. Singapore: Springer, 2024: 516-525.

[30] 钱振江, 黄皓, 宋方敏. 操作系统汇编级形式化设计和验证方法[J]. 软件学报, 2016, 27(12): 3143-3157.
Qian Zhenjiang, Huang Hao, Song Fangmin. Method of formal design and verification of OS on assembly layer[J]. Journal of Software, 2016, 27(12): 3143-3157. (in Chinese)

[31] 钱振江, 黄皓, 宋方敏. 操作系统形式化设计与安全需求的一致性验证研究[J]. 计算机学报, 2014, 37(5): 1082-1099.
Qian Zhenjiang, Huang Hao, Song Fangmin. Research on consistency verification of formal design and security requirements for operating system[J]. Chinese Journal of Computers, 2014, 37(5): 1082-1099. (in Chinese)

[32] 钱振江, 刘苇, 黄皓. 操作系统对象语义模型(OSOSM)及形式化验证[J]. 计算机研究与发展, 2012, 49(12): 2702-2712.
Qian Zhenjiang, Liu Wei, Huang Hao. OSOSM: Operating system object semantics model and formal verification[J]. Journal of Computer Research and Development, 2012, 49(12): 2702-2712. (in Chinese)

[33] Lawall J, Nishimura K, Lozi J P. Should we balance? Towards formal verification of the Linux kernel scheduler[M]// Static Analysis. Cham: Springer Nature Switzerland, 2025: 194-215.

[34] Li Weihong, Bilen H. Knowledge distillation for multi-task learning[C]//Proceedings of Computer Vision - ECCV 2020 Workshops. Cham: Springer, 2020: 163-176.

[35] Xu Yangyang, Yang Yibo, Zhang Lefei. Multi-task learning with knowledge distillation for dense prediction[C]// Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway: IEEE, 2023: 21493-21502.

[36] Liu Jia, Yue Yinlei, Jin Yongjun, et al. Collaborative estimation of tropical cyclone wind radii with multitask learning and multiteacher distillation[J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2025, 18: 12545-12558.

[37] Jacob G M, Agarwal V, Stenger B. Online knowledge distillation for multi-task learning[C]//Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. Piscataway: IEEE, 2023: 2358-2367.

[38] Yi Qingqing, Tang Jingjing, Zeng Yujian, et al. DMMP: A distillation-based multi-task multi-tower learning model for personalized recommendation[J]. Knowledge-Based Systems, 2024, 284: 111236.

[39] Lê H Â, Pham M T. Leveraging knowledge distillation for partial multi-task learning from multiple remote sensing datasets[C]//IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium. Piscataway: IEEE, 2024: 8019-8023.

[40] Matei V C, Tăiatu I M, Smădu R A, et al. Enhancing Romanian offensive language detection through knowledge distillation, multi-task learning, and data augmentation[C]// Proceedings of 29th International Conference on Applications of Natural Language to Information Systems on Natural Language Processing and Information Systems. Cham: Springer, 2024: 317-332.

[41] Yang Zhendong, Zeng Ailing, Li Zhe, et al. From knowledge distillation to self-knowledge distillation: A unified approach with normalized loss and customized soft labels[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway: IEEE, 2023: 17139-17148.

[42] Marvin G, Hellen N, Jjingo D, et al. Prompt engineering in large language models[M]//Data Intelligence and Cognitive Informatics. Singapore: Springer Nature Singapore, 2024: 387-402.

[43] Lan Weichao, Cheung Y M, Xu Qing, et al. Improve knowledge distillation via label revision and data selection[J]. IEEE Transactions on Cognitive and Developmental Systems, 2025, 17(6): 1377-1388.

[44] Sun Wujie, Chen Defang, Siwei Lyu, et al. Knowledge distillation with refined logits[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Piscataway: IEEE, 2025: 1110-1119.

[45] Limantoro S E. Parameter-free logit distillation via sorting mechanism[J]. IEEE Signal Processing Letters, 2025, 32: 3849-3853.

[46] Seidenfeld T. Entropy and uncertainty[J]. Philosophy of Science, 1986, 53(4): 467-491.
[47] Mao Anqi, Mohri M, Zhong Yutao. Cross-entropy loss functions: Theoretical analysis and applications[C]//Proceedings of the 40th International Conference on Machine Learning. New York: Curran Associates, Inc., 2023: 23803-23828.
[48] Cui Jiequan, Tian Zhuotao, Zhong Zhisheng, et al. Decoupled Kullback-Leibler divergence loss[C]//Proceedings of the 38th International Conference on Neural Information Processing Systems. New York: Curran Associates Inc., 2024: 2370.
[49] Cipolla R, Gal Y, Kendall A. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 7482-7491.
[50] 鲁银圆, 许升全, 谢娟英. AI-DETR: 自适应加权的可解释目标检测方法[J]. 电子学报, 2025, 53(7): 2279-2304.
Lu Yinyuan, Xu Shengquan, Xie Juanying. AI-DETR: Interpretable object detection method based on adaptive weighting[J]. Acta Electronica Sinica, 2025, 53(7): 2279-2304. (in Chinese)
[51] Besson F, Blazy S, Wilke P. A verified CompCert front-end for a memory model supporting pointer arithmetic and uninitialised data[J]. Journal of Automated Reasoning, 2019, 62(4): 433-480.
[52] Wang Ruiqi, Yang Zezhou, Gao Cuiyun, et al. An empirical study of knowledge distillation for code understanding tasks[PP/OL]. V1. arXiv (2025-08-21) [2025-09-04]. https://arXiv.org/abs/2508.15423.
[53] Laurens V D M, Hinton G. Visualizing data using t-SNE[J]. Journal of Machine Learning Research, 2008, 9(86): 2579-2605.
[54] Garreau D, Luxburg U. Explaining the explainer: A first theoretical analysis of LIME[C]//Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Palermo: PMLR, 2020: 1287-1296.

作者简介

张满青　男, 1997 年 7 月出生于山东省济南市。现为西北工业大学软件学院博士研究生。主要研究方向为自动程序定理证明、自动证明修复。
E-mail: zmqgeek@mail.nwpu.edu.cn

董云卫　男, 1968 年 7 月出生于云南省大理白族自治州。现为西北工业大学软件学院教授、博士生导师。主要研究方向为软件智能合成、自主决策模型测试、软件漏洞检测与修复、信息物理融合系统。
E-mail: yunweidong@nwpu.edu.cn

张　涛　男, 1982 年 3 月出生于宁夏回族自治区银川市。现为澳门科技大学计算机科学与工程学院教授、博士生导师。主要研究方向为智能化软件工程与软件安全。
E-mail: tazhang@must.edu.mo