软件学报 ISSN 1000-9825, CODEN RUXUEW
2024,35(12):5452−5469 [doi: 10.13328/j.cnki.jos.007088] [CSTR: 32375.14.jos.007088]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

IADT: 基于解释分析的深度神经网络差分测试*

谢瑞麟[1], 崔展齐[1], 陈 翔[2], 李 莉[1]

[1](北京信息科技大学 计算机学院, 北京 100101)

[2](南通大学 信息科学技术学院, 江苏 南通 226019)

通信作者: 崔展齐, E-mail: czq@bistu.edu.cn

摘 要: 随着深度神经网络 (deep neural network, DNN) 的迅猛发展, 其在某些特定任务上的准确性已可媲美甚至超过人类. 然而, DNN 与传统软件一样不可避免地存在缺陷, 若将带缺陷的 DNN 模型应用于安全攸关的领域甚至可能引发严重事故, 如何有效检测缺陷 DNN 模型已成为亟需解决的问题. 传统的差分测试方法将测试目标在同一测试输入下的输出结果作为差异分析的依据. 然而, 即使是相同训练程序和数据重复训练的不同 DNN 模型在同一测试输入下也常会产生不同的输出结果. 因此, 不能认为模型输出具有差异的两个模型中存在缺陷模型, 基于测试对象输出结果的传统差分测试方法难以直接用于 DNN 模型的缺陷检测. 为解决上述问题, 提出基于解释分析的 DNN 模型差分测试方法 IADT (interpretation-analysis-based differential testing), 利用解释方法分析 DNN 模型对于测试输入的行为解释, 并使用统计方法分析模型对测试集行为解释的显著性差异来检测缺陷模型. 使用真实缺陷模型进行实验的结果表明, 解释方法的引入使 IADT 能有效检测缺陷 DNN 模型, 检测缺陷模型的 $F1$ 值比 DeepCrime 高 0.8%–6.4%, 而所消耗的时间仅为 DeepCrime 的 4.0%–5.4%.

关键词: 深度神经网络; 差分测试; 可解释性; 软件测试

中图法分类号: TP311

中文引用格式: 谢瑞麟, 崔展齐, 陈翔, 李莉. IADT: 基于解释分析的深度神经网络差分测试. 软件学报, 2024, 35(12): 5452–5469. http://www.jos.org.cn/1000-9825/7088.htm

英文引用格式: Xie RL, Cui ZQ, Chen X, Li L. IADT: Interpretability-analysis-based Differential Testing for Deep Neural Network. Ruan Jian Xue Bao/Journal of Software, 2024, 35(12): 5452–5469 (in Chinese). http://www.jos.org.cn/1000-9825/7088.htm

IADT: Interpretability-analysis-based Differential Testing for Deep Neural Network

XIE Rui-Lin[1], CUI Zhan-Qi[1], CHEN Xiang[2], LI Li[1]

[1](Computer School, Beijing Information Science and Technology University, Beijing 100101, China)

[2](School of Information Science and Technology, Nantong University, Nantong 226019, China)

Abstract: With the rapid development of deep neural network (DNN), the accuracy of DNN has become comparable to or even surpassed that of humans in some specific tasks. However, like traditional software, DNN is inevitably prone to defects. If defective DNN models are applied to safety-critical fields, they may cause serious accidents. Therefore, it is urgent to propose effective methods to detect defective DNN models. The traditional differential testing methods rely on the output of the testing target at the same test input as the basis for difference analysis. However, even different DNN models trained with the same program and dataset may produce different outputs under the same test input. Therefore, it is difficult to directly use the traditional differential testing method for detecting defective DNN models. To solve the above problems, this study proposes interpretation-analysis-based differential testing (IADT), an interpretation-analysis-based differential testing method for DNN models. IADT uses interpretation methods to analyze the behavior explanation of DNN models and uses statistical methods to analyze the significant differences in the models' behavior interpretations to detect defective models.

* 基金项目: 江苏省前沿引领技术基础研究专项 (BK20202001); 国家自然科学基金 (61702041); 北京信息科技大学"勤信人才"培育计划 (QXTCP C201906)
收稿时间: 2023-03-17; 修改时间: 2023-06-09, 2023-08-01; 采用时间: 2023-11-02; jos 在线出版时间: 2024-08-14
CNKI 网络首发时间: 2024-08-15

Experiments carried out on real defective models show that the introduction of interpretation methods makes IADT effective in detecting defective DNN models, while the $F1$-value of IADT in detecting defective models is 0.8% –6.4% greater than that of DeepCrime, and the time consumed by IADT is only 4.0%–5.4% of DeepCrime.

Key words: deep neural network (DNN); differential testing; interpretability; software testing

深度神经网络 (deep neural network, DNN) 通过引入多层感知机和非线性激活函数建立了类似人脑的神经元结构, 从而学习底层输入特征到高层语义输出的高维映射关系[1]. 目前深度学习已经取得了显著进展, 并在计算机视觉[2,3]、自然语言处理[4,5]和软件工程[6–9]等领域得到实际应用. 随着先进的模型结构和大型数据集的不断推出, DNN 在某些特定任务上的准确性已可媲美甚至超过人类. 然而, DNN 与传统软件一样不可避免地存在缺陷. 在一些安全攸关的领域, DNN 模型中未被测试发现的隐藏缺陷将可能引发灾难性的后果. 如特斯拉和优步的自动驾驶汽车因 DNN 缺陷而导致了多起严重事故[10,11].

由于 DNN 模型的安全性和可靠性持续受到关注, 大量工作致力于研究 DNN 模型缺陷检测方法[12]. 现有 DNN 模型缺陷检测方法大多通过比较 DNN 模型在测试数据下的输出和预期输出来检测缺陷. 然而 $F1$ 值和精确率等指标只能反映模型的性能差异, 难以反映模型内部的行为差异. 由于不同 DNN 模型可能根据不同的行为产生相同的预测结果 (例如根据图像中不同区域的特征但输出相同的分类结果), 因此模型行为差异引起的缺陷难以通过比较模型输出结果或测试精度来检测.

目前已有工作发现模型行为差异可能在实际应用中引起安全性和公平性问题[13]. 通过比较多个测试目标之间的行为差异, 差分测试 (differential testing)[14]可用于检测与模型行为差异相关的缺陷. 因其只需关注测试目标的行为差异而无需关注待测目标的内部实现而被广泛应用于深度学习平台缺陷检测[15,16]、对抗样本生成[17]和解决测试预言问题[18,19]. 传统的差分测试方法将测试目标在同一测试输入下的输出结果作为模型差异分析的依据, 目前已有工作将差分测试用于机器学习模型的缺陷检测工作[20]. 然而 DNN 模型的训练过程具有随机性因素, 即使是相同训练程序训练的不同 DNN 模型在同一测试输入下也可能具有不同输出结果. 因此, 不能认为模型输出具有差异的两个模型之间存在缺陷模型, 基于测试对象输出结果的传统差分测试方法难以直接用于 DNN 模型的缺陷检测.

差分测试方法需要量化被测对象之间的行为差异来检测缺陷, 但随机性因素带来的不确定性使 DNN 模型的行为差异量化工作具有一定挑战. 为缓解 DNN 模型训练过程中随机性因素对模型差异分析结果的影响, DeepCrime[21,22]提出了进行多次训练并使用统计学方法分析两组 DNN 模型之间测试精度的显著性差异的方法, 来比较被测试目标模型之间的差异. DeepCrime 提出的 DNN 模型差异分析方法在一定程度上消除了随机性因素的影响, 但其仍然依靠基于模型输出结果的测试精度指标作为差异分析的依据. 当测试集不能触发模型之间足够多的输出差异时, DeepCrime 将可能无法检测到模型之间的差异, 进而使差分测试无法准确检测缺陷模型. 同时, DeepCrime 需要使用训练程序多次训练待测模型, 这使其只能检测训练程序中由编码开发阶段引入的缺陷, 而无法检测单个模型中由训练阶段引入的缺陷. 此外, 多次训练也将消耗较多的时间成本.

为解决上述问题, 本文提出了基于解释分析的 DNN 模型差分测试方法 IADT (interpretation-analysis-based differential testing), 利用解释方法分析 DNN 模型对于测试输入的行为解释, 并使用统计方法分析模型对测试集行为解释的显著性差异来检测缺陷模型. 解释是指对 DNN 模型的预测行为进行分析, 并使人能理解其行为的过程. 解释方法可更细粒度地量化 DNN 模型之间的差异, 这使 IADT 摆脱了 DNN 差分测试方法对模型输出结果的依赖, 转而使用模型内部的行为解释作为判断 DNN 模型之间差异性的依据, 使其无需重复训练就能在两个模型之间比较行为差异. 其中, 模型行为是指 DNN 模型在测试输入下的预测活动, 通常表现为输出结果或测试精度等, 而本文方法则使用解释方法输出的模型行为解释来更具体的表示模型行为. 相比依赖测试精度且需要多次训练来判断模型差异的 DeepCrime, IADT 有效提高了缺陷模型的检测能力和时间效率.

本文的主要贡献如下.

- 提出了基于解释分析的 DNN 模型差分测试方法 IADT, 根据行为差异量化结果检测 DNN 模型行为差异缺陷. 其中, 使用解释方法和统计分析量化 DNN 模型之间的行为差异, 并缓解了随机性因素对量化结果的影响.

● 基于所提出方法实现了原型工具, 在 276 个 DNN 模型上进行对比实验的结果表明, 与基于测试精度的 DNN 模型差异分析方法 DeepCrime 相比, IADT 具有更高的缺陷模型检测能力和时间效率.

本文第 1 节介绍相关工作并分析 IADT 的创新性. 第 2 节介绍本文方法的动机和相关示例. 第 3 节详细介绍所提出的基于解释分析的差分测试方法 IADT. 第 4 节介绍实验设计, 并对实验结果进行分析和讨论. 第 5 节对实验结果和 IADT 的有效性进行讨论. 最后第 6 节总结全文并对未来工作进行展望.

1 相关工作

DNN 的普遍应用及几起严重事故[10,11]引起了人们对其安全性和可靠性的广泛关注. DNN 模型复杂的内部结构使其能学习到输入数据中的细微特征, 但只需向输入数据添加难以被人类察觉的微小扰动即可能使 DNN 模型输出错误结果. 而训练和预测过程难以理解使传统软件测试方法难以有效检测 DNN 模型的缺陷. 因此, 亟需有效的 DNN 模型验证和测试方法以保障其质量.

验证是 DNN 模型质量保障的重要手段, 现有的验证方法主要使用约束求解技术对鲁棒性、安全性等问题进行求解. Huang 等人[23]提出了基于 SMT 求解器的 DNN 模型自动安全性验证方法 DLV, 将模型输入视为空间中的点, 采用图像变异方法对图像附近的输入空间进行探搜索. 此外, DLV 通过 SMT 求解器对 DNN 模型进行逐层传播分析来验证 DNN 模型的安全性. 实验结果表明, 相比其他对抗样本生成方法, DLV 可发现具有更细微扰动的对抗样本. Lomuscio 等人[24]使用混合整数线性规划 (MILP) 和线性规划来表示激活函数的约束和输出范围, 通过求解 MILP 的可达性问题来验证输出集合, 以对 DNN 模型的鲁棒性进行评估. Bunel 等人[25]通过 MILP 公式, 引入了用于神经网络验证的分支定界框架 BaB 和 BaBSB, 即一种有效的 ReLU 非线性分支策略, 能够高效且成功地处理具有卷积网络架构的高输入维问题. 实验结果表明, BaB 和 BaBSB 在发现错误的效率上都要优于其他方法. 验证技术能对 DNN 模型的鲁棒性和安全性等属性进行评价, 但在应用于结构较为复杂的 DNN 模型时效果受限, 且在不同结构模型间的通用性也有限[26]. 测试技术则通过发现潜在的威胁或缺陷来保障 DNN 模型质量, 现有的 DNN 模型测试工作主要关注测试度量指标和测试输入生成两个方面.

测试度量指标主要用于评估测试集的充分性, 更充分的测试集将更有可能检测到 DNN 模型的缺陷. 受传统测试覆盖思想的启发, Pei 等人[19]首次将神经元覆盖率作为衡量 DNN 模型测试充分性的指标, 并提出了白盒测试方法 DeepXplore. 在神经元覆盖率的基础上, Ma 等人[27]将 DNN 模型的测试覆盖评估分为神经元级和层级, 并提出了包括 *k*-multisection Neuron Coverage 和 Top-*k* Neuron Coverage 在内的 5 种测试覆盖率标准. 然而, 以上覆盖率标准缺少对于单个测试输入的分析且粒度较粗. 与测试覆盖率指标不同, 在变异测试方法[28]中, 通过执行变异操作来生成一系列含有缺陷的变异体程序. 这些根据不同变异规则对待测程序进行变异的操作被称为变异算子. 变异测试通过评估测试集对缺陷变异体的识别能力来评价测试充分性. Ma 等人[29]将变异测试思想引入 DNN 系统测试中, 提出了 DeepMutation 及其后续改进工作 DeepMutation++[30], 分别针对 DNN 的训练代码、训练数据和模型参数提出了 8 种变异算子. 此外, DNN 模型的测试度量指标也为深度随机森林[31]等其他深度学习模型的测试提供了参考.

测试输入生成主要用于生成足够的测试输入以提高测试充分性, 并对 DNN 模型的一般行为和各种角盒行为[27]进行充分测试. 主流的测试输入生成通常分为基于覆盖[32]和基于对抗[33,34]的方法. 在基于覆盖的测试输入生成方法中, Lee 等人[35]提出的 ADAPT 方法, 其根据给定的测试覆盖率标准、数据集和 DNN 模型自适应地学习和生成神经元选择策略, 并计算所选择神经元相对于输入的梯度. 然后向梯度上升的方向修改原始测试输入来生成新测试输入以增加所选神经元的输出值并提高测试集的覆盖率. 在基于对抗的测试输入生成方法中, Xiao 等人[36]提出了基于空间变换的图像测试输入生成方法, 在空间上移动图像中的现有像素, 以通过不添加额外扰动的方式生成新测试输入, 而非直接对图片中的像素值进行变异. 实验结果表明, 该方法生成的测试输入与 FGSM[37]和 C&W[38]相比更接近原始测试输入.

以上方法均通过比较 DNN 模型对测试输入的输出和预期输出来检测缺陷. 而这不可避免地存在测试预言问

题[39], 即如何确定待测对象的测试输出是否符合测试用例的预期输出. 为解决传统软件测试中的测试预言问题, McKeeman 等人[14]认为具有相同功能的待测系统在给定相同测试输入时应该具有相同的输出, 并基于这一思想提出了差分测试的概念. 而对于深度学习软件的测试工作, 差分测试也同样被广泛应用. Guo 等人[17]提出了第 1 个差分模糊测试框架 DLFuzz. 其将神经元覆盖差异和预测结果差异作为评价变异测试输入的适应度函数, 通过迭代添加细微扰动来生成具有高适应度的测试输入, 且所生成的测试输入无需手动标记而是交叉引用来自具有相同功能的其他 DNN 模型来确定测试预言. Pham 等人[15]提出了面向深度学习库的差分测试方法 CRADLE. 他们认为, 许多功能在不同的深度学习库中都有等价的实现, 对于相同测试输入, 不同库中相同功能的实现应产生相同输出. CRADLE 通过使用 Keras 比较同一 DNN 模型在这些不同深度学习库下的表现, 能够有效地进行缺陷检测. 然而 Keras 可配置加载不同后端的特性因过高的成本而不再继续更新维护, 在不同深度学习库之间直接进行差分测试的成本过于高昂. 为解决这一问题, Wang 等人[16]提出了针对单一深度学习库的差分测试方法 EAGLE, 其通过构建等效图来实现对单一深度学习库的差分测试. 等效图使用不同的应用程序编程接口、数据类型或优化器来实现相同的功能. 在给定相同测试输入的情况下两个等效图应产生相同的输出. 他们设计了 16 个深度学习库等价规则, CRADLE 使用这些等价规则构建具体的等价图对, 并通过交叉检查这些等价图对的输出来检测深度学习库中的不一致错误. 他们使用 CRADLE 在 TensorFlow 和 PyTorch 两个深度学习库中检测到了 25 个不一致错误, 其中 13 个为此前未知的错误.

以上方法将差分测试用于深度学习库的测试工作或用其解决测试预言问题. 由于 DNN 模型的训练过程具有随机性因素, 将差分测试用于检测与行为差异相关的缺陷 DNN 模型存在一定挑战. 为解决这一问题, 本文提出了基于解释分析的 DNN 模型差分测试方法 IADT, 使用解释方法分析具有相同功能的不同 DNN 模型在相同测试输入下的行为解释, 并使用统计方法消除随机性因素的影响并分析行为解释的显著性差异来检测缺陷模型.

2 动机示例

传统差分测试方法通过比较不同版本测试对象的输出差异来检测缺陷. 然而, 即使是相同训练程序训练的不同 DNN 模型在同一测试输入下也可能具有不同输出结果. 因此, 不能认为模型输出或测试精度具有差异的两个模型中一定存在缺陷模型. 测试精度只能大致反映 DNN 模型的性能水平, 而无法准确表示模型在测试输入下的具体行为, 若使用解释方法对 DNN 模型的预测行为进行解释分析, 或将能更细粒度的反映模型行为并准确分析 DNN 模型之间的行为差异.

如图 1 为分别使用测试精度和解释方法对 DNN 模型进行差分测试的示例. 其中, 红色待测模型为具有缺陷的 DNN 模型, 其余两个模型为正常 DNN 模型. 虚线上方为使用测试精度表示模型行为的差分测试方法. 将测试集分别输入 3 个待测模型并使用模型输出计算其测试精度, 由于 DNN 模型训练阶段的随机性因素, 使两个正常模型之间也存在一定测试精度差异, 在两两模型之间的行为差异分析中均呈现“有差异”的判断, 因此无法准确检测缺陷模型.

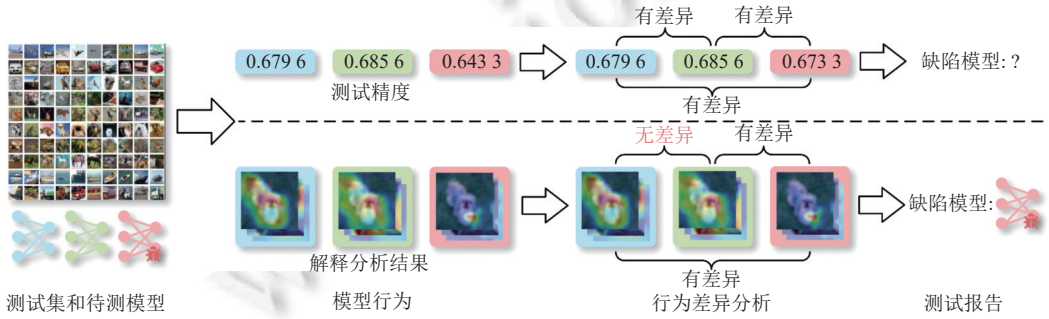

图 1 使用不同行为差异分析方法进行差分测试的示例

虚线下方为使用解释方法对 DNN 模型的预测行为进行解释分析, 并对模型行为解释进行差异分析的方法. 将测试集分别输入待测模型后使用解释方法对模型预测行为进行分析并输出解释分析结果. 其中, 红色缺陷模型的行为解释与其余两个正常模型的行为解释出现较大差异, 而两个正常模型的行为解释之间虽具有一定差异, 但总体较为相似而可被认为“无差异”. 因此, 与其他两个模型具有较大差异的红色 DNN 模型被准确检测为缺陷模型. 基于以上思想, 本文提出了基于解释分析的 DNN 模型差分测试方法 IADT.

3 基于解释分析的 DNN 模型差分测试方法

与传统差分测试方法比较不同版本之间测试对象的输出差异不同, IADT 使用解释方法分析并比较 DNN 模型在相同测试输入下的行为差异, 并以此作为判断 DNN 模型之间差异性的依据. IADT 的工作流程分为解释分析、模型行为解释降维和行为差异分析 3 个阶段, 具体过程如图 2 所示. 首先, 在解释分析阶段将测试集和多个 DNN 模型分别使用解释方法分析模型行为解释 (model behavior interpretation, *MBI*); 然后, 将 *MBI* 降至一维序列; 最后, 在行为差异分析阶段分析不同 DNN 之间的行为解释的差异, 当某个模型的行为解释与其他模型之间存在显著差异时则认为其具有缺陷 (例如图 1 中的红色 DNN 模型).

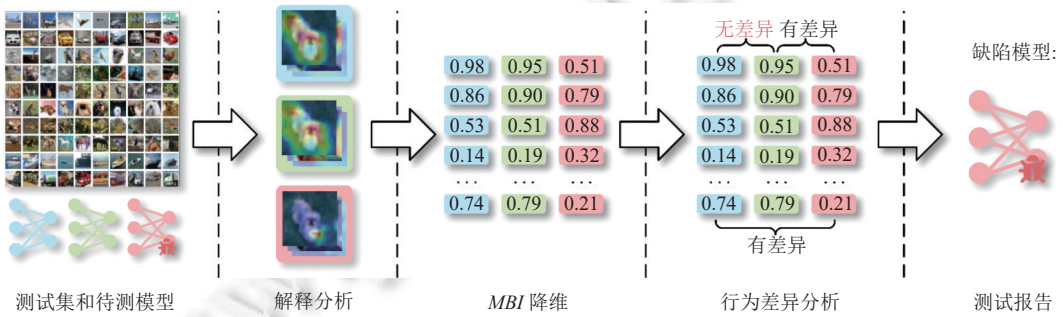

图 2 基于解释分析的深度神经网络差分测试方法流程图

IADT 的具体步骤如算法 1 所示, 算法的输入为具有相同功能的多个 DNN 模型 *models* 和带标签的测试集 *testInput*. 算法的输出为检测到缺陷的 DNN 模型. 首先使用解释方法依次分析每个 DNN 模型对测试集 *testInput* 中每条测试输入的行为解释并添加进模型行为解释集 S_{MBI} 并降至一维序列 (第 1–5 行), 解释分析和降维方法将在第 3.1 节介绍; 然后分别比较 S_{MBI} 中每两个模型之间行为解释的差异 (第 8–11 行), 当两个模型之间的行为解释存在显著性差异时, 分别将两个模型在差异统计数组 *diffCount* 中的对应值加 1 (第 12–14 行), 行为解释的差异分析方法将在第 3.2 节介绍; 最后将所有模型按差异统计数组 *diffCount* 中的差异数量进行降序排序, 排序后的模型列表作为缺陷模型怀疑度输出 (第 20、21 行).

算法 1. 基于解释分析的深度神经网络差分测试方法.

输入: *models*: multiple DNN models with same function; *testInput*: test set with ground truth label;
输出: DNN model with detected defects.

1. Initialize $S_{\text{MBI}} = \varnothing$
2. **for** each *m* in *models* **do**
3. $S_{\text{MBI}} = S_{\text{MBI}} \cup$ interpretationMethod(*m*, *testInput*) //根据模型结构选择解释方法
4. **end for**
5. S_{MBI} = oneDim(S_{MBI}) //降至一维
6. Initialize $i \leftarrow 0$
7. Initialize *diffCount* ← int[len(*models*)]

```
8. while i < len(models) do //比较每两个模型的行为差异
9.     Initialize j ← 0
10.    while j < i do
11.        isDiff = isDifferent(S_MBI[i], S_MBI[j]) //判断是否存在差异
12.        if isDiff is True do
13.            diffCount [i] += 1
14.            diffCount [j] += 1
15.        end if
16.        j += 1
17.    end while
18.    i += 1
19. end while
20. defectDNN = sort(diffCount, models) //所有模型按差异数量排序
21. return models
```

3.1 解释分析方法及 *MBI* 降维

差分测试的核心原理基于如下假设, 即“具有相同功能的软件应具有相同或类似的行为”[14]. 行为差异较大的软件则可能具有缺陷. 如何分析并量化不同软件之间行为差异是差分测试的关键. 传统差分测试方法根据测试输出结果判断待测目标之间的行为差异, 然而相同功能的 DNN 模型之间产生不同的预测结果是正常现象, 正常模型之间可能同样会被认为存在行为差异. 基于模型输出的行为量化指标将无法区分正常模型和缺陷模型. DNN 模型的解释方法通过分析模型的预测行为, 通过一种更细粒度的方式来揭示模型之间的行为差异. 相比测试精度和 *F*1 值等基于模型预测结果的指标, 解释方法能够更精确地帮助差分测试方法比较不同 DNN 模型的差异.

以 DNN 为代表的深度学习模型已在许多领域展现出了卓越的性能. 但相比依靠人工设计程序逻辑的传统软件, 深度学习模型的内部逻辑源自模型根据样本数据学习特征并做出预测, 其预测过程更为复杂且缺乏可解释性, 导致用户难以理解和预测其行为. 为此, 学术界和工业界将可解释性作为深度学习模型的重点研究问题[40,41], 并陆续提出了多种解释分析方法. 以卷积神经网络 (convolutional neural network, CNN) 为例, 常见的 CNN 解释方法有基于梯度加权类激活映射的 Grad-CAM[13]和 Grad-CAM++[42], 基于元素梯度加权的 Layer-CAM[43]等. 在指定 CNN 模型的目标层和目标分类后, 以上方法可计算输入图像中不同区域对 CNN 模型分类结果的重要性程度, 并生成 CNN 模型的 *MBI*. 其本质为与输入图像大小相同的二维矩阵, 其中元素值越大, 输入图像中相同位置的像素对模型分类结果的重要性就越高. 因而可将 CNN 的预测行为解释为: CNN 模型根据 *MBI* 中元素值较高的区域在输入图像中的特征而将输入图像分类为目标类型.

如图 3 所示为以上 3 种解释方法对不同 CNN 模型生成的 *MBI* 与输入图像叠加的示例. 其中, 3 个 CNN 模型为使用 CIFAR-10[44]数据集以不同超参数训练的 LeNet-5 模型, 第 1、2 列为正常 CNN 模型 (测试精度在 68% 以上) 的 *MBI*, 第 3 列为缺陷 CNN 模型 (测试精度为 23%) 的 *MBI*, 且示例图像均选自 CIFAR-10 数据集中被 3 个 CNN 模型正确分类的图像. 如图 3 所示, 3 种解释方法可对模型的预测行为做出解释, 并均能区分相同功能 CNN 模型的行为差异. 3 个模型对相同测试输入产生了相同的预测结果, 但其 *MBI* 之间存在明显差异, 其中缺陷 CNN 模型与其他两个模型差异最大, 且 Grad-CAM 生成的 *MBI* 可更明显地反映这一差异.

本文所关注的对 DNN 模型进行解释分析的方法通常会生成模型的类激活图[13] (如图 3), 此类形式的 *MBI* 为二维矩阵格式. 而 IADT 的行为差异分析阶段将两组一维数组作为输入并检验二者的显著性差异. 因此, 对 DNN 模型进行差异性分析需要将二维矩阵格式的 *MBI* 降维至单个浮点数, 并将模型对测试集中所有测试输入的 *MBI* 组合为一维数组. 为将 *MBI* 的二维数组降维至单个浮点数并尽可能保留不同 *MBI* 之间的差异, 可将 *MBI* 的二维

矩阵视为 n 维空间中的一点 (n 为矩阵中的元素数量), 并使用欧几里得范数 (L_2 范数)[45]计算 *MBI* 距离原点的距离以将二维数组降维至单个浮点数.

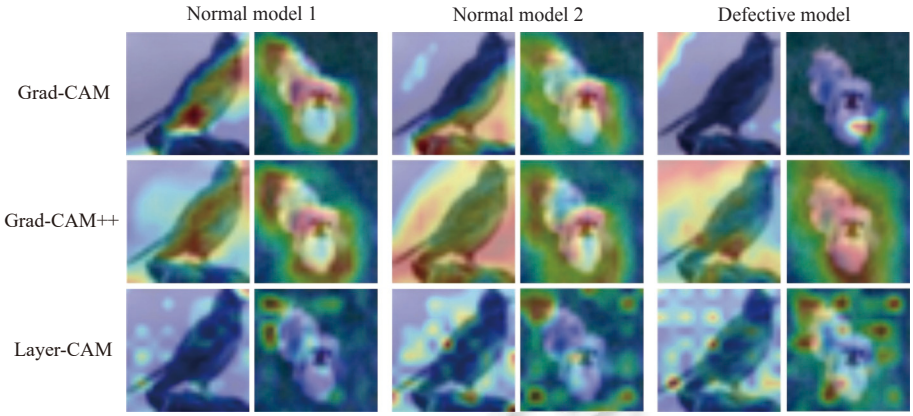

图 3 不同解释方法对不同 CNN 模型生成的 *MBI* 与输入图像叠加示例

令 $MBI=\{c_1,c_2,\ldots,c_n\}$ 为类激活图的二维矩阵, $c_i \in MBI\ (1 \leqslant i \leqslant n)$ 为矩阵中的其中一个元素值, n 为矩阵中的元素数量. L_2 范数的具体计算方法如公式 (1) 所示:

$$\|MBI\|_2 = \sqrt{\sum_i^n (c_i)^2} \tag{1}$$

3.2 行为解释的差异分析方法

解释方法能对 DNN 模型的预测行为进行解释, 存在差异的模型在相同的测试输入下将产生不同的 *MBI*. 然而 DNN 模型的训练过程存在随机性, 相同训练程序经过多次训练得到的 DNN 模型在相同测试输入下可能产生不同行为. 因此, 即使在单个测试输入下的行为解释或预测输出存在明显差异也依然无法确定 DNN 模型之间存在显著性差异.

为解决这一问题, DeepCrime[21,22]将两个待测训练程序分别训练多次后得到两组数量相同的 DNN 模型, 将测试集分别输入所有模型后得到两组 DNN 模型的测试精度, 并用广义线性模型 (generalised linear model, GLM)[46]判断两组精度之间的显著性差异. 当两组精度之间存在显著性差异时则认为两个待测程序所训练的 DNN 模型之间也存在显著性差异. 通过多次训练加统计学显著性检验的方法 DeepCrime 在一定程度上避免了 DNN 模型训练过程中的随机因素对模型差异性分析的影响. 然而 DeepCrime 使用基于模型预测输出的测试精度作为 GLM 的判断依据. 如第 2 节的示例所示, 当正常模型之间的精度差异较大时, DeepCrime 可能将正常模型错误地识别为缺陷模型, 进而导致较高的误报率. 例如 DeepCrime 在 MNIST 数据上的召回率可达 98.7%, 但由于误报率高达 40%, 导致精确率仅为 60% (见第 4.4 节).

为同时保证较高的精确率和召回率, 需要使用更细粒度的方法反映模型行为, 考虑到即使相同训练程序训练的不同 DNN 模型在同一测试输入下也可能具有不同输出结果, 使用测试精度或预测输出无法准确反映模型行为, 因此本文使用解释方法分析 *MBI* 作为 DNN 模型之间差异性分析的依据. 两个待测模型的在测试输入下的 *MBI* 属于两两配对关系且属于非正态分布. 因此, IADT 使用配对样本 *Wilcoxon* 符号秩和检验 (paired samples *Wilcoxon* signed-rank-sum test)[47]作为 *MBI* 的统计显著性检验方法. 配对样本 *Wilcoxon* 符号秩和检验是一种非参数的统计假设检验, 可使用两组配对样本比较其总体差异. 由于 DNN 模型训练过程的随机性因素, 不能认为 *MBI* 具有差异的两个模型中一定存在缺陷模型, 因此除了使用假设检验方法验证 *MBI* 的显著性差异以外, 还应对差异进行量化, 只有当量化后的 *MBI* 差异大于一定阈值时才可认为两个模型之间存在显著性差异. 使用配对样本 *Wilcoxon* 符号秩和检验进行行为解释的差异分析属于连续型数据的组间差异分析, 我们选择续型数据组间差异分析中常用的效应量计算方法 Cohen's d 对 *MBI* 进行量化比较. 该方法具有计算简单、易于理解、时间效率高等优点. 此外,

DeepCrime 同样选择使用 Cohen's d 作为效应量计算方法来缓解随机性因素影响.

模型行为解释的统计显著性检验方法的具体计算方法如公式 (2) 所示.

$$isDiff(m_a,m_b,T)=\begin{cases}\text{true}, & Wilcoxon(MBI(m_a,T),MBI(m_b,T))<\alpha\\ & \text{and } effectSize(MBI(m_a,T),MBI(m_b,T))\geqslant\beta\\ \text{false}, & \text{otherwise}\end{cases} \tag{2}$$

令 m_a 和 m_b 为待比较的 DNN 模型, T 为测试集. $MBI(m_{a/b},T)$ 表示将测试集 T 输入模型 m_a 或 m_b 并使用解释方法分析得到的行为解释. $Wilcoxon(MBI(m_a,T),MBI(m_b,T))$ 表示使用配对样本 *Wilcoxon* 符号秩和检验计算模型 m_a 和 m_b 在测试集 T 下的行为解释的差异置信度值. $effectSize(MBI(m_a,T),MBI(m_b,T))$ 表示使用 Cohen's d 计算模型 m_a 和 m_b 在测试集 T 下的行为解释之间的效应量. 当 P 值低于预设的阈值 α 时且效应量高于或等于预设的阈值 β 时认为模型 m_a 和 m_b 之间存在显著性差异.

4 实验设计与结果分析

4.1 实验设计

实验部分将关注 IADT 对 DNN 模型的行为差异分析能力. 为验证 IADT 对多个具有相同功能的 DNN 模型进行行为差异分析以检测缺陷模型的能力, 我们使用 AUTOTRAINER[48]提供的正常训练程序和缺陷训练程序 (AUTOTRAINER, https://github.com/shiningrain/AUTOTRAINER) 训练实验所用的正常模型和缺陷模型, AUTOTRAINER 为 CIFAR-10[44]数据集提供了 35 个正常训练程序和 45 个具有缺陷的训练程序, 为 MNIST[49]数据集提供了 78 个正常训练程序和 38 个具有缺陷的训练程序. 此外, 由于 SVHN[50]数据集具有和 CIFAR-10 数据集相同的数据结构, 我们还使用 CIFAR-10 数据集的训练程序和 SVHN 数据集训练实验模型. 实验将验证 IADT 分析 DNN 模型行为差异和检测缺陷模型的能力, 并选择同样消除了 DNN 训练随机性因素的模型差异分析方法 DeepCrime 作为比较对象. 所有实验均在配备 NVIDIA RTX 3070Ti GPU 和 AMD 7700X CPU 的计算机上进行.

为验证所提出基于解释分析的深度神经网络差分测试方法的有效性, 实验将研究并回答以下问题.

RQ1: IADT 相比其他模型差异性分析方法是否具有更高的缺陷模型检测能力和时间效率?

为验证 IADT 对 DNN 模型进行差异分析的有效性和所消耗的时间成本, 设计了 RQ1 以分析 IADT 的缺陷模型检测能力及时间效率.

RQ2: 解释方法的引入是否提高了 IADT 的缺陷模型检测能力?

解释方法的引入是 IADT 将差分测试用于检测缺陷 DNN 模型的重要步骤, 为验证其对 IADT 的缺陷模型检测能力的影响, 设计了 RQ2 以对比解释方法引入前后 IADT 的缺陷模型检测能力.

RQ3: 不同解释方法如何影响 IADT 检测变异体模型的能力?

使用不同的解释方法将影响 IADT 检测变异体模型的能力, 为此设计了 RQ3 用以讨论解释方法的差异和选择.

4.2 评价标准

实验将使用同一个数据集训练的每个正常模型与其他模型 (其余正常模型和所有缺陷模型) 两两配对进行模型差异分析. 对 CIFAR-10、MNIST 和 SVHN 数据集的正常模型和缺陷模型进行配对将分别产生 2 170 个 ($C_{35}^2+(35\times45)$)、5 967 个 ($C_{78}^2+(78\times38)$) 和 2 170 个 (与 CIFAR-10 相同) 样本, 并分别使用准确率、精确率、召回率和 $F1$ 值评价缺陷模型检测能力. 对于缺陷模型检测问题, 正常模型和缺陷模型之间被正确识别为"有差异"称为 *TP*, 正常模型和正常模型之间被错误识别为"有差异"称为 *FP*, 正常模型和正常模型之间被正确识别为"无差异"称为 *TN*, 正常模型和缺陷模型之间被错误识别为"无差异"称为 *FN*. 公式 (3) 为准确率 (*Accuracy*) 的定义, 即所有模型的差异分析中, 结果正确的数量占总量的比例; 公式 (4) 为精确率 (*Precision*) 的定义, 即所有被识别为"有差异"的样本中, 缺陷模型所占的比例; 公式 (5) 为召回率 (*Recall*) 的定义, 即所有将正常模型和缺陷模型对进行差异分析时, 被正确识别为"有差异"的比例; 公式 (6) 为 $F1$ 值 (*F1-score*) 的定义, 其同时考虑了精确率和召回率.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \tag{3}$$

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$F1\text{-}score = \frac{2\times Precision\times Recall}{Precision+Recall} \tag{6}$$

4.3 实验参数及解释方法选择

实验部分涉及的参数主要为置信度值的阈值α、效应量阈值β. 其中, α在实验部分被设置为常用值 0.05, 当置信度值小于或等于 0.05 时表示至少有 95% 的置信度可认为待测模型的 *MBI* 之间存在显著性差异. 与置信度值不同, 效应量通常没有固定的常用取值[51]. 因此, 实验从所有正常模型和缺陷模型中随机取 20% 进行初步实验以确定β的取值. 初步实验结果表明, 当β取 1.1 时 IADT 可达到最佳 *F*1 值. 对于不同解释方法, 实验选择使用泛用性最广且效率较高的 Grad-CAM 作为实验部分 IADT 的解释方法.

4.4 RQ1 实验结果分析

为研究 IADT 与其他模型差异性分析方法的缺陷模型检测能力, 实验使用 DeepCrime[21,22]作为比较对象. DeepCrime 提出的模型差异性分析方法被广泛用于 DNN 模型与其变异体模型之间的差异性分析工作[52–54]. 我们从 DeepCrime 的 GitHub 仓库 (DeepCrime, https://github.com/dlfaults/deepcrime) 中获取其模型差异分析的实现代码, 并使用与其论文中相同的参数设置来检测缺陷模型, 并与 IADT 的缺陷模型检测能力进行比较. 为使实验所用的原始模型和缺陷模型可应用 DeepCrime 的模型差异性分析方法, 实验对所有正常模型和缺陷模型分别重复训练 20 次并计算其测试精度, 以适配 DeepCrime 的模型差异性分析方法.

分别使用 IADT 和 DeepCrime 对缺陷模型进行检测的实验结果如表 1 和图 4 所示. 其中, 表 1 中的 *TP*、*FP*、*TN*、*FN* 为 IADT 和 DeepCrime 在不同数据集下对缺陷模型进行检测的具体结果, 图 4 及后续图中的 *P*、*R*、*F*1 和 *ACC* 分别表示不同方法检测缺陷模型的精确率、召回率、*F*1 值和准确率. 如表 1 所示, IADT 的 *TP* 和 *FN* 比 DeepCrime 少 44、1 030 和 83, 使 DeepCrime 具有比 IADT 更高的召回率. 但 DeepCrime 的 *TN* 和 *FP* 比 IADT 少 290、1 783 和 253, 使 IADT 具有比 DeepCrime 更高的精确率. 此外, IADT 在 *TN* 上的优势比在 *TP* 上的劣势更为显著, 使 IADT 具有比 DeepCrime 更高的准确率. 如图 4 所示, IADT 检测缺陷模型的召回率在 CIFAR-10、MNIST 和 SVHN 数据集分别比 DeepCrime 低 2.8%、34.8% 和 5.2%, 但精确率分别比 DeepCrime 高 13.3%、31.9% 和 10.5%. 综合来看, IADT 的 *F*1 值比 DeepCrime 高 6.4%、0.8% 和 3.8%, 准确率高 11.3%、12.6% 和 7.8%.

表 1　不同模型差异分析方法的缺陷模型检测结果

差异分析方法	CIFAR-10				MNIST				SVHN			
	TP	*FP*	*TN*	*FN*	*TP*	*FP*	*TN*	*FN*	*TP*	*FP*	*TN*	*FN*
IADT	1 522	127	468	53	1896	168	2 835	1 068	1 450	212	383	125
DeepCrime	1 566	417	178	9	2 926	1951	1 052	38	1 532	464	130	43

IADT 在 3 个数据集上均取得了高于 DeepCrime 的 *F*1 值和准确率, 这是由于使用解释方法生成的 *MBI* 能更准确地反映模型行为. 当模型行为存在较大差异时 IADT 能更准确地检测到模型行为差异, 进而具有较高的精确率. 然而, 有部分正样本中的缺陷模型与正常模型虽然在测试精度上存在较大差异, 但在解释方法生成的 *MBI* 上差异较小而没有被 IADT 检测, 进而影响了 IADT 召回率. 虽然在小部分模型上解释方法没能正确反映行为差异使 IADT 的召回率低于 DeepCrime, 但较高的 *F*1 值和准确率依然可说明 IADT 缺陷模型检测能力的优越性. IADT 在 3 个数据集上的缺陷模型检测能力具有一定差距, 虽都在 *F*1 值取得了高于 DeepCrime 的结果但在 MNIST 数据集上的性能提升低于 CIFAR-10 和 SVHN 数据集. 经分析, 这可能是由于 CIFAR-10 和 SVHN 数据集相比 MNIST 数据集具有更复杂的背景和更高维的特征, 解释方法在 MNIST 数据集上生成的类激活图中的重要区域更

集中于相似区域, 这使 DNN 模型在 MNIST 数据集的行为差异更难以反映 *MBI* 中. 这说明 IADT 更善于在使用复杂高维数据训练的 DNN 模型之间检测缺陷模型.

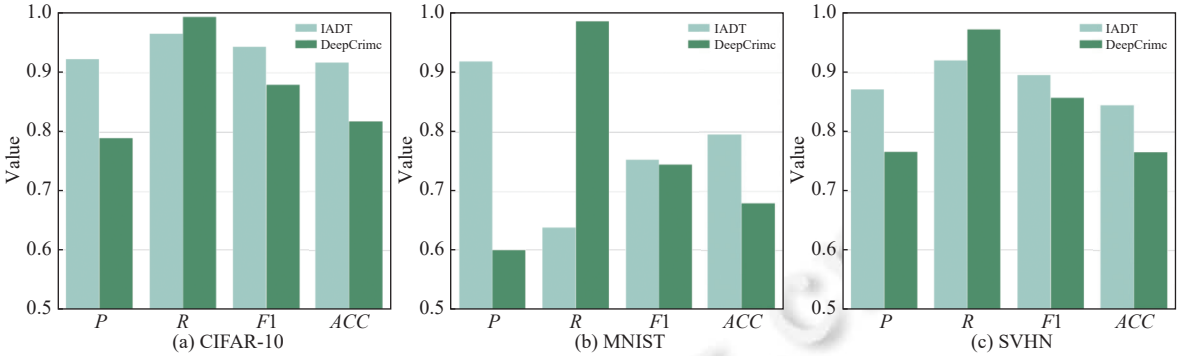

图 4 不同模型差异分析方法的真实缺陷模型检测能力对比

为进一步探究 IADT 和 DeepCrime 检测缺陷模型能力的差异, 对两个方法的缺陷模型检测情况进行了 Win/Lose/Draw 分析. 具体来说, 我们在收集表 1 数据时记录了两种方法对每个样本的判断结果, 当 IADT 判断正确且 DeepCrime 判断错误时记为 Win, 反之记为 Lose, 两种方法判断相同时记为 Draw. 记录结果如表 2 所示. 表 2 中数据为在 3 个数据集所记录到 Win、Lose 和 Draw 的数量和所占比例.

表 2 缺陷模型检测的 Win/Lose/Draw 分析

数据集	Win	Lose	Draw
CIFAR-10	606 (27%)	105 (5%)	1 494 (68%)
MNIST	2 002 (33%)	1 172 (19%)	2 871 (48%)
SVHN	477 (22%)	270 (12%)	1 458 (66%)

从表 2 可见, IADT 和 DeepCrime 对大部分样本具有相同的判断结果, Draw 的数量分别占 68%、48% 和 66%, 均高于 Win 和 Lose 的数量. 而 Win 的数量为 Lose 的 5.8 倍、1.7 倍和 1.8 倍, 占分歧样本中的大多数. 这表明 IADT 具有更高的缺陷模型检测能力. 同时, 5%、19% 和 12% 的 Lose 表明 IADT 和 DeepCrime 检测不同缺陷模型时具有一定的互补关系, DeepCrime 可正确检测部分 IADT 未能正确识别的缺陷模型.

除缺陷模型检测能力, 时间效率也是本文关注的指标之一. 为验证 IADT 相比其他模型差异性分析方法是否具有更高的时间效率, 在收集表 1 数据时统计了 IADT 和 DeepCrime 所消耗的时间. 结果如表 3 所示, 其中数据为 IADT 和 DeepCrime 在不同阶段处理单个模型平均所消耗的时间. 在模型训练和差异分析 2 个阶段, IADT 所消耗的时间均低于 DeepCrime. 在 CIFAR-10 和 MNIST 两个数据集上 IADT 所消耗的总时间分别仅为 DeepCrime 的 5.3%、4% 和 5.4%.

表 3 不同模型差异分析方法的时间效率对比 (s)

阶段	CIFAR-10		MNIST		SVHN	
	IADT	DeepCrime	IADT	DeepCrime	IADT	DeepCrime
模型训练	139.6	2 792.5	302.9	7 682.5	153.7	3 064.2
差异分析	11.1	28.9	6.4	16.1	13.6	31.7
总计	150.7	2 821.4	309.3	7 698.6	167.3	3 095.9

模型训练为模型差异分析方法的前置准备阶段, 由于 DeepCrime 需要 20 个正常子模型和 20 个缺陷子模型的测试精度以分析待测模型之间的差异, 因此需要对同一个模型重复训练 20 次获取子模型. 而 IADT 可使用单个原始模型及其变异体模型的 *MBI* 以分析模型差异. 因此 IADT 在模型训练阶段的时间消耗仅约为 DeepCrime 的 4.2%. 差异分析为模型差异分析方法主要阶段, DeepCrime 需要分别将测试集输入 20×2 个子模型并计算其测试精

度以分析其差异, 而 IADT 只需要将测试集输入 2 个待测模型即可计算 *MBI* 并分析其差异, 虽然解释方法将消耗一定时间以计算 *MBI*, 但 IADT 在模型差异分析阶段的时间消耗依然只有 DeepCrime 的 38.4%、39.8% 和 42.9%.

对 RQ1 的结论: IADT 能有效分析 DNN 模型之间的行为差异, 与 DeepCrime 的模型差异分析方法相比, 对于 CIFAR-10、MNIST 和 SVHN 数据集, *F*1 值分别高 6.4%、0.8% 和 3.8%, 准确率分别高 9.9%、11.6% 和 7.8%. 此外, IADT 无需重复训练就能直接分析两个模型之间的行为差异, 在 3 个数据集上检测缺陷模型所消耗的时间分别仅为 DeepCrime 的 5.3%、4.0% 和 5.4%, 具有较高的时间效率.

4.5 RQ2 实验结果分析

解释方法是 IADT 准确分析并反映 DNN 模型行为的重要部分. 为研究解释方法的引入对 IADT 变异体模型检测能力的影响, 实验将对比 IADT 和 IADT$^-$的变异体模型检测能力. 其中, IADT$^-$是指将 DNN 模型的预测结果代替解释方法输出的 *MBI* 进行模型差异分析以检测变异体模型. 除使用 DNN 模型的预测结果作为模型差异分析的依据以外, 其他实验设置均与 IADT 相同. 实验结果如图 5 所示, 在 CIFAR-10、MNIST 和 SVHN 数据集上 IADT 的 *F*1 值分别比 IADT$^-$高 44%、4.8% 和 26%, 准确率高 39%、2.1% 和 23.2%.

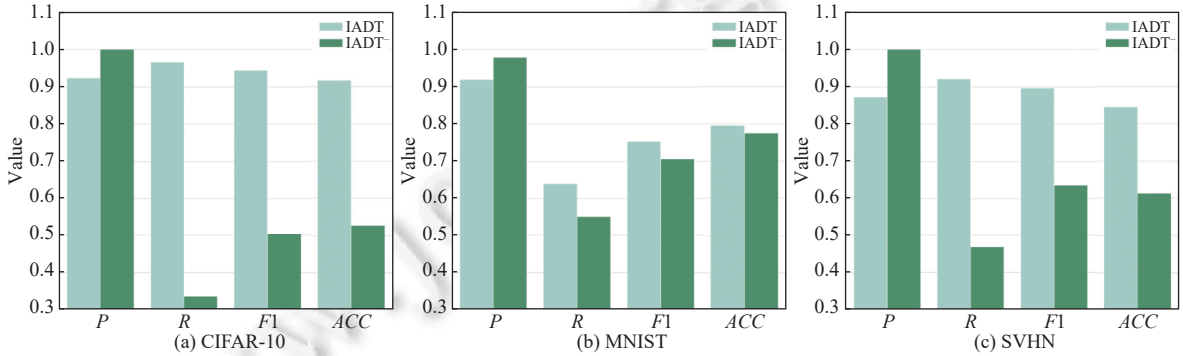

图 5 解释方法对 IADT 的真实缺陷模型检测能力影响

从图 5 可见, 去除解释方法后 IADT$^-$虽在精确率上相比完整的 IADT 具有一定提升, 但召回率远低于 IADT. 由于解释方法计算的 *MBI* 相比模型预测输出能更准确地反映 DNN 模型的行为差异. 当测试集不足以触发足够多的模型输出差异时 IADT$^-$将无法检测到缺陷模型. 因此, IADT$^-$只能检测到与正常模型的输出差异较大的缺陷模型. 这使其具有较高的精确率, 但也因为只检测到了少量缺陷模型而具有较低的召回率.

对 RQ2 的结论: 解释方法的引入提高了 IADT 的缺陷模型检测能力, 检测缺陷模型的 *F*1 值提高了 44.0%、4.8% 和 26.0%, 准确率提高了 39.0%、2.1% 和 23.2%.

4.6 RQ3 实验结果分析

近年来, 除 Grad-CAM 外有许多 DNN 解释方法被提出. 为研究不同解释方法对 IADT 缺陷模型检测能力的影响, 选择在 DNN 模型可解释性研究中被广泛使用的 Grad-CAM[13]、Grad-CAM++[42]和 Layer-CAM[43]作为 IADT 的解释方法对缺陷模型进行检测. 实验结果如图 6 所示, 对于 CIFAR-10 和 SVHN 数据集, IADT 在使用 Grad-CAM 作为解释方法时的精确率、*F*1 值和准确率均明显高于 Grad-CAM++和 Layer-CAM. 在使用 Layer-CAM 时的召回率虽然比使用 Grad-CAM 高 2.1% 和 6.6%, 但其较低的精确率使 *F*1 值和准确率均低于 Grad-CAM. 对于 MNIST 数据集, 虽然使用 3 种解释方法的差异较小, 但使用 Grad-CAM 时的 *F*1 值和准确率仍然比 Grad-CAM++和 Layer-CAM 高 1.1% 和 0.2%.

使用 3 种解释方法均可有效检测缺陷模型, 但检测能力具有一定差异, 使用 Grad-CAM 可达到比 DeepCrime 更佳的缺陷模型检测能力, 使用 Grad-CAM++或 Layer-CAM 则只能在 MNIST 数据集上取得优于 DeepCrime 的 *F*1 值, 而在 CIFAR-10 和 MNIST 数据集上的 *F*1 值比 DeepCrime 低 0.6%–4.7%. 考虑到 DNN 模型有部分行为差异是由角盒区域中的缺陷引起[27], 由于角盒区域在模型预测行为中只提供较低的贡献, 特征图中经角盒区域处理

的区域将计算得到较低的梯度. Grad-CAM 使用全局平均池化计算计算特征图的权重, 这使一些梯度较低但特征区域较大的特征图也能分配到较高的权重, 进而能在 *MBI* 中更明显地反映角盒区域的行为解释. 而 Grad-CAM++则使用反向传播提高了高梯度特征图的权重, LayerCAM 则使用元素级的权重分配方法直接对特征图中的不同区域分配权重, 降低了低梯度区域所分配的权重, 这使得低梯度的角盒区域差异难以体现在 Grad-CAM++和 Layer-CAM 生成的 *MBI* 中, 难以检测到由角盒区域中的缺陷引起的行为差异. 因此, 相比 Grad-CAM++和 Layer-CAM, Grad-CAM 能更全面地分析模型的行为解释.

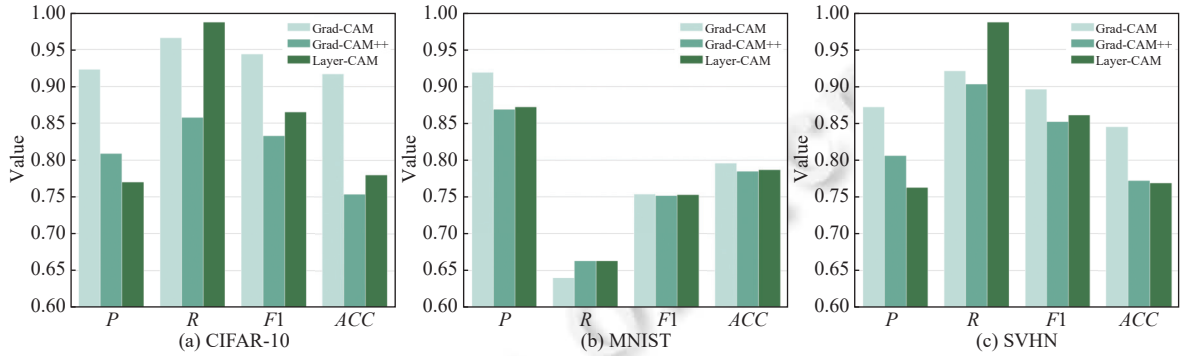

图 6 不同解释方法对 IADT 缺陷模型检测能力的影响

对 RQ3 的结论: 使用 3 种解释方法均可有效检测缺陷模型, 且当使用 Grad-CAM 时可达到优于 DeepCrime 的缺陷模型检测能力.

5 讨论及有效性分析

本节将讨论 IADT 对模型行为细微差异的检测能力以及和与 DeepCrime 的适用场景差异, 并对有效性威胁进行分析.

5.1 IADT 对细微模型差异的检测能力

在第 4 节中, 实验使用真实的缺陷模型对 IADT 的有效性进行了验证, 其中所用的缺陷模型和正常模型之间的 *MBI* 往往存在较大差异. 与上述情况不同, 为了更细粒度地度量测试数据质量, 在 DeepMutation[29]和 DeepMetis[52]等变异测试方法中, 所生成的变异体模型和原始模型之间通常仅存在细微的行为差异.

为讨论 IADT 对细微模型差异的检测能力, 我们使用正常模型作为原始模型, 并使用 DeepMutation++提供的 DNN 变异体生成工具 (DeepMutation++, https://sites.google.com/view/deepmutationpp) 对原始模型注入缺陷, 去除两个与原始模型不兼容的变异算子“删除隐藏层”和“复制隐藏层”后, 对原始模型应用了 6 种变异算子以注入缺陷, 并将变异体的精度阈值 (所生成变异体与原始模型的最低精度比例) 分别设置为 0.95、0.96、0.97 和 0.98, 其他参数保持 DeepMutation++的默认设置. 使用 6 种变异算子在 4 个精度阈值设置下对 148 个原始模型注入缺陷, 共生成了 3 503 个含有缺陷的变异体模型用于讨论 IADT 识别变异体模型的能力. 6 种变异算子的详细介绍如表 4 所示.

表 4 使用的变异算子

变异算子	简称	描述
高斯模糊 (Gaussian fuzzing)	GF	使用高斯模糊在一定范围内修改部分神经元的权重
权重乱序 (weight shuffling)	WS	选择部分神经元并打乱其与上一层连接的权重顺序
阻断神经元 (neuron block)	NB	选择部分神经元并屏蔽其输出
反转神经元激活值 (neuron activation inverse)	NAI	选择部分神经元并在输入激活函数前改变输出的符号
交换神经元 (neuron switch)	NS	随机交换同一层中部分神经元的位置
添加隐藏层 (layer addition)	LA	随机复制其中一个隐藏层

由于 DeepMutation++的设计初衷为注入微小的缺陷, 因此变异体模型与其原始模型之间的行为差异小于使用 Cohen's d 计算效应量时的有效值, 在此情况下计算效应量缺乏实际意义, 因此将 IADT 的阈值 β 和 DeepCrime 的效应量阈值和设置为 0, 阈值 α 则保持和实验部分相同的 0.05. 由于所生成的变异模型与其原始模型之间的行为差异小于不同原始模型之间的行为差异, 因此这里只讨论在原始模型及对应变异体之间行为差异分析的结果.

分别使用 IADT、IADT$^-$和 DeepCrime 对变异体模型进行识别的结果如表 5 所示, 其中精度阈值为使用 DeepMutation++所生成变异体模型与原始模型的最低精度比例, IADT、IADT$^-$和 DeepCrime 之间较高的变异体模型识别率被加粗标出. 由表 5 可见, IADT 对不同变异算子在不同精度阈值下所生成变异体模型的识别率均保持在 68% 以上, 对不同数据集的平均识别率则保持在 80% 以上. 对于 CIFAR-10 数据集, IADT$^-$对除 LA 以外的其余 5 种变异算子生成的变异体模型具有与 IADT 相近的识别率, 但对变异算子 LA 所生成变异体模型的识别率低于 IADT. DeepCrime 则仅对 WS、NB 和 NAI 这 3 个变异算子在精度阈值为 0.95 时生成 CIFAR-10 数据集变异体模型的识别率高于 IADT, 对其他变异体模型的识别率均低于 IADT.

表 5　不同模型差异分析方法的变异体识别率对比

变异算子	精度阈值	CIFAR-10			MNIST			SVHN		
		IADT	IADT$^-$	DeepCrime	IADT	IADT$^-$	DeepCrime	IADT	IADT$^-$	DeepCrime
GF	0.95	**0.743**	0.686	0.314	**0.833**	0.359	0.487	**0.829**	0.686	0.229
	0.96	0.686	**0.800**	0.229	**0.808**	0.359	0.487	**1.000**	0.771	0.200
	0.97	0.714	**0.800**	0.200	**0.885**	0.423	0.500	**0.800**	0.829	0.200
	0.98	0.743	**0.857**	0.200	**0.833**	0.333	0.436	**0.886**	0.829	0.200
WS	0.95	0.714	0.800	**0.829**	**0.910**	0.526	0.718	**0.857**	0.800	0.600
	0.96	0.714	**0.771**	0.657	**0.923**	0.500	0.769	**0.943**	0.800	0.486
	0.97	**0.829**	0.800	0.543	**0.936**	0.538	0.692	**0.971**	0.829	0.514
	0.98	**0.800**	0.771	0.314	**0.897**	0.590	0.641	**0.886**	0.743	0.343
NB	0.95	0.743	0.800	**0.829**	**0.885**	0.782	0.718	**0.829**	0.914	0.714
	0.96	**0.857**	**0.857**	0.657	**0.949**	0.667	0.756	**0.943**	0.657	0.686
	0.97	**0.800**	**0.800**	0.543	**0.974**	0.603	0.782	**0.914**	0.829	0.600
	0.98	0.686	**0.771**	0.257	**0.897**	0.628	0.731	**1.000**	0.886	0.514
NAI	0.95	0.771	**0.886**	**0.886**	**0.910**	0.526	0.821	**1.000**	0.714	0.714
	0.96	**0.857**	0.829	0.743	**0.910**	0.474	0.795	**0.914**	0.800	0.714
	0.97	**0.771**	**0.771**	0.514	**0.923**	0.526	0.731	**0.971**	0.829	0.686
	0.98	**0.800**	0.771	0.286	**0.923**	0.436	0.692	**0.971**	0.886	0.400
NS	0.95	0.800	**0.829**	0.800	**0.936**	0.436	0.795	**0.943**	0.886	0.800
	0.96	0.743	**0.800**	0.714	**0.923**	0.526	0.718	**0.914**	0.829	0.743
	0.97	**0.914**	0.743	0.486	**0.859**	0.513	0.744	**0.857**	0.829	0.629
	0.98	**0.800**	0.714	0.314	**0.923**	0.462	0.769	**0.971**	0.857	0.429
LA	0.95	**0.943**	0.686	0.057	**0.987**	0.551	0.859	**0.971**	0.829	0.543
	0.96	**0.971**	0.429	0.029	**0.974**	0.577	0.833	**0.971**	0.857	0.457
	0.97	**1.000**	0.486	0.000	**0.974**	0.449	0.808	**1.000**	0.857	0.429
	0.98	**1.000**	0.429	0.029	**0.974**	0.462	0.744	**0.943**	0.800	0.143
平均识别率		**0.808**	0.745	0.435	**0.915**	0.510	0.709	**0.929**	0.814	0.499
平均标准差		**0.050**	0.059	0.170	**0.023**	0.051	0.044	**0.053**	0.057	0.117

DeepMutation++所生成的变异体模型的测试精度与原始模型的测试精度之比被要求保持在精度阈值之上, 这使得在较高精度阈值设置下生成的变异体模型的测试精度将更接近原始模型. 而 DeepCrime 将模型测试精度的显著性差异作为识别变异体模型的依据, 在较高的精度阈值设置下生成的变异体模型与原始模型在测试精度上的差异难以被 DeepCrime 所识别. 如表 5 所示, 当在精度阈值设置为 0.97 或 0.98 时, DeepCrime 对使用 GF、NB 和 NS 这 3 个变异算子所生成变异体模型 (MNIST 数据集) 的识别率没有明显变化趋势, 但对其他变异体模型的识别率均出现明显下降. 而 IADT 直接使用 *MBI* 作为模型差异分析的依据, 不依赖于模型的测试精度, 其变异体模型

识别率与精度阈值之间则没有呈现明显相关性. 如表 5 所示, IADT 在不同数据集下的平均标准差均低于 DeepCrime, 这表明 IADT 的变异体模型识别能力更为稳定, 而 DeepCrime 则容易受到精度阈值等因素的影响.

变异算子 LA 通过随机复制 DNN 模型全连接层中的其中一个隐藏层以生成变异体模型. 与其余 5 个对神经元进行变异操作的变异算子不同, LA 作为层级的变异算子, 由于不改变任何神经元参数而使变异体模型的测试精度与原始模型之间只有少量细微差异, 导致 DeepCrime 难以识别其生成的变异体模型. 但向全连接层中添加的一个隐藏层将在对全连接层求梯度时造成较大影响, 使基于梯度的解释方法对变异体模型进行行为分析时得到与原始模型有较大差异的 *MBI*, 使得 IADT 达到了较高的变异体模型识别率.

5.2 IADT 与 DeepCrime 的适用场景差异

在第 4.4 节的实验中, 通过分析图 4 中的数据可知, IADT 由于能更细粒度的分析模型行为差异而具有较高的精确率, 但召回率低于 DeepCrime. 而在表 2 的 Win/Lose/Draw 分析中, 出现 Loss 的情况占 5%、19% 和 12%. 这说明两种方法之间存在一定的互补关系, 某些缺陷模型只能由其中一种方法检测到, 若同时使用 IADT 和 DeepCrime 识别缺陷模型可能提高有效性. 为验证这一猜想, 设计了如下两种策略以同时使用 IADT 和 DeepCrime 识别缺陷模型.

- IADT∧DeepCrime: 即当两种方法均显示有差异时, 认为待测模型之间存在差异.
- IADT∨DeepCrime: 即只要两种方法的其中一个显示有差异时, 认为待测模型之间存在差异.

使用不同模型差异分析方法的缺陷模型检测能力如表 6 所示. 其中, IADT∧DeepCrime 和 IADT∨DeepCrime 的参数设置与第 4.4 节实验中保持一致. 表中不同评价指标的最优值被加粗表示. 由表 6 可见, 同时使用 IADT 和 DeepCrime 识别缺陷模型未能在单独使用 IADT 的基础上提高 *F*1 值, 但能提高精确率或召回率. 在使用 IADT∧DeepCrime 策略时, 精确率比 IADT 提高了 1.7%、5.2% 和 7%, 使用 IADT∨DeepCrime 策略时, 召回率比 DeepCrime 提高了 0.6%、0.8% 和 2.7%.

表 6 不同模型差异分析方法的缺陷模型检测能力对比

差异分析方法	CIFAR-10				MNIST				SVHN			
	P	*R*	*F*1	*ACC*	*P*	*R*	*F*1	*ACC*	*P*	*R*	*F*1	*ACC*
IADT	0.923	0.966	**0.944**	**0.917**	0.919	0.640	**0.754**	**0.796**	0.872	0.921	**0.896**	0.845
DeepCrime	0.790	0.994	0.880	0.804	0.600	0.987	0.746	0.667	0.767	0.973	0.858	0.766
IADT∧DeepCrime	**0.940**	0.740	0.828	0.780	**0.971**	0.554	0.705	0.773	**0.942**	0.693	0.799	0.751
IADT∨DeepCrime	0.807	**1.000**	0.893	0.829	0.606	**0.995**	0.753	0.681	0.826	**1.000**	0.905	**0.850**

实验结果表明, IADT∧DeepCrime 能更精确的检测缺陷模型, 从而提高精确率, 而 IADT∨DeepCrime 则由于提高了正样本被正确预测的数量而具有更高的召回率. 但同时使用 IADT 和 DeepCrime 的 *F*1 值和准确率均低于单独使用 IADT 的结果. 因此, 当在实际应用时需要达到较高的精确率可使用 IADT∧DeepCrime 策略, 当需要达到较高的召回率时可使用 IADT∨DeepCrime 策略, 当需要兼顾精确率和召回率而达到较高的 *F*1 或准确率时可单独使用 IADT.

5.3 有效性分析

本节将对所提方法的有效性威胁进行分析.

- 内部有效性影响. 首先是实验中 DNN 模型的训练过程以及所使用的解释方法的正确性, 实验所使用的所有原始模型和缺陷模型均使用 AUTOTRAINER[48]提供的开源代码进行训练, Grad-CAM、Grad-CAM++和 Layer-CAM 这 3 种解释方法均使用第三方开源代码 (tf-keras-vis. https://github.com/keisen/tf-keras-vis) 实现, 以尽量保证模型的训练和行为解释分析的正确性. 其次是模型差异性分析和评价指标计算过程是否正确, 为尽可能确保代码实现的正确性, 对 DNN 模型的 *MBI* 差异性分析和评价指标计算代码经过了多次检查和测试.
- 外部有效性影响. 首先是实验对象及实验结果的代表性, 实验使用 AUTOTRAINER 提供的开源代码训练目标模型, 其提供的训练程序和所使用 LeNet-5 模型结构在 DNN 质量保证领域具有较为广泛的应用[15,30,55]. 同时,

作为 DNN 质量保障领域的常用数据集[52,56], 实验中使用了 CIFAR-10[44]和 MNIST[49]数据集进行模型训练和验证. 虽无法保证 IADT 在其他模型或数据集上的有效性, 但实验结果具有一定代表性. 其次是 IADT 对其他深度学习模型的泛用性, 对于非图像分类任务的 CNN 模型, 可直接使用或对现有解释方法修改后进行解释分析. 例如, 在本文之前的工作中, 我们提出了 IATG 方法, 对 Grad-CAM 进行了修改以实现对自动驾驶转向角预测 CNN 模型的解释分析, 从而针对性地生成高质量的测试数据[41]. 除 CNN 模型之外, 对于其他具有自我解释或外部解释方法的 DNN 模型, IADT 可使用相应解释方法的结果进行差异性分析, 来检测缺陷模型. 本文实验部分为使实验目标模型兼容 DeepCrime 的模型差异分析方法, 选择使用 AUTOTRAINER 提供的开源代码训练目标模型, 无法保证 IADT 在其他模型上的有效性.

● 构造有效性影响. 构造有效性的主要因素受到评价标准的影响. 实验使用准确率、精确率、召回率和 $F1$ 值作为缺陷模型检测能力的评价标准, 这 4 项指标常被用于分类算法性能评价指标[56], IADT 的模型差异分析可认为是判断待测模型之间是否存在差异的二分类算法, 因此这 4 项指标可用于 IADT 缺陷模型检测能力的评价标准.

6 总结和未来工作

为更有效地在多个 DNN 模型中检测存在缺陷的模型, 本文提出了基于解释分析的 DNN 模型差分测试方法 IADT, 利用解释方法分析 DNN 模型对于测试输入的行为解释, 并使用统计方法分析模型对测试集行为解释的显著性差异来检测缺陷模型. 与传统差分测试方法相比, IADT 摆脱了 DNN 差分测试方法对模型输出结果的依赖, 转而使用模型内部的行为解释作为判断 DNN 模型之间差异性的依据. 使用 3 种数据集和真实缺陷模型进行实验的结果表明, IADT 所使用的解释方法可有效提高对缺陷 DNN 模型的检测能力. 与 DeepCrime 的模型差异分析方法相比, IADT 具有更强的缺陷模型检测能力和更高的时间效率. IADT 的 *MBI* 降维方法能将 *MBI* 从二维矩阵降维至一维以适配统计检验方法, 但目前的降维方法会丢失部分特征信息, 影响 IADT 分析 DNN 模型之间行为差异的准确性. 未来, 我们将探索更有效的 *MBI* 降维方法使 IADT 更准确地检测缺陷 DNN 模型.

References:

[1] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436–444. [doi: 10.1038/nature14539]

[2] He KM, Zhang XY, Ren SQ, Sun J. Deep residual learning for image recognition. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016. 770–778. [doi: 10.1109/CVPR.2016.90]

[3] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego, 2015.

[4] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural Computation, 2000, 12(10): 2451–2471. [doi: 10.1162/089976600300015015]

[5] Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent neural networks. In: Proc. of the 2013 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing. Vancouver: IEEE, 2013. 6645–6649. [doi: 10.1109/ICASSP.2013.6638947]

[6] Chen JJ, Ma HY, Zhang LM. Enhanced compiler bug isolation via memoized search. In: Proc. of the 2020 IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: ACM, 2020. 78–89. [doi: 10.1145/3324884.3416570]

[7] Li X, Li W, Zhang YQ, Zhang LM. DeepFL: Integrating multiple fault diagnosis dimensions for deep fault localization. In: Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Beijing: ACM, 2019. 169–180. [doi: 10.1145/3293882.3330574]

[8] Yang YM, Xia X, Lo D, Grundy J. A survey on deep learning for software engineering. ACM Computing Surveys, 2022, 54(10s): 206. [doi: 10.1145/3505243]

[9] Zeng ZR, Zhang YQ, Zhang HT, Zhang LM. Deep just-in-time defect prediction: How far are we? In: Proc. of the 30th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2021. 427–438. [doi: 10.1145/3460319.3464819]

[10] 6park.news. Tesla hits a white truck again with two passengers rushing to ICU-Tesla Tesla electric car. 2021. https://6park.news/en/tesla-hits-a-white-truck-again-with-two-passengers-rushing-to-icu-tesla-tesla-electric-car.html

[11] Lubben A. Uber is giving up on self-driving cars in California after deadly crash. 2018. https://www.vice.com/en/article/9kga85/uber-is-giving-up-on-self-driving-cars-in-california-after-deadly-crash

[12] Wang Z, Yan M, Liu S, Chen JJ, Zhang DD, Wu Z, Chen X. Survey on testing of deep neural networks. Ruan Jian Xue Bao/Journal of

Software, 2020, 31(5): 1255–1275 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5951.htm [doi: 10.13328/j.cnki.jos.005951]

[13] Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: Proc. of the 2017 IEEE Int'l Conf. on Computer Vision. Venice: IEEE, 2017. 618–626. [doi: 10.1109/ICCV.2017.74]

[14] McKeeman WM. Differential testing for software. Digital Technical Journal, 1998, 10(1): 100–107.

[15] Pham HV, Lutellier T, Qi WZ, Tan L. CRADLE: Cross-backend validation to detect and localize bugs in deep learning libraries. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering. Montreal: IEEE, 2019. 1027–1038. [doi: 10.1109/ICSE.2019.00107]

[16] Wang JN, Lutellier T, Qian SS, Pham HV, Tan L. EAGLE: Creating equivalent graphs to test deep learning libraries. In: Proc. of the 44th IEEE/ACM Int'l Conf. on Software Engineering. Pittsburgh: IEEE, 2022. 798–810. [doi: 10.1145/3510003.3510165]

[17] Guo JM, Jiang Y, Zhao Y, Chen Q, Sun JG. DLFuzz: Differential fuzzing testing of deep learning systems. In: Proc. of the 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Lake Buena Vista: ACM, 2018. 739–743. [doi: 10.1145/3236024.3264835]

[18] Nejadgholi M, Yang JQ. A study of oracle approximations in testing deep learning libraries. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 785–796. [doi: 10.1109/ASE.2019.00078]

[19] Pei KX, Cao YZ, Yang JF, Jana S. DeepXplore: Automated whitebox testing of deep learning systems. In: Proc. of the 26th Symp. on Operating Systems Principles. Shanghai: ACM, 2017. 1–18. [doi: 10.1145/3132747.3132785]

[20] Srisakaokul S, Wu ZK, Astorga A, Alebiosu O, Xie T. Multiple-implementation testing of supervised learning software. In: Proc. of the Workshops at the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 384–391.

[21] Humbatova N, Jahangirova G, Tonella P. DeepCrime: Mutation testing of deep learning systems based on real faults. In: Proc. of the 30th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2021. 67–78. [doi: 10.1145/3460319.3464825]

[22] Jahangirova G, Tonella P. An empirical evaluation of mutation operators for deep learning systems. In: Proc. of the 13th Int'l Conf. on Software Testing, Validation and Verification. Porto: IEEE, 2020. 74–84. [doi: 10.1109/ICST46399.2020.00018]

[23] Huang XW, Kwiatkowska M, Wang S, Wu M. Safety verification of deep neural networks. In: Proc. of the 29th Int'l Conf. on Computer Aided Verification. Heidelberg: Springer, 2017. 3–29. [doi: 10.1007/978-3-319-63387-9_1]

[24] Lomuscio A, Maganti L. An approach to reachability analysis for feed-forward ReLU neural networks. arXiv:1706.07351, 2017.

[25] Bunel R, Turkaslan I, Torr PHS, Kumar MP, Lu JY, Kohli P. Branch and bound for piecewise linear neural network verification. The Journal of Machine Learning Research, 2020, 21(1): 42.

[26] Ji SL, Du TY, Deng SG, Cheng P, Shi J, Yang M, Li B. Robustness certification research on deep learning models: A survey. Chinese Journal of Computers, 2022, 45(1): 190–206 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2022.00190]

[27] Ma L, Juefei-Xu F, Zhang FY, Sun JY, Xue MH, Li B, Chen CY, Su T, Li L, Liu Y, Zhao JJ, Wang YD. DeepGauge: Multi-granularity testing criteria for deep learning systems. In: Proc. of the 33rd ACM/IEEE Int'l Conf. on Automated Software Engineering. Montpellier: IEEE, 2018. 120−131. [doi: 10.1145/3238147.3238202]

[28] Jia Y, Harman M. An analysis and survey of the development of mutation testing. IEEE Trans. on Software Engineering, 2011, 37(5): 649–678. [doi: 10.1109/TSE.2010.62]

[29] Ma L, Zhang FY, Sun JY, Xue MH, Li B, Juefei-Xu F, Xie C, Li L, Liu Y, Zhao JJ, Wang YD. DeepMutation: Mutation testing of deep learning systems. In: Proc. of the 29th Int'l Symp. on Software Reliability Engineering. Memphis: IEEE, 2018. 100−111. [doi: 10.1109/ISSRE.2018.00021]

[30] Hu Q, Ma L, Xie XF, Yu B, Liu Y, Zhao JJ. DeepMutation++: A mutation testing framework for deep learning systems. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 1158–1161. [doi: 10.1109/ASE.2019.00126]

[31] Cui ZQ, Xie RL, Chen X, Liu XL, Zheng LW. DeepRanger: Coverage-guided deep forest testing approach. Ruan Jian Xue Bao/Journal of Software, 2023, 34(5): 2251–2267 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6422.htm [doi: 10.13328/j.cnki.jos.006422]

[32] Odena A, Olsson C, Andersen D, Goodfellow I. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 4901–4911.

[33] Zhang SH, Liu S, Sun J, Chen YQ, Huang WZ, Liu JY, Liu J, Hao JY. FIGCPS: Effective failure-inducing input generation for cyber-physical systems with deep reinforcement learning. In: Proc. of the 36th IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: IEEE, 2021. 555–567. [doi: 10.1109/ASE51524.2021.9678832]

[34] Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A. Practical black-box attacks against machine learning. In: Proc. of the 2017 ACM on Asia Conf. on Computer and Communications Security. Abu Dhabi: ACM, 2017. 506–519. [doi: 10.1145/3052973.

3053009]

[35] Lee S, Cha S, Lee D, Oh H. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy. In: Proc. of the 29th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2020. 165–176. [doi: 10.1145/3395363.3397346]

[36] Xiao CW, Zhu JY, Li B, He W, Liu MY, Song D. Spatially transformed adversarial examples. In: Proc. of the 6th Int'l Conf. on Learning Representations. Vancouver: OpenReview.net, 2018.

[37] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego, 2015.

[38] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proc. of the 2017 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2017. 39−57. [doi: 10.1109/SP.2017.49]

[39] Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S. The oracle problem in software testing: A survey. IEEE Trans. on Software Engineering, 2015, 41(5): 507–525. [doi: 10.1109/TSE.2014.2372785]

[40] Yang PB, Sang JT, Zhang B, Feng YG, Yu J. Survey on interpretability of deep models for image classification. Ruan Jian Xue Bao/Journal of Software, 2023, 34(1): 230–254 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6415.htm [doi: 10.13328/j.cnki.jos.006415]

[41] Xie RL, Cui ZQ, Chen X, Zheng LW. IATG: Interpretation-analysis-based testing method for autonomous driving software. Ruan Jian Xue Bao/Journal of Software, 2024, 35(6): 2753–2774 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6836.htm [doi: 10.13328/j.cnki.jos.006836]

[42] Chattopadhay A, Sarkar A, Howlader P, Balasubramanian VN. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In: Proc. of the 2018 IEEE Winter Conf. on Applications of Computer Vision. Lake Tahoe: IEEE, 2018. 839–847. [doi: 10.1109/WACV.2018.00097]

[43] Jiang PT, Zhang CB, Hou QB, Cheng MM, Wei YC. LayerCAM: Exploring hierarchical class activation maps for localization. IEEE Trans. on Image Processing, 2021, 30: 5875–5888. [doi: 10.1109/TIP.2021.3089943]

[44] Krizhevsky A, Nair V, Hinton G. The CIFAR-10 dataset. 2014. http://www.cs.toronto.edu/~kriz/cifar.html

[45] Horn RA, Johnson CR. Norms for vectors and matrices. In: Horn RA, Johnson CR, eds. Matrix Analysis. Cambridge: Cambridge University Press, 1990.

[46] Nelder JA, Wedderburn RWM. Generalized linear models. Journal of the Royal Statistical Society. Series A (General), 1972, 135(3): 370–384. [doi: 10.2307/2344614]

[47] Wilcoxon F. Individual comparisons by ranking methods. In: Kotz S, Johnson NL, eds. Breakthroughs in Statistics: Methodology and Distribution. New York: Springer, 1992. 196–202. [doi: 10.1007/978-1-4612-4380-9_16]

[48] Zhang XY, Zhai J, Ma SQ, Shen C. AUTOTRAINER: An automatic DNN training problem detection and repair system. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering. Madrid: IEEE, 2021. 359–371. [doi: 10.1109/ICSE43902.2021.00043]

[49] LeCun Y, Cortes C, Burges CJC. The MNIST database of handwritten digits, 1998. http://yann.lecun.com/exdb/mnist/

[50] Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY. Reading digits in natural images with unsupervised feature learning. In: Proc. of the 2011 NIPS Workshop on Deep Learning and Unsupervised Feature Learning. 2011.

[51] Correll J, Mellinger C, McClelland GH, Judd CM. Avoid Cohen's 'small', 'medium', and 'large' for power analysis. Trends in Cognitive Sciences, 2020, 24(3): 200–207. [doi: 10.1016/j.tics.2019.12.009]

[52] Riccio V, Humbatova N, Jahangirova G, Tonella P. DeepMetis: Augmenting a deep learning test set to increase its mutation score. In: Proc. of the 36th IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: IEEE, 2021. 355–367. [doi: 10.1109/ASE51524.2021.9678764]

[53] Wang Z, You HM, Chen JJ, Zhang YY, Dong XY, Zhang WB. Prioritizing test inputs for deep neural networks via mutation analysis. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering. Madrid: IEEE, 2021. 397–409. [doi: 10.1109/ICSE43902.2021.00046]

[54] Hort M, Zhang JM, Sarro F, Harman M. Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods. In: Proc. of the 29th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Athens: ACM, 2021. 994–1006. [doi: 10.1145/3468264.3468565]

[55] Wardat M, Cruz BD, Le W, Rajan H. DeepDiagnosis: Automatically diagnosing faults and recommending actionable fixes in deep learning programs. In: Proc. of the 44th IEEE/ACM Int'l Conf. on Software Engineering. Pittsburgh: IEEE, 2022. 561–572. [doi: 10.1145/3510003.3510071]

[56] Ma SQ, Liu YQ, Lee WC, Zhang XY, Grama A. MODE: Automated neural network model debugging via state differential analysis and input selection. In: Proc. of the 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Lake Buena Vista: ACM, 2018. 175–186. [doi: 10.1145/3236024.3236082]

附中文参考文献:

[12] 王赞, 闫明, 刘爽, 陈俊洁, 张栋迪, 吴卓, 陈翔. 深度神经网络测试研究综述. 软件学报, 2020, 31(5): 1255–1275. http://www.jos.org.cn/1000-9825/5951.htm [doi: 10.13328/j.cnki.jos.005951]

[26] 纪守领, 杜天宇, 邓水光, 程鹏, 时杰, 杨珉, 李博. 深度学习模型鲁棒性研究综述. 计算机学报, 2022, 45(1): 190–206. [doi: 10.11897/SP.J.1016.2022.00190]

[31] 崔展齐, 谢瑞麟, 陈翔, 刘秀磊, 郑丽伟. DeepRanger: 覆盖制导的深度森林测试方法. 软件学报, 2023, 34(5): 2251–2267. http://www.jos.org.cn/1000-9825/6422.htm [doi: 10.13328/j.cnki.jos.006422]

[40] 杨朋波, 桑基韬, 张彪, 冯耀功, 于剑. 面向图像分类的深度模型可解释性研究综述. 软件学报, 2023, 34(1): 230–254. http://www.jos.org.cn/1000-9825/6415.htm [doi: 10.13328/j.cnki.jos.006415]

[41] 谢瑞麟, 崔展齐, 陈翔, 郑丽伟. IATG: 基于解释分析的自动驾驶软件测试方法. 软件学报, 2024, 35(6): 2753–2774. http://www.jos.org.cn/1000-9825/6836.htm [doi: 10.13328/j.cnki.jos.006836]

谢瑞麟(1996－), 男, 硕士生, CCF 学生会员, 主要研究领域为智能软件工程, 可信人工智能.

陈翔(1980－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为软件测试, 软件维护, 实证软件工程, 软件仓库挖掘.

崔展齐(1984－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为智能软件工程, 可信人工智能.

李莉(1992－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为数据科学与智能系统, 数据融合.