数据采样是否改善了基于深度学习的能力漏洞检测？是啊！不！

徐杨、王少伟
加拿大马尼托巴大学
yangx4@myumanitoba.ca, shaowei.wang@umanitoba.ca

李毅、王少华
美国新泽西理工学院
yl622@njit.edu, davidshwang@ieee.org

第45届软件工程国际会议（ICSE）| 978-1-6654-5701-9/23/$31.00©2023IEEE|DOI: 10.1109/ICSE48619.2023.00192

摘要：深度学习（DL）的最新进展引发了人们对使用DL自动检测软件漏洞的兴趣，并在检测漏洞方面被证明了有希望的结果。然而，漏洞检测的一个突出和实际问题是数据不平衡。先前的研究发现，在现实世界的不平衡数据中，基于dl的最先进的（SOTA）的漏洞检测（DLVD）方法的性能急剧下降，而在所有研究方法中，f1分数平均下降了73%。如此显著的性能下降可能会禁用任何DLVD方法的实际使用。数据采样在缓解机器学习模型的数据不平衡方面是有效的，并已在各种软件工程任务中得到证明。因此，在本研究中，我们进行了系统和广泛的研究，从两个方面评估数据抽样对DLVD数据失衡问题的影响：i)DLVD的有效性，以及ii)DLVD正确推理的能力（基于真实脆弱的陈述做出决策）。我们发现，一般来说，过采样优于欠采样，原始数据采样优于潜在空间采样，通常原始数据上的随机过采样在所有研究数据中（包括高级一次采样和OSS）表现最好。令人惊讶的是，OSS并不能帮助缓解DLVD中的数据不平衡问题。如果继续进行召回，随机欠抽样是最好的选择。对原始数据的随机过采样也提高了DLVD方法学习真实脆弱模式的能力。然而，对于相当一部分情况（在我们的数据集中至少有33%），DVLD方法不能基于真正的脆弱陈述来解释他们的预测。我们为从业者和研究人员提供可操作的建议和路线图。

索引术语——漏洞检测、深度学习、数据采样、可解释的人工智能

Ⅰ.介绍

基于机器学习的VD方法吸引了研究界更多的关注，因为它们可以从之前的脆弱代码中自动从[1]-[5]中学习漏洞模式。特别是，深度学习（DL）的最新进展引发了人们对使用DL自动检测软件漏洞的兴趣。事实上，最近的研究已经证明了非常有希望的结果，在检测漏洞[3]-[5]的高精度。

漏洞检测的突出和实际问题是数据不平衡。现实世界项目中脆弱案例和非脆弱案例的比例极其不平衡。脆弱病例远少于非脆弱病例。先前的研究[3]观察到，最先进的（SOTA）基于dl的漏洞检测（DLVD）方法的性能在现实世界的不平衡数据中急剧下降。在所有模型中，f1的平均下降了73%。

如此显著的性能下降可能会禁用任何DLVD方法的实际使用。开发了各种方法来处理这个问题，如数据采样、成本敏感学习和集成方法。数据采样是一种广泛应用的技术，可以帮助解决缺陷预测[6]、[7]、软件质量预测[8]、软件变化预测[9]等软件工程任务中的数据不平衡问题。尽管在DLVD中普遍存在着不平衡问题，但目前还没有研究对DLVD中的数据不平衡问题进行系统的、广泛的数据抽样研究。因此，迫切需要了解DLVD中数据不平衡问题的数据采样。

在本文中，我们的目的是评估数据抽样对现有的SOTA DLVD方法的有效性及其学习脆弱模式的能力的影响。为此，我们使用三个基准数据集，对四种数据采样方法（随机欠采样/过采样、渗透[10]和OSS [11]）进行了广泛的研究，以及它们对四种SOTA DLVD方法的影响： Devign [12]、Reveal [3]、IVDetect [5]和LineVul [13]。请注意，一些先进的数据采样方法只能应用于已投影到潜在空间（特征空间）的数据点，因为它们需要在潜在空间中进行计算，比如流行的打击[10]。而简单的数据，如随机欠/过采样，可以应用于原始数据（没有任何投影）和潜在空间。一方面，潜在空间采样比对原始数据的采样更便宜，因为它节省了数据预处理的资源和时间(e。g., 数据清理和特征提取)和训练模型。另一方面，投射到潜在空间会导致信息丢失。目前还不清楚哪个策略更好。因此，我们也研究了这两种采样策略——对原始数据进行采样(即没有任何预处理和特征的原始代码
提取)和在潜在空间上的采样（即，代码的表示向量）。我们将我们的研究表述为以下研究问题：

RQ1：数据抽样是否提高了现有DLVD方法的有效性？

我们将各种数据采样方法应用于最先进的DLVD方法，并将其与没有抽样的方法进行了比较。结果：一般情况下，过采样

1558-1225/23/$31.00©2023IEEE2287
DOI 10.1109/ICSE48619.2023.00192

优于欠采样，对原始数据的采样优于对潜在空间的采样。对原始数据的随机过抽样优了所有其他研究的方法，包括高级打击和OSS。令人惊讶的是，如果DLVD方法最初在不平衡数据上表现不佳，那么OSS并不能帮助缓解DLVD中的数据不平衡问题。随机欠采样在提高不平衡数据集上的查全率方面表现最好。

RQ2：数据采样是否提高了DLVD学习脆弱模式的能力？

我们的目的是了解一个训练过的DL模型是否可以做出更多的预测决策推理

在函数中的真实脆弱语句比相同的DL

没有抽样方法的模型。我们使用可解释的人工智能技术（LIME [14]和GNNexlaner[15]）来解释DLVD方法的预测，并检查他们的决策是否基于真正的脆弱模式（即脆弱陈述）。结果：随机过采样提高了DLVD方法学习真实脆弱模式的能力。然而，在相当一部分情况下（在我们研究的数据集中至少有33%），DVLD方法不能基于真实的脆弱的陈述来进行预测。对于DLVD方法，学习真正脆弱模式的能力仍有提高的空间。

总之，我们的论文的贡献包括：

据我们所知，我们进行了第一个系统和广泛的研究，以评估数据抽样对DLVD数据不平衡问题的影响。我们使用三个基准数据集研究了四种数据采样方法和两种数据采样策略及其对四种SOTA DLVD方法的影响。我们进行了1680个实验，花费了超过10200个GPU小时。

我们为DLVD的从业者和研究人员提供了可操作的建议和路线图。是的，1：建议进行过采样，而不是过采样。是的，2：建议对原始数据进行采样，而不是对潜在空间进行采样。是的3：建议对原始数据进行随机过采样来处理DLVD比较中的数据不平衡问题。是的4：未来的研究建议开发新的数据增强，以提高DLVD方法学习真正脆弱模式的能力，因为仍有改进的空间。不确定性1：不建议使用OSS来处理DLVD中的数据不平衡问题。

一个复制包https://github.com/WIP2022/数据采样4DLVD为未来的研究和改进。

II .背景

在本节中，我们将介绍与基于深度学习的漏洞检测和数据采样相关的背景。

A.基于深度学习的漏洞检测概览

通常，DLVD方法包括三个阶段：特征提取、模型训练和模型部署。图1以绿色矩形表示了DLVD方法的一般框架。首先，在特征提取阶段，从代码单元中提取了各种特征，这些特性可以很好地捕获代码的语义和语法特性，以区分脆弱代码和非脆弱代码。其次，将提取的特征转换为一个重值向量，这是一个代码单元的紧凑表示（即表示学习）。提取了哪些特征以及如何学习表示取决于所选择的技术。当表示学习完成后，将选择一个二元分类模型来进行漏洞检测。我们将在第二节-B节中讨论关于具体表示学习的更多细节。在部署阶段，提取在训练阶段提取的相同特征，并将相同的表示学习技术应用于目标软件系统的代码单元，生成表示向量。

B.现有的基于dl的漏洞检测方法

一般来说，DLVD方法在两个方面有所不同：从代码单元中提取的特征类型(i。e., 并将提取的特征转化为a
表示向量（即，表示学习）。在此基础上，DLVD方法可以分为两类：基于标记的方法或基于图的方法。

基于标记：在基于标记的方法中，代码被认为是一个标记序列，并通过文本嵌入技术表示为一个向量(e。g., Word2Vec [16]，GloVe [17])。例如，LineVul [13]利用CodeBERT [18]将整个令牌序列嵌入到一个函数中，以进行漏洞检测。CodeBERT是一种基于转换器的模型，在各种语言的大量源代码语料库上进行预训练，可以使用注意机制来学习令牌之间的关系。与其考虑整个代码，不如使用VulDeePecker [2]和SySeVR [19]等方法，从代码中感兴趣的点中提取切片(例如，API调用、数组索引、指针使用等。)，并使用它们来进行漏洞检测，因为它们假设不同的代码行对于漏洞检测并不同等重要。

基于图的方法：除了考虑顺序标记中表示的语义信息外，基于图的方法还将代码视为图，并在生成表示向量时使用图神经网络（GNN）[20]将信息纳入不同的语法和语义依赖关系中。不同类型的句法/语义图。g., 可以使用抽象语法树（AST）、程序依赖图（PDG）、代码属性图（CPG））。例如，Devign [12]和Reveal [3]利用代码属性图（CPG）[21]来构建他们的基于图的漏洞检测模型。它们都使用门控图神经网络（GGNN）来表示代码。IVDetect [5]和LineVD [4]考虑脆弱的语句，并通过程序依赖图捕获它们周围的上下文。IVDetect使用特征注意GCN模型（FA-GCN）进行图嵌入，而LineVD使用图注意网络（GAT）模型进行图嵌入。不同的是，IVDetect执行功能级检测，而LineVD执行行级检测。

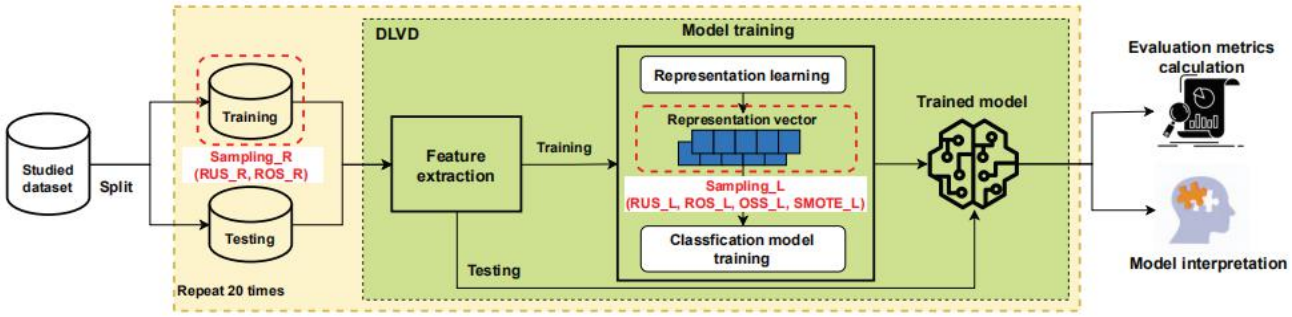

图1：我们的方法的概述。基于深度学习的漏洞检测框架为绿色矩形框架。我们标记了可以用红色圆矩形执行适当的数据采样方法的步骤。

表一：之前使用深度学习进行漏洞检测的方法。

方法	提取的特征	表示学习	分类器	检测水平
LineVul [13]	代币	代码BERT	MLP	函数线函数
VulDeePecker [2]	令牌+代码切片	word2vec	布尔斯特姆	功能
SySeVR [22]	令牌+代码切片	word2vec	布鲁	功能
Devign [12]	令牌+香蕈	word2vec + GGNN	MLP	功能
显示[3]	令牌+香蕈	word2vec + GGNN	MLP	功能
IVDetect [5]	令牌+广播+pdg	GloVe + FA-GCN	MLP	线条
LineVD [4]	令牌+PDG	CodeBERT + GAT	MLP	

在基于图和基于文本的模型中，一旦学习了代码单元的表示，向量将被传递到一个类符中进行最终学习。在以前的研究中已经研究了各种分类器。例如，IVDetect [5]和Reveal [3]使用多层感知器（MLP）作为分类模型。表一总结了上述DLVD方法的框架。

C.数据采样

一般来说，数据抽样方法可以分为两类：过采样和欠抽样。**采样不足。**过采样欠采样是指从数据集的多数类中删除数据点，使剩余的多数类与少数类[23]的数量近似相等。在漏洞检测的情况下，采样不足会从训练数据中删除一定数量的非脆弱案例，从而使脆弱/非脆弱案例的比例近似等于1。例如，随机欠采样（RUS）随机地从大多数类中删除案例，无论是否进行替换。Tomek链接定位所有跨类最近邻对，并删除多数类中最接近少数类[24]的相应情况。单边选择（OSS）结合了Tomek链接和密集的最近邻（CNN）规则[11]。然而，采样不足很有可能会发生丢弃有用的或重要的样本[23]，并导致信息丢失。

过采样增加了少数类中的实例数量，以平衡数据集[23]。例如，随机过采样（ROS）从少数类中随机选择案例，并进行替换，并将它们添加到训练数据集中。ROS已被证明是健壮的[25]。ROS的一个限制是，它没有为模型提供任何额外的信息。另一种流行的过采样技术称为合成少数族过采样技术（SMOTE）[23]，通过增加少数类的数据来解决这个问题。它首先随机选择一个少数类实例a，并找到其最近的多数类邻居b。然后，通过在特征空间中连接a和b的线上选择一点来创建合成实例。

原始数据采样vs。潜在空间采样一些先进的数据采样方法只能应用于已投影到潜在空间（或特征空间）的数据点，因为它们需要计算(e。g.，计算点之间的距离)在潜在的空间中，如打击。而一些简单的数据，如RUS和ROS，可以在原始数据（在我们的例子中是原始代码单元）上完成，而没有任何投影到潜在空间。为简单起见，我们将没有任何进一步处理的对原始数据执行的采样称为sampling_R，将对潜在空间中的数据执行的采样称为sampling_L。根据该定义，对原始数据和潜在空间都可以进行随机过采样和随机过采样，而打击和OSS只能在潜在空间上进行。在DLVD的背景下，我们可以对原始训练数据执行sampling_R，并在表示学习后对表示向量执行sampling_L，如图1所示。一方面，sampling_L比sampling_R更便宜，因为sampling_L节省了资源和时间来进行预处理（例如，数据清理、特征提取），并针对添加在少数类中的额外实例对主干模型进行训练。另一方面，投射到潜在空间会导致信息丢失。我们在rq中进行比较。为了简单起见，我们将应用于原始数据和潜在空间的相同采样方法作为本文其余部分的两种方法。例如，我们将潜在空间上的随机过采样称为ROS_L和原始空间

数据作为ROS_R。

III .实验设计

在本节中，我们将介绍我们的研究问题（RQs）、我们研究的数据集、DLVD方法、数据抽样方法和我们的RQs分析方法。

A.研究问题

我们的目的是回答以下研究问题：

RQ1：数据抽样是否提高了现有DLVD方法的有效性？

RQ2：数据采样是否提高了DLVD学习脆弱模式的能力？

数据采样在缓解缺陷预测[6]、[7]和质量预测[8]等各种软件工程任务中的数据不平衡问题方面的有效性。关于数据抽样对SOTA DLVD方法性能的影响的研究很少。在RQ1中，我们研究了数据抽样是否可以提高现有的SOTA DLVD方法的有效性。通过RQ1，我们可以为从业者提供基于其上下文选择合适的数据抽样方法的见解。

先前的研究表明，即使一个DL模型能够
正确地预测一个实例，预测并不总是基于真实的模式[3]，[14]。例如，在漏洞检测上下文中，假设DLVD方法正确地预测一个函数为脆弱。然而，模型做出决策的理由并不是函数中真正脆弱的陈述。换句话说，该模型并没有真正学习从区分脆弱代码到非脆弱代码的脆弱模式。因此，在RQ2中，我们的目的是了解数据采样是否可以帮助DLVD方法提高其学习真正脆弱模式的能力。

B.数据集

为了回答我们的研究问题，我们对三个流行的脆弱性数据集进行了研究，包括BigVul [26]、Reveal [3]和Devign [12]。BigVul [26]涵盖了2002年至2019年从300多种不同的样本中提取的CWEs
开源C/C++项目，并包含跨越91种不同漏洞类型的可信源代码漏洞。它包含+10K脆弱方法和+160K非脆弱方法。显示数据集[3]包含+12K方法和9。16%的弱势群体。Devign数据集[12]有从FFMPeg和Qemu项目中收集的+22K方法，其中45.0%的方法容易受到攻击。Devign数据集与其他两个数据集相比是平衡的。这些数据集在[3]-[5]，[12]，[13]的研究中，广泛用于评估各种DLVD方法。请注意，只有BigVul数据集有漏洞修复信息。

表二：所研究的数据集的概述。

数据集	#Vulnerablities	#Non 脆弱性	比率
显示[3]	1,664	10,547	1:9.9
Devign [12]	10,067	12,294	1:1 2
BigVul [26]	10,547	168,752	1:16.3

C.研究了DLVD和数据抽样方法

DLVD。我们根据两个标准选择了DLVD方法。首先，DLVD应该代表DLVD方法的两种类型，即基于标记的方法和基于图的方法
如第II-B节所述。其次，这些方法应该在最近才被开发起来，并利用深度学习技术。该标准背后的目标是促进我们的结果对SOTA方法的通用性和适用性。我们选择了Devign [12]、Reveal [3]和IVDetect [5]作为基于图的方法的代表。对于基于标记的方法，我们选择了LineVul [13]作为代表，因为它是利用CodeBERT [18]的SOTA方法，并且已经被证明比其他方法更有效(e。g.，偏离、揭示和自我检测)。请注意，LineVul可以同时用于执行功能级和线级的漏洞检测。在本研究中，我们使用LineVul来进行函数级的一致性检测。

数据采样。我们选择了基于两个标准的数据抽样方法。首先，所选择的方法应涵盖本节II-C中的两类抽样方法，即过采样和欠采样。其次，所选择的方法可以同时应用于原始数据和潜在空间。因此，我们选择了随机欠采样（RUS）和随机过采样（ROS）作为我们的研究方法。此外，我们还希望包括除ROS和RUS之外的高级方法。我们选择单侧选择（OSS）作为欠采样的代表，并选择合成少数过采样技术（SMOTE）作为过采样的代表。请注意，OSS和SMOTE只能应用于第II-C节中所讨论的潜在空间。

D.评价指标

我们考虑了两类类型的评估指标来进行评估
DLVD方法的有效性。首先，考虑脆弱性检测作为一个二元分类任务，我们考虑了四个流行的评价指标[27]——cele（R的简称）、精度(P)、F1和曲线下面积（AUC），根据之前的研究[2]、[3]、[12]、[28]、[29]。其次，考虑到从返回的列表中审查潜在的脆弱代码的成本，我们使用流行的评估度量对算法[5]进行排序，[30] -精度@k（P@k的缩写），
计算结果为，k是返回列表的大小。$\frac{true\ positive}{k}$我们选择了k，分别为10、20、50和100。

E.RQ的方法

我们首先在我们的研究中介绍了某些设置。*我们将在一个数据集上对一个DLVD方法的数据采样方法的评估作为一个实验实例。*为简单起见，我们使用“{DLVD}+{数据集}”格式来表示其余部分中的一个特定实例。因此，我们有12个实验实例，即4个DLVD * 3数据集。

对于每个实验实例，我们有以下三种数据采样设置来应用和比较它们对DLVD方法的影响：

无采样：我们在不应用任何数据采样方法的情况下，在原始训练数据上训练DLVD方法，并对测试数据进行评估DLVD方法。Sampling_R：我们对原始训练数据（如图1中红色圆矩形所示）应用数据采样方法（如RUS和ROS），并对采样的训练数据进行DLVD方法训练。然后，我们在测试数据上评估DLVD方法。我们将原始数据上的RUS和ROS分别称为RUS_R和ROS_R。

Sampling_L：此设置类似于Sampling_R。唯一的区别是，我们没有对原始训练数据应用数据抽样方法，而是应用数据抽样(i。e., 对训练数据的表示向量（潜在空间）进行了研究。我们将潜在空间上的RUS、ROS、OSS、OSS和SMOTE分别称为RUS_L、ROS_L、OSS_L和SMOTE_L。

用于生成训练和测试数据的数据分割涉及一个随机过程，为了缓解随机性造成的偏差，我们按照之前的研究[31]-[33]重复该过程20次，并计算每个评估指标的20个分割的平均值作为我们的结果。根据之前的研究[3]，[34]，[35]，我们将训练和测试分为80%/20%。

RQ1的1)方法：在RQ1中，我们的目标是检验数据采样是否可以提高DLVD的有效性，如果是，哪种数据采样方法表现最好。为此，我们根据我们所研究的评估指标对每个实验实例中的数据抽样方法进行排序（详见第三-d节）。例如，假设我们在揭示数据集上应用所研究的采样方法来训练IVDetect模型(i。e., IV检测+显示)。在这个实验实例中，ROS_R在F1方面表现最好（排名第1），我们注意到ROS_R在IVDetect + Reveal实例中在F1方面排名第1。为了在所研究的实验实例中获得数据抽样方法的总体秩，我们计算了一个平均秩。例如，假设ROS_R在6个实例中排名第一，在另一个实例中排名第二
6个例子，那么它的平均排名是1.5。因此，我们可以根据不同的评价指标来比较数据抽样方法的排名。排名越小，表示效率越高。我们还通过比较每种抽样方法所获得的实验实例的数量来研究哪种抽样方法表现最好。

RQ2的2)方法：为了了解训练后的模型是否在一个函数中对真实的脆弱语句进行了正确的预测决策推理，我们选择了使用可解释的人工智能技术。我们研究的三个基于图的模型都使用gnn来捕获与图相关的属性。因此，我们选择了Reveal模型作为基于图的方法的代表，并选择了LineVul作为基于文本的方法的代表。我们选择了BigVul数据集，因为只有这个数据集为我们提供了脆弱代码的行信息，也就是说，使代码易受攻击的特定语句。因此，我们对两个实验实例，即LineVul + BigVul和Reveal + BigVul进行了分析。

我们使用了局部可解释模型不可知论解释

表三：所研究的dlvd的参数设置。

参数	IV检测	邪恶的	揭示	LineVul
埃普克斯	50	50	50	4
学习率	1e-4	1e-4	1e-4	5e-5
GNN_Layer	4	6	6	NaN
图嵌入大小	100	200	200	NaN
特征嵌入大小	100	100	100	768
总参数	575K	924K	402K	123M

（LIME）[14]是一种广泛使用的模型不可知的可解释算法，用于解释深度学习模型，用于解释LineVul。LIME返回对预测做出重要贡献的令牌。对于我们的案例，我们将LIME的结果与函数中真正的脆弱语句进行了比较。更具体地说，我们为LineVul设计了以下实验：我们选择了BigVul测试集中由LineVul预测的真阳性（TP）情况（正确预测的脆弱函数），并使用LIME对这些情况进行分析。在LIME的结果中，对于每个真正的阳性情况，我们选择了驱动模型的前k个最重要的标记来做决定，看看这些标记是否在脆弱的语句中。如果至少有一个前k在脆弱语句中，我们将它视为命中，并比较非抽样和所有抽样方法之间对所有TP情况的命中率。我们考虑了k的1、35和10。

我们使用GNN解释器[15]来解释之前的研究[5]之后的揭示。同样，我们解释了由Reveal模型预测的真实阳性（TP）病例。GNNExclaner提供了图中每条边的重要性。Reveal使用了代码属性图（CPG），每一行代码都是图中的一个节点，我们在CPG上应用了GNN-解释器。GNNExcrener并没有直接告诉我们哪些节点是重要的，因此我们认为由GNNExcraner返回的重要边连接的节点是图中的重要节点。与Lime的实验类似，我们选择了前k个最重要的边及其连接节点，这些节点驱动模型来预测函数的脆弱性，并看看这些节点（代码行）是否在函数的脆弱性语句中。

F. 实施细节

我们在加拿大计算Narval HPC [36]上运行了我们的实验，使用英伟达A100图形处理器和一个Linux服务器与四个英伟达RTX 3090图形处理器，AMD锐龙48核CPU与256 GB内存。我们使用了在GitHub存储库中研究的DLVD方法的实现。对于LIME和GNN解释器，我们使用了GithubLime和dgl-GNN解释器[37]中的实现。

我们使用原始论文中推荐的参数对DLVD方法进行了微调。参数设置见表三。注意到，我们在LineVul模型中使用了5e-5的学习率，因为它是一个预先训练过的模型，我们对该模型的训练可以被认为是对下游任务-DLVD的编译码的微调。因此，学习率低于其他模型。我们的实验涉及12个实验实例，每个实例

实验实例我们对数据有7种抽样方法（不抽样加上各种抽样方法）来检验。因此，我们有84个（7*12）个组合。对于每种数据抽样方法，我们运行了20次，以获得可靠的结果。我们总共运行了12x7x20 = 1,680个实验，这花费了超过10,200个GPU小时。

增值结果

A. RQ1：数据抽样是否提高了现有DLVD方法的有效性？

表四：每种数据抽样方法在不同评价指标下的平均排名。对于每个度量，我们突出显示比非采样小的记录，较深的颜色表示更好。

取样	AUC	P	R	F1	P@10	P@20	P@50	P@100
不抽样的	3.9			3.3	6.8	6.6	4.0	3.8
RUS_R	4.2			4.3				
ROS_R	3.0			4.6	2.3	3.3	4.5	4.7
SMOTE_L	4.4			4.2				
OSS_L	3.4			2.9	4.3	2.8	2.4	2.5
RUS_L	3			2.5				
ROS_L	5.3			5.0	3.3	3.3	4.3	3.9
	3.8			3.9				
				4.8	4.5	4.8		
				2.5	3.8	4.2		
				4.0	3.8	3.8		

表V给出了12个实验实例中各种数据抽样方法在各种评价指标方面的详细结果。我们观察到，几乎所有的数据采样技术都提高了F1中所有实验实例的有效性，这是二进制分类任务的一个主要评估指标。我们在表四中观察到相同的结果，其中显示了每种研究数据抽样方法的平均秩，包括12个实验实例中的无抽样，所有研究数据抽样
该方法在f1方面的秩小于不抽样方法。在不平衡数据集（Reveal和BigVul）上观察P@k时，也观察到类似的结果。例如，在P@10方面，除了与Reveal模型相关的实例外，至少有一种数据采样方法在所有实验实例中都优于无采样。

发现1：一般来说，数据采样提高了所有研究模型的有效性，特别是在不平衡的数据集上。

在表四中，只有ROS_R在所有评估指标方面都优于不抽样。在表V中，我们观察到ROS_R在F1的12个实例中赢得了第7个（排名第1位）。如果只考虑不平衡的数据集，ROS_R在F1的8个实例中赢得了7个，除了Devign + Reveal。ROS_R在所有实验实例中改进了F1的无采样。虽然ROS_R不是Devign + Reveal中最好的，但它仍然从0提高了31.9%。188年到0.248年。在不平衡数据集上的8个实验实例中，当考虑到ROS_R优于无采样的改进时，ROS的无采样至少提高了31.9%。在P@10方面，ROS_R在不平衡数据集上的8个实例中有5个优于无采样。令人惊讶的是，ROS_R在所有评估指标上都比SMOTE_L好得多，这是一个高级抽样
该方法至少可以与之前的[38]-[40]研究中报道的随机过抽样相比较。

发现2：令人惊讶的是，简单的方法ROS_R优于所有其他研究的抽样方法（包括先进的SMOTE_L），而且它显著改进
DLVD方法的有效性
研究指标。

当比较过采样和欠采样时，我们观察到除表IV中的AUC和R外，ROS_R在所有指标上都优于RUS_R。同样地，ROS_L在所有的P@k和R上都优于RUS_L。这样的发现是意料之中的，因为欠采样从数据集中删除数据点，这将导致信息丢失，而过采样保留所有的原始数据。有趣的是，当观察召回率的改进时，我们观察到RUS_R在不平衡数据集（Reveal和BigVul）上的所有实验实例中都取得了最好的性能，除了LineVul + Reveal，其中RUS_L表现最好。虽然在某些情况下，RUS_R不是最好的，但RUS_R仍然提高了所有实验实例的召回率，至少比无采样提高了10%。我们的发现与之前的研究[41]相一致，即欠采样实现了最大的回忆改善。我们在第V-B节中调查了这背后的原因。

发现3：一般来说，过采样优于欠采样，而随机欠采样在提高不平衡数据集上的召回率方面表现最好。

当比较sampling_L和sampling_R时，我们观察到ROS_R在表4中的所有指标方面都优于ROS_L。对于RUS，RUS_R在所有P@k方面的排名与RUS_L相似，但RUS_R在AUC、P、R和F1方面都优于RUS_L。因此，一般来说，sampling_R的性能优于sampling_L，通常是对于ROS。我们推测可能的原因有两方面。首先，将代码投影到潜在空间会导致信息丢失，并带来比较偏差
使用原始代码。在潜在空间上的采样放大了这种偏差。其次，由于采样是在表示学习之后应用的，换句话说，数据采样在表示学习过程中没有帮助。

发现4：一般来说，sampling_R优于sampling_L。

表1显示，对于模型揭示和Devign，它们在原始数据集上表现相对较好(e。g.，)，OSS有助于提高他们的性能（例如，在F1方面）。虽然OSS不能帮助DLVD方法处理模型IVDetect和LineVu的数据不平衡，但它们在不平衡的数据集上表现不佳(i。e.，揭示和BigVul)在无采样场景中。例如，IVDetect和LineVul在不平衡的数据集（Reveal和BigVul）上表现较差。e.，所有指标的值几乎等于0。原因是这两种模型对训练数据的不平衡没有容忍度，它们预测所有的数据点都是非脆弱的。应用程序

表五：每种数据抽样方法在不同的评估指标方面的结果，包括不抽样。为了表明在每个度量中哪种采样方法比无采样更有效，比无采样值更大的记录用绿色突出显示，而较深的颜色表示更好。增加了f1中每个实验实例的最佳记录。

	DLVD 取样	IVDetect AUC	P	R	F1	P@10	P@20	P@50	P@100	邪恶的 AUC	P	R	F1	P@10	P@20	P@50	P@100
邪恶的	不抽样的	0.573	0 0	0.494	0.512	0.	0.65	0.64	0.67	0.569	0	0.479		0.788	.763	0 0	0 0
	RUS_R	0	0	0.523	0.547	7	0.6	0.62	0.65	0 0	0 0	0.532	0 0	0.62	.59.	0 0	0 0
	ROS_R	0.59	0.53	0.544	0.535	0.7	0.75	0.66	0.68	0 0	0 0	0 0	0.51	0.7	7.73	0 0	0 0
	SMOTE_L	0.555	4.52	0.858	**0.61**	0 0	0.631	0.626	0.618	0	0	0 0	4.49	0.74	.67.	0.67	0.64
	OSS_L	0 0	4.52	0.549	0.542	0	0.621	0.614	0.611	0.56	0.50	0.48	5.52	0.64	660	8.64	8.60
	RUS_L	0.60	8.48	0.538	0.537	0.66	0.645	0.615	0.62	4.56	1.50	8.55	7	0.74	0 0	8.67	8.64
	ROS_L	8.60	3	0.533	0.535	7.62	0.645	0.626	0.619							8.68	
			0.535													.684	
			0.537													.652	
			0.537													.668	
揭示	不抽样的	0.402	0.008	1E-04	3E-04	0	0	0	0	0.711	0 0	0 0	0.188	0.838	0.775	0.625	0.541
	RUS_R	0.718	0.196	0.688	0.305	0	0.3	0.3	0.31	0 0	0 0	0	0.278	0.4	0.3	0.325	0.33
	ROS_R	0.71	0.286	0.492	**0.46**	0	0.45	0.44	0.43	0 0	0 0	0 0	0.248	0.867	0.839	0.813	0.786
	SMOTE_L	0.624	0.142	0.593	0.228	0.4	0.303	0.242	0.22	0	0.27	0	0.304	0.914	0.886	0.814	0.75
	OSS_L	0.38	0	0	0	0	0	0	0	0.70	4.17	0.18	0.279	0.943	0.907	0.84	0.741
	RUS_L	0	0	0.661	0	0	0.305	0.245	0.213	1.69	8.21	8.65	**0.322**	0.886	0.836	0.789	0.734
	ROS_L	0.61	0.13	0.626	0.22	0.26	0.303	0.237	0.22	7.69	3.24	6.3.	0.31	0.943	0.921	0.834	0.733
										5.70	8.27	397.					
										6.70	3.25	287.					
										8.69	.25	459.					
										7		414					
比格维尔	不抽样的	0.461	0	0	0	0	0	0	0	0 0	0.326	0 0	0.247	0.68	0.72	0.668	0.586
	RUS_R	0.679	0.111	0.628	0.188	0	0.2	0.14	0.12	0 0	0	0 0	0.174	0.2	0.18	0.156	0.146
	ROS_R	0.77	0.244	0.622	**0.347**	0	0.1	0.14	0.15	0 0	0 0	0 0	**0.271**	0.48	0.42	0.472	0.488
	SMOTE_L	0.568	0.08	0.507	0.137	0.2	0.088	0.08	0.097	0.70	0 0	0.20	0.269	0.614	0.65	0.631	0.557
	OSS_L	0.447	0.1	5E-05	1E-04		0.003	0.001	5E-04	1.68	0.1.	1.68	0.253	0.714	0.686	0.611	0.577
	RUS_L	0	0.075	0.584	0.005		0.088	0.086	0.094	7.70	295.	2.25	0.268	0.714	0.714	0.646	0.571
	ROS_L	0.56	0.078	0.541	0	0	0.088	0.084	0.097	3.71	259.	3.28	0.27	0.7	0.714	0.654	0.6
					0.13	0.06				7.71	316.	4.22					
										2.71	221.	1.35					
										1.72	232	4.33					
										3		3					

	DLDV 取样	显示F1 AUC	P	R	P@10		P@20	P@50	P@100	LineVul AUC	P	R	F1	P@10	P@20	P@50	P@100
邪恶的	不抽样的	0.555	0 0	0.42	0.4	0.525	0.5	0.438	0.444	0.71	0 0	0 0	0.573	0.942	0.942	0.934	0.918
	RUS_R	0 0	0	0.545	**0.519**	0 0	0.47	0.446	0.46	0 0	0 0	0 0	**0.592**	0.989	0.981	0.973	0.967
	ROS_R	0 0	0.51	0.492	0.497	0 0	0.445	0.416	0.418	0 0	0 0	0 0	0.58	0.968	0.971	0.973	0.965
	SMOTE_L	0	2.49	0.511	0.502	0	0.458	0.474	0.467	0	0.61	0.53	0.586	0.95	0.95	0.937	0.919
	OSS_L	0.55	9.50	0.547	**0.519**	0.47	0.45	0.468	0.458	0.70	6.59	7.59	0.573	0.95	0.945	0.937	0.912
	RUS_L	3.55	8.49	0.525	0.508	.49.	0.435	0.436	0.44	4.70	5.60	.557	0.585	0.965	0.943	0.932	0.916
	ROS_L	3.54	5	0.506	0.5	42.4	0.445	0.442	0.453	5.66	6.56	.618	0.558	0.965	0.953	0.929	0.912
		.54.	0.494			4.40				1.66	3.56	.587					
		541.	0			5.41				3.66	4.56	.616					
		541	0.49			5				1.66	.568	.554					
			3.49														
			6														
揭示	不抽样的	0.583	0.282	0.119	0.159	0.625	0.556	0.56	0.484	0.775	0	0	0	0	0	0	0
	RUS_R	0.674	0 0	0.594	0.273	0 0	0.45	0.415	0.395	0.841	0.4	0.24	0.29	0.73	0.72	0.68	0.62
	ROS_R	0.636	0 0	0.414	**0.292**	0 0	0.61	0.588	0.576								5.49
	SMOTE_L	0.621	0	0.311	0.235	0	0.343	0.237	0.215								1
	OSS_L	0.636	0.17	0.211	0.219	0.42	0.318	0.255	0.244	0.811	0.442	0.506	0.478	0.905	0.963		
	RUS_L	0.676	8.23	0.557	0.277	5.61	0.325	0.34	0.3	0.948	0.90						
	ROS_L	0.63	.192	0.311	0.24	.415	0.265	0.239	0.203	0 0	0 0	0.51	0 0	0 0	0 0	0 0	0 0
			.245			.34.				0	0	0 0	0	0	0	0	0
			.187			31.2				0.71	0.30	0.24	0.35	0.73	0.74	0.68	0.62
			.197			9				6.72	1.42	8.59	5.29	5.72	.742	7.68	5.62
																	6.64
																	6.64
																	7
比格维尔	不抽样的	0.679	0.387	0.109	0.168	0.72	0.7	0.648	0.578	0.554	0.07	0.002	0.005	0.07	0.07		
	RUS_R	0.684	0 0	0.617	0.182	0 0	0.23	0.248	0.242	0.063	0.046						
	ROS_R	0 0	0 0	0.304	**0.261**	0 0	0.5	0.477	0.444	0 0	0 0	0 0	0.266	0 0	0 0	0 0	0 0
	SMOTE_L	0 0	0	0.276	0.224	0	0.375	0.394	0.398	0 0	0 0	0 0	**0.406**	0 0	0 0	0 0	0 0
	OSS_L	0.66	0.10	0.224	0.235	0.28	0.43	0.437	0.44	0	0	0	0 0	0	0	0	0
	RUS_L	1.61	7.23	0.564	0.189	.457	0.29	0.273	0.279	0.82	0.16	0.72	0	0.59	0.61	0.62	0.59
	ROS_L	7.63	2.19	0.285	0.23	.34.	0.323	0.338	0.353	5.74	4.45	2.36	0.11	.947	8.95	7.94	2.90
		2.66	.248			41.2				5.53	5.06	7.58	2.00	.175	8.14	3.13	4.12
		4.62	.114			9.33											
			.193			5											

OSS_L并不能帮助提高IVDetect和LineVul的有效性。为什么OSS不能很好地平衡数据？OSS依赖于Tomek链接[24]和压缩最近邻（CNN）[42]来决定删除哪些非脆弱的数据。但是，这种被删除的非脆弱数据点必须满足某些标准。如果没有发现更多符合标准的数据点，OSS将停止删除非脆弱的数据，即使在运行OSS后，数据仍然保持不平衡。因此，OSS并不能保证将多数阶级的规模缩小到少数阶级。我们通过应用OSS_L对这些数据集的Reveal和BigVul进行了研究，发现这些数据集仍然是非常不平衡的。Reveal数据集的脆弱性和非脆弱性的比例为1：8.7，BigVul数据集的比例为1：16。

发现5：令人惊讶的是，如果DLVD接近，OSS并不能帮助缓解DLVD中的数据失衡问题。e.，）最初在不平衡的数据

B. RQ2：数据采样是否提高了DLVD学习脆弱模式的能力？

表六：实验实例中各种数据采样方法在LineVul + BigVul和Reveal + BigVul中的命中率。

	LineVul + BigVul				揭示+ BigVul			
	前1名	**前3名**	**前5名**	**前10名**	**前1名**	**前3名**	**前5名**	**前10名**
无取样	0.39	0.55	0.66	0.66	0.55	0.76	0.81	0.90
ROS_R	0.41	0.62	0.72	0.82	0.67	0.83	0.89	0.92
RUS R	0.32	0.45	0.56	0.71	0.69	0.85	0.91	0.95
ROS L	0.22	0.39	0.52	0.68	0.65	0.84	0.90	0.95
RUS_L	0.15	0.37	0.50	0.66	0.65	0.86	0.91	0.95
smote_l	0.13	0.35	0.47	0.63	0.67	0.84	0.90	0.95
OSS L	0.32	0.53	0.61	0.72	0.69	0.84	0.90	0.95
ROS R 2X	0.47	0.67	0.76	0.85	0.67	0.83	0.89	0.93
ROS_R_4X	0.45	0.66	0.75	0.85	0.64	0.86	0.91	0.94

在LineVul + BigVul和Reveal + BigVul中，各种数据抽样方法的命中率如表六所示。在Reveal + BigVul中，所有研究的数据抽样

与不抽样方法相比，该方法提高了命中率。在LineVul + BigVul中，只有ROS_R提高了命中率。一种可能的解释是，LineVul对不平衡数据的BigVul学习不好，只有ROS_R帮助处理数据不平衡（见表V中的LineVul对BigVul的有效性）。然而，ROS_R在这两个实例上都改进了降采样能力。例如，在ROS_R之后，当k=1、3、5和19时，LineVul +的命中率从0.39、0.39、0.55、0.66和0.66增加到0.41 0.62、0.72和0.82，分别提高了5.6%、12.7%、9。分别为1%和24.2%。同样，当k=1、3、5和10时，ROS_R将Reveal + BigVul中的命中率分别提高了21.8%、8.6%、9.4%和2.4%。这些结果表明，ROS_R有助于LineVul和Reveal更好地了解脆弱模式的特征。例如，图2显示了ROS_R前后一个脆弱函数的LIME结果。该函数在用绿色矩形[43]标记的行处有一个无后使用漏洞。在ROS_R之前，LineVul根据蓝色突出显示的声明预测它为非脆弱线，而在ROS_R之后，LineVul能够正确地基于橙色突出显示的脆弱线进行预测。**观察结果可能表明，原始训练数据集中的重复数据点对模型不一定有噪声，它们可以对模型产生积极的影响，以学习正确添加时检测漏洞的真实模式。**为了验证这一假设，我们增加了脆弱病例和非脆弱病例的比例(比率维尔)从1：1到2：1，和4：1使用ROS_R，并比较他们的命中率。不同比例下的结果如表六所示维尔，i.e.，ROS_R为1：1，ROS_R_2X为2：1，ROS_R_4X为4：1。我们可以看到，前1名的命中率从0.41增加到0.47维尔从1增加到2，当比例上升时略有下降维尔对于LineVul，可达到4（0.45）。同时，增加重复次数后，LineVul和Reveal的有效性也随增加了，即应用ROS_R、ROS_R_2X和ROS_R_4X后，LineVul的F1值分别为0.406、0.42和0.42。我们观察到类似的现象。

我们进一步进行了额外的分析，以了解不同乘法的影响(i。e.,在数据集BigVul（LineVul+BigVul）的模型LineVul上的2倍和4倍)的脆弱情况。结果显示，F1得分分别为0.406（ROS_R）~0.420（ROS_R_2x）和0.418（ROS_R_4x）。我们观察到AUC的相同趋势。前1的命中率达到2倍达到最佳值，之后略有下降。过多的脆弱病例的倍增可能会使分类产生偏差，并降低模型的有效性。结果表明，ROS_R_2x对LineVul+BigVul实例效果最好。结果表明，适当增加重复脆弱病例的规模，可以提高学习脆弱模式的有效性和能力。

发现6：ROS_R提高了DLVD方法学习真正脆弱模式的能力。

虽然我们的研究结果表明，ROS_R可以提高DLDV捕获脆弱模式并基于其进行预测的能力。即使在应用数据抽样，它仍有未来改进的潜力。如果我们看前1名的结果，LineVul+BigVul和Reveal + BigVul的命中率分别为0.39和0.55。即使在应用RUS_R后，命中率也会提高到0.42和0.67。仍有相当一部分情况下，该决定并不是真正根据脆弱的声明作出的。58%和33%的脆弱功能被预测为脆弱的功能而不是基于他们对LineVul和揭示的真正脆弱的声明。

发现7：在相当一部分情况下（至少33%），DVLD方法不能使他们的预测相对于真正脆弱的陈述。对于DLVD，学习真正脆弱模式的能力仍有提高的空间

V.讨论

A.为什么ROS_R在所有被研究的方法中表现最好？

在RQ1中，我们观察到，ROS_R在跨实验实例的所有研究指标中，在所有研究方法中表现最好，通常是对于不平衡的数据集。为什么RQS_R是国王？在某种程度上，RQ2中的结果提供了见解，i。e.，ROS_R提高了DLVD方法学习真正脆弱模式的能力。我们推测ROS_R与其他方法相比表现最好。首先，与采样不足相比(i。e.，RUS_R)，而不是从训练数据中删除导致信息丢失的非脆弱代码，ROS_R，因为过采样方法没有这个问题。其次，与其他在潜在空间上的采样方法相比，ROS_R对原始数据进行采样。我们推测，sampling_R比sampling_L更能区分脆弱病例和非脆弱病例。例如，为了说明ROS_L和ROS_R潜在空间数据分布的影响，我们使用t分布随机邻居嵌入（t-SNE）[44]可视化数据点的分布时使用ROS_L和使用ROS_R LineVul+BigVul，F1值ROS_R和ROS_L 0.406
0.分别为121。图3为应用ROS_R和ROS_L后数据点的分布情况。应用ROS_R时比应用ROS_L更容易分离脆弱和非脆弱数据。

B.为什么随机抽样不足会提高回忆率？

召回度量DLVD方法识别所有易受攻击的代码的程度。先前的研究表明，在代码语料库[45]-[47]中有相当一部分类似的代码。因此，我们假设围绕脆弱案例可能存在大量类似的非脆弱案例，并引入显著的噪声来识别模型来识别各种脆弱代码。随机欠采样减少了这种噪声，

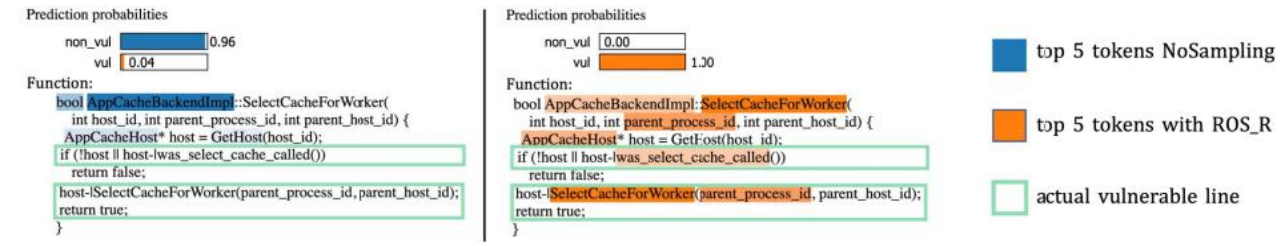

图2：LIME对脆弱功能的解释，在ROS（左）被错误预测为非脆弱，在ROS_R（右）被预测为正确预测。

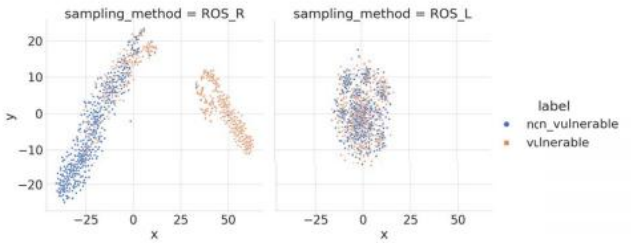

图3：应用ROS_R和ROS_L后，实验实例LineVul+BigVul中数据点的分布。

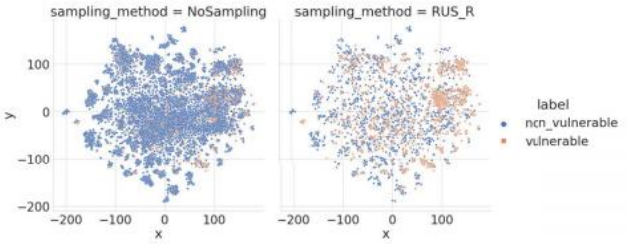

图4：没有（无采样）和不应用RUS_R的揭示数据集中的数据点的分布。

因此，它提高了查全率，尽管它由于信息丢失而降低了精度。

为了验证我们的假设，我们对揭示模型的实验实例进行了一个案例研究。我们将t-SNE应用于由Reveal模型生成的表示向量上，并可视化了RUS_R前后表现最好的点的分布。为了定量验证我们的假设，我们还计算了生成的t-SNE图中每个脆弱点的平均邻居数和中等邻居数。给定地图大小为400*400，我们将所有与脆弱点的欧氏距离小于4的非脆弱点作为其邻居。我们讨论了由阈值选择(i。e.,4）在第V-D节中。

图4显示了Reveal + Reveal的数据点分布（由于空间限制，我们没有显示另外两个数据集的图）。我们观察到，在不应用RUS_R（无采样）的情况下，非脆弱点覆盖了大部分的非脆弱点，脆弱点和非脆弱点几乎混合在一起。应用RUS_R后，

表七：与Reveal模型相关的实验实例中，RUS_R前后的邻居数。

数据集	邪恶的		揭示		比格维尔	
取样	平均	中位数	平均	中位数	平均	中位数
NoSampling	6.68	6	38.11	38	41.43	43
RUS_R	5.43	5	3.60	3	2.60	2

在脆弱点和非脆弱点之间观察到更好的分离。表7为RUS_R前后每个脆弱点邻居的中位数/平均数。在所有研究的数据集上，平均/中位数显著减少，但不同数据集的减少比例不同。在平衡数据（Devign）上，平均值从6.68降低到5.43，而在不平衡数据集上，
平均值从38减少了。11至3.6和41.43至2.6上
显示和BigVul数据集的减少超过10倍。我们观察到回忆的改善和减少的大小之间的相关性。在BigVul和Reveal数据集上，召回率分别提高了464%和397%，而在Devign上仅提高了12.4%。这些结果在一定程度上解释了为什么随机欠采样可以减少脆弱代码的噪声，从而提高召回率。

C.我们的发现的含义

我们建议未来的从业者使用过度抽样过欠采样，sampling_R过sampling_L，通常使用ROS_R来处理DLVD中的数据不平衡问题。在RQ1中，我们观察到过采样通常优于欠采样，sampling_R优于sampling_L。通常，ROS_R在所有被研究的抽样方法中表现最好。此外，我们观察到，ROS_R可以提高在DLVD方法（包括基于文本和基于图形的模型）的源代码中学习真实脆弱模式的能力。因此，建议使用ROS_R来处理DLVD中的数据不平衡问题。

我们不建议从业者对DLVD中不平衡的数据使用OSS_L。在RQ1，我们观察到在所有研究抽样方法，OSS_L是唯一一个不帮助提高DLVD不平衡数据集由于其机制不能保证真正平衡目标数据集，通常当脆弱和非脆弱的代码之间的比率很大。

因此，我们不建议从业人员对不平衡的数据使用OSS_L来训练DLVD。n

我们建议未来的从业者使用RUS_R，如果他们希望改善召回情况。在RQ1中，我们观察到欠采样方法，通常是RUS_R，可以提高DLVD方法的召回率。在V-B节中，我们的分析表明，RUS_R显著减少了脆弱点周围类似的非脆弱点的数量，以降低噪声。在实践中，如果实践者的目标是改善
回顾一下他们的方法，他们可能可以考虑使用RUS_R。

我们鼓励未来的研究开发新的数据增强技术，以提高DLVD方法学习真正脆弱模式的能力。在RQ2中，尽管我们的研究结果显示ROS_R可以提高DLDV学习脆弱模式的能力。然而，仍有相当数量的案例（LineVul+BigVul和Leveal+BigVul中58%和33%的脆弱功能），其中决策不是基于脆弱的声明做出的。即使在应用数据抽样，它仍有未来改进的潜力。我们的发现为解决/缓解这个问题提供了线索。例如，数据增强可能是一个有价值的研究方向，因为我们的结果显示，简单的重复策略(e。g., ROS_R)在RQ2中很有帮助。

D.对有效性的威胁

内部有效性：一个威胁与超参数设置有关
当训练所研究的DLVD方法时。因为超参数调优对于我们所研究的包含数百万个参数的方法来说是极其昂贵的。我们使用了以前研究中推荐的参数。此外，由于数据集处于函数级别，而不是提交级别，因此本文没有考虑时间级评估场景（使用早期数据用于训练，而使用后期数据用于测试）。需要注意的是，研究方法（IVDetect、devg、Reveal和LineVul）的结果与原始论文中报道的结果并不完全相同。一种可能的解释是，我们进行了20次实验，取了一个平均值，而他们报告的是最好的一次。为了迁移威胁，我们重用了作者发布的实现，并遵循了他们的论文中指定的DLVD方法的实验设置。然而，我们的目标不是比较这些DLVD方法的有效性，而是我们专注于研究数据抽样方法对DLVD方法的影响。在V-B节中，我们选择4作为阈值来计算脆弱病例的邻域，这可能会对结果带来偏差。为了迁移威胁，我们使用不同的阈值进行了相同的实验，我们的发现仍然成立。同样，在RQ2中，我们选择检查LIME和GNNExcrener返回的k令牌，不同的k可能导致不同的结果。为了迁移这种威胁，我们检查了k的不同值1、3、5、10，发现发现适用于不同的k。此外，最近的研究LIME可能过于随机，可能产生重要标记[48]的不一致结果，并可能给我们的研究带来偏差。

我们鼓励未来的研究尝试更先进的技术来解释。我们发布了我们的复制包，以提高我们工作的透明度。
外部效度对外部效度的威胁与我们的研究结果的普遍性有关。我们的发现可能不能推广到其他数据集，DLVD方法，数据抽样方法。我们鼓励未来研究更多的方法和数据集。

VI .相关工作

A.基于机器学习的漏洞检测

随着机器学习的快速发展及其在各个领域的成功应用，通常是深度学习。研究人员已经开始研究机器学习和基于深度学习的漏洞检测方法。基于机器学习的脆弱性检测方法的早期研究通常需要人类定义的特征来识别脆弱性，然后使用机器学习模型来训练这些特征[1]，[49]。例如，斯坎达里亚托等人。[1]使用文本挖掘技术，并利用特定术语的出现频率进行漏洞检测。与机器学习方法不同，基于深度学习的方法通常不预先定义特征，而是让模型通过更深层次的网络自己提取特征进行学习。有关之前的DLVD方法的更多细节，请参见第二节-A节。纳皮尔等人。研究发现，基于文本的机器学习模型在检测项目内部或项目之间的漏洞和漏洞类型[50]方面并不有效。与以往关注于提高基于ml的VD方法的有效性的研究不同，我们在广泛的范围内调查了各种数据抽样方法对DLVD方法的影响。我们还调查了我们的发现背后的原因，并为未来的从业者和研究人员提供了可操作的建议。

B.软件工程任务中的数据采样

数据采样已被用于处理各种软件工程任务中的数据不平衡问题，如缺陷预测[6]，[7]，[38]，[40]，[51]－[55]，bug分类[54]，[56]，软件质量预测[8]和软件变化预测[9]。大多数研究表明，数据抽样有助于改善给定的任务[6]，[7]，[53]-[55]，[57]，这与我们的发现相似。例如，Yedida和Menzies研究了使用深度学习[55]对深度学习的缺陷预测的价值，并表明过采样（模糊采样）可以显著改善大多数缺陷数据集的先验SOTA DL方法。在我们的研究中也有类似的发现，过采样可以改善SOTA DLVD。郑等人。对安全漏洞报告分类[54]的类再平衡方法进行了比较研究。他们评估了不同的抽样方法(e。g., SMOTE，ADASYN，和Rose)在多个分类器(e。g., 逻辑回归和随机森林)，发现Rose +随机森林的组合效果最好。Kaby等人。比较了故障易发模块检测任务中的欠采样和过采样，发现欠采样和过采样

具有类似的有效性[57]，而我们的研究结果表明，在DLVD中，过采样优于过采样。更重要的是，与之前的研究不同，我们专注于一个新的任务——DLVD。我们还比较了两种抽样策略-潜在空间抽样和。原始数据采样，并研究数据采样是否可以提高DLVD捕获真实脆弱模式的能力。

罗马数字 7结论

我们进行了第一个系统和广泛的研究，以评估数据抽样对SOTA DLVD方法中数据不平衡问题的影响。一般来说，过采样优于欠采样，对原始数据的采样优于潜在空间的采样，通常对原始数据的随机过采样在所有研究数据中（包括高级数据和OSS）表现最好。令人惊讶的是，OSS根本并不能帮助缓解DLVD中的数据不平衡问题。而如果进行回忆，随机欠抽样是最好的选择。对原始数据的随机过采样也提高了DLVD方法学习真实脆弱模式的能力。然而，对于相当一部分情况（在我们的数据集中至少有33%），DLVD方法不能基于真实的脆弱的陈述来进行预测。我们为从业者和研究人员提供了可操作的建议和路线图，例如，建议使用随机过采样来处理数据在DLVD中存在不平衡问题，而不推荐使用OSS。参考文献

[1] R.Scandariato, J.瓦尔登，A。霍夫西比安和W。*Joosen，“通过文本挖掘预测脆弱的软件组件”，《软件工程交易》，第1卷。*40岁，没有。10, pp.993– 1006, 2014.

[2] Z.李，D。Zou, S.Xu，X。Ou, H.金，S。王，Z。邓和Y。钟，“漏洞专家：基于深度学习的漏洞检测系统”，第25届年度网络和分布式系统安全研讨会，NDSS 2018，圣地亚哥，美国，2018年2月18-21日。

[3] S.查克拉伯蒂，R。克里希纳，Y。丁和B。Ray，“基于深度学习的漏洞检测：我们还存在了吗？”，IEEE软件工程事务报，2021年。

[4] D.欣，A。Kan, H.陈和M。A.“Lanevd：链接：使用图神经网络的声明级漏洞检测”，IEEE/ACM第19届采矿软件存储库国际会议，MSR 2022，匹兹堡，美国，美国，2022年5月23-24日，页。596 – 607.

[5] Y.李，S。王和T。N.Nguyen，“带有细粒度解释的脆弱性检测”，第29届ACM欧洲软件工程基础会议和研讨会论文集，2021页，页。292 – 303.

[6] L.佩拉约和S。迪克，“评估分层替代方案，以改进软件缺陷预测”，IEEE交易的可靠性，卷。61岁，没有。2, pp.516 – 525, 2012.

[7] H.徐，R。段，S。杨和L。*郭教授，“关于即时缺陷预测的数据抽样的实证研究”，发表在人工智能与安全国际会议上。*施普林格，2021年，页。54 – 69.

[8] C.塞弗特，T。M.霍什戈夫塔尔和J。*Van Hulse，“通过数据采样和增强改进软件质量预测”，IEEE系统、人与控制论学报-A部分：系统与人类，卷。*39日，没有。6, pp.1283– 1294, 2009.

[9] R.马尔霍特拉和M。*张娜，“利用不平衡数据进行软件变化预测的实证研究”，实证软件工程，第1卷。*22日，没有。6, pp.2806 – 2851, 2017.

[10] K.W.鲍耶，N。V.ChawlaL。O.霍尔和W。P.“打击：合成少数过采样技术，”科尔，卷。abs/1106.1813, 2011.在线可用： http://arxiv。org/abs/1106.1813

[11] M.库巴特，S。Matwin等人。，“解决不平衡训练集的诅咒：单边选择”，在Icml，卷。97年，没有。1.Citeseer，1997，p。179.

[12] Y.周，S。刘，J。Siow, X.杜和Y。刘：“通过图神经网络学习综合程序语义的有效漏洞识别”，《神经信息处理系统的进展》，第1卷。32, 2019.

[13] M.傅和C。“直线：基于变压器的脆弱性预测”，2022年。

[14] M.T.里贝罗，S。辛格和C。*“我为什么要信任你呢？”《第22届ACM SIGKDD知识发现与数据挖掘国际会议论文集，2016年，第3页。*1135 – 1144.

[15] Z.英，D。资产阶级，J。你，M。Zitnik和J。“产生图形神经网络的解释”，神经信息处理系统的进展，卷。32, 2019.

[16] T.米科洛夫，K。陈，G。科拉多和J。迪安，“向量空间中词表示的有效估计”，ICLR研讨会论文集，第1卷。2013, 01 2013.

[17] J.彭宁顿，R。Socher和C。D.曼宁：“手套：单词表示的全球向量”，2014年自然语言处理经验方法（EMNLP）会议论文集，2014年，页。1532 – 1543.

[18] Z.冯，D。郭，D。唐。段，X。冯，M。公，L。寿，B。秦，T。刘，D。江等人。，“科德伯特：编程和自然语言的预训练模型”，arXiv预印本arXiv：2002.08155,2020年。

[19] Z.李，D。Zou, S.徐，H。金，Y。朱和Z。陈，“Sysevr：使用深度学习检测软件漏洞的框架”，《IEEE关于可靠和安全计算交易》，2021年。

[20] Z.吴，S。Pan，F。陈，G。长，C。张和S。Y.菲利普，“关于图神经网络的综合调查”，IEEE关于神经网络和学习系统的交易，卷。32岁，没有。1, pp.4 – 24, 2020.

[21] F.山口，N。Golde，D。Arp和K。Rieck，“用代码属性图建模和发现漏洞”，2014年IEEE安全与隐私研讨会。IEEE，2014，页。590 – 604.

[22] R.拉塞尔。Kim，L。汉密尔顿，T。Lazovich, J.Harer，哦。奥兹迪米尔，P。艾灵伍德和M。McConley，“使用深度表示学习的源代码中的自动漏洞检测”，2018年第17届IEEE机器学习和应用国际会议（ICMLA）。IEEE，2018，页。757 – 762.

[23] N.V.查拉，K。W.鲍耶，L。O.霍尔和W。P.“烟雾：合成少数过采样技术”，《人工智能研究杂志》，第1卷。16, pp.321 – 357, 2002.

[24] I.*Tomek，“CNN的两个修改”，IEEE交易的系统，人，和控制论，卷。*7(2), pp.679 – 772, 1976.

[25] C.X.凌和C。李彦，《直接营销的数据挖掘：问题和解决方案》。*“在Kdd，卷。*98, 1998, pp.73 – 79.

[26] J.风扇，Y。李，S。王和T。N.Nguyen，“带有代码更改和代码摘要的交流/交流++代码漏洞数据集”，第17届挖掘软件存储库国际会议论文集，2020年，页。508 – 512.

[27] D.M.评估：“从精度、召回率和f-度量到roc、信息性、标记性和相关性”，arXiv预印本arXiv：2010.16061,2020。

[28] G.K.Rajbahadur, S.王，G。安萨尔迪，Y。龟甲和A。E.哈桑，《特征重要性方法对缺陷分类器解释的影响》，IEEE软件工程学报，2021年。

[29] G.K.Rajbahadur, S.王，Y。龟甲和A。E.“因变量，离散噪声对机器学习分类器的影响”，《软件工程学报》，第1卷。47岁，没有。7, pp.1414– 1430, 2019.

[30] TD..-B.勒，D。Lo和M。*李彦，“定位故障的约束特征选择”，2015年IEEE软件维护和进化国际会议（ICSME）。*IEEE，2015，页。501 – 505.

[31] A.Okutan和O。T.*Yıldız，“利用贝叶斯网络的软件缺陷预测”，《经验软件工程》，第1卷。*19日，没有。1, pp.154– 181, 2014.

[32] N.陈，S。C.Hoi和X。*Xiao，“软件过程评估：机器学习方法”，2011年第26届IEEE/ACM自动化软件工程国际会议（ASE 2011）。*IEEE，2011，页。333 – 342.

[33] W.傅，五。Nair和T。孟席斯，“为什么差异进化比网格搜索调优缺陷预测器更好？”arXiv预印本，arXiv：1609.02613,2016年。

[34] J.周，M。帕切科，Z。万，X。夏，D。Lo, Y.王和A。E.哈桑，“大海捞针：自动挖掘沉默”

漏洞修复”，在2021年第36届IEEE/ACM自动化软件工程国际会议（ASE）。IEEE，2021，页。705－716.
[35] D.Zou，Y.朱，S。徐，Z。李，H。金和H。叶，“解释基于启发式搜索的基于深度学习的漏洞检测器预测”，ACM软件工程和方法交易*ogy（TOSEM），卷*。30岁，没有。2，pp.1－31，2021.
[36] https://docs.alliancecan.ca/wiki/Narval/en/.
[37] M.王，D。郑，Z。是的，问。甘，M。李，X。宋，J。周，C。妈妈，L。Yu，Y。Gai，T.小，T。他，G。Karypis，J.李和Z。*Zhang，“深度图库：一个以图为中心、高性能的图神经网络包”，arXiv预印本arXiv：1909.01315,2019*。
[38] K.E.本宁，J。康，A。蒙登，P。Phannachitta和S。Mensah，“数据采样方法对软件缺陷优先级和分类的显著影响”，2017年ACM/IEEE经验软件工程和测量国际研讨会（ESEM）。IEEE，2017，页。364－373.
[39] L.龚，S。Jiang和L。蒋建，“通过基于聚类的过滤过采样来解决软件缺陷预测中的类不平衡问题”，第1卷。7，pp.145 725－ 145 737，2019.
[40] K.E.本宁，J。KeungP。Phannachitta。蒙登和S。“马哈基尔：基于多样性的过采样方法来缓解软件缺陷预测中的类别不平衡问题”，《软件工程学报》，第1卷。44岁，没有。6，pp.534－550，2017.
[41] C.Tantithamthorn。E.哈桑和K。松本本，“阶级再平衡技术对缺陷预测模型的性能和解释的影响”，《软件工程学报》，第1卷。46岁，没有。11，pp.1200－ 1219，2018.
[42] P.“凝聚的最近邻规则（corresp.）”，“IEEE跨-*在信息论上的行动，第1卷*。14日，没有。3，pp.515－516，1968.
[43]https://cve.米特尔。org/cgi-bin/cvename.cgi?name=CVE-2015-6766.
[44] L.范德马顿和G。“使用t-sne可视化数据。”机器学习研究杂志，第1卷。9，没有。11，2008.
[45] R.只是，D。Jalali和M。D.*恩斯特，“缺陷4j：现有故障以控制java程序的数据库”，2014年软件测试和分析国际研讨会论文集，2014，页*。437－440.
[46] M.马丁内斯。杜里奥克斯。萨默拉德，J。玄和M。*“java中真实错误的自动修复：对缺陷4j数据集的大规模实验”，实证软件工程，第1卷*。22日，没有。4，pp.1936－ 1964，2017.
[47] M.Gharehyazie，B.雷和V。*Filkov，“一些来自这里，有些来自那里：github中的跨项目代码重用”，2017年IEEE/ACM第14届挖掘软件库国际会议（MSR）*。IEEE，2017，页。291－301.
[48] C.Pornprasit，C.Tantithamthavorn，J.Jiarpakdee，M.Fu和P。“物理解释者：解释即时缺陷模型的预测”，2021年第36届IEEE/ACM自动化软件工程国际会议（ASE）。IEEE，2021，页。407－418.
[49] T.T.阮，H。A.阮，N。H.Pham，J。M.Al-Kofahi和T。N.*Nguyen，“基于图形的多对象使用模式的挖掘”，第七届欧洲软件工程会议第七届联合会议和ACM软件工程基础研讨会，ser*。ESEC/FSE '09.纽约，纽约，美国：计算机机械协会，2009，p。383－392.在线可用性：https://doi.org/10.1145/1595696.1595767
[50] N.科林，B。Tanmay和W。绍伟，“基于文本的漏洞检测机器学习模型的实证研究”，实证软件工程出版社，2022年。
[51] M.谭，L。谭，S。Dara和C。*Mayeux，“不平衡数据的在线缺陷预测”，2015年IEEE/ACM第37届IEEE软件工程国际会议，第1卷*。2.IEEE，2015，页。99－ 108.
[52] K.E.本宁，J。W.Keung和。*“关于软件缺陷预测的数据重采样方法的相对价值”，《经验软件工程》，第1卷*。24日，没有。2，pp.602－636，2019.
[53] S.冯，J。KeungX。Yu，Y。肖，K。E.本宁，M。A.Kabir和M。张教授，“基于复杂度的过采样技术来缓解软件缺陷预测中的类不平衡问题”，《信息与软件技术》，第1卷。129，p.106432，2021.
[54] W.郑，Y。荀、X。吴，Z。邓，X。陈和Y。*“安全错误报告分类的类再平衡方法的比较研究”，IEEE可靠性交易，卷*。70岁，没有。4，pp.1658－ 1670，2021.
[55] R.Yedida和T。“关于深度过采样的价值”*《软件缺陷预测中的学习》，《IEEE软件工程学报》，2021年*。
[56] R.舒，T。夏，J。陈，L。威廉姆斯和T。孟席斯，“如何更好地区分安全错误报告（使用双超参数优化），”经验软件工程，第1卷。26日，没有。3，pp.1－37，2021.
[57] Y.家美，A。蒙登，S。松本，T。Kakimoto和Ki。.-*松本，“过采样和过采样对易故障模块检测的影响”，在第一次经验软件工程和测量国际研讨会（ESEM 2007）*。IEEE，2007，页。196－204.