张浩严, 吕文涛, 郭庆, 等. 基于解释-蜕变测试的深度学习模型错误定位研究[J]. 智能计算机与应用,2025,15(4):184-190. DOI:10.20169/j.issn.2095-2163.250426

基于解释-蜕变测试的深度学习模型错误定位研究

张浩严[1], 吕文涛[1], 郭 庆[2], 徐羽贞[2], 余润泽[3]
(1 浙江理工大学 浙江省智能织物与柔性互联重点实验室, 杭州 310018;
2 浙江省技术创新服务中心, 杭州 310007; 3 中国移动通信集团设计院有限公司浙江分公司, 杭州 310012)

摘 要: 近年来深度学习模型层次的日益复杂,使得其预测更难以解释,同时也使得深度学习模型的缺陷检测和错误定位面临巨大挑战。目前,蜕变测试作为深度学习模型测试重要手段的同时,也与错误定位技术相结合,从而对深度学习模型可疑神经元和污染特征进行定位。以上方法主要利用蜕变关系满足与否以及神经元或特征是否覆盖进行定位计算。然而,蜕变关系满足并不代表深度学习模型是正确的。本文主要结合蜕变测试、深度学习可解释性的方法对深度学习模型的错误定位以及缺陷检测等方面进行研究讨论。

关键词: 蜕变测试; 积分梯度法; 基于频谱的错误定位技术
中图分类号: TP181 **文献标志码**: A **文章编号**: 2095-2163(2025)04-0184-07

Research on error location of deep learning model based on interpretation and metamorphic testing

ZHANG Haoyan[1], LÜ Wentao[1], GUO Qing[2], XU Yuzhen[2], YU Runze[3]
(1 Key Laboratory of Intelligent Textile and Flexible Interconnection of Zhejiang Province, Zhejiang Sci-Tech University, Hangzhou 310018, China; 2 Zhejiang Technology Innovation Service Center, Hangzhou 310007, China; 3 China Mobile Communications Group Design Institute Co., Ltd., Zhejiang Branch, Hangzhou 310012, China)

Abstract: In recent years, the increasing complexity of deep learning models has made their predictions more difficult to interpret, and has also posed significant challenges to defect detection and error localization in deep learning models. At present, as an important means of deep learning model testing, metamorphic testing is also combined with error location technology to locate suspicious neurons and pollution features of deep learning model. The above methods mainly use whether the transformation relationship is satisfied or not and whether the neurons or features cover for localization calculation. However, satisfying the transformation relationship does not necessarily mean that the deep learning model is correct. This paper mainly discusses the error location and defect detection of deep learning model based on metamorphic testing and interpretability of deep learning.

Key words: metamorphic testing; integrated gradients method; spectrum-based fault localization

0 引 言

近年来,随着机器学习的发展,机器学习相关技术在计算机视觉等多个领域取得了一系列先进成果,机器学习模型也被广泛地应用到一些重要的现实任务中,如人脸识别、自动驾驶、恶意软件检测等。在一些现实任务中,机器学习模型的表现和效率超过了人类[1]。

由于深度学习范式模型层次的日益复杂、大量数据被用于此类系统的训练和开发,使得其预测更难以解释[2],也使得深度学习模型的缺陷检测和错误定位面临严峻挑战。传统的测试技术将测试用例的执行结果与预期结果进行对比,从而发现软件的错误,但是对于一些不可测试的或者极其复杂的问题,传统的测试方法在可行性方面也有待进一步改进,这种问题被称为 Oracle 问题[3]。为了解决 Oracle 问题,引入

基金项目: 国家自然科学基金(U1709219,61601410); 浙江省科技厅重点研发计划项目(2022C01079); 2021 年产业技术基础公共服务平台项目(2021-0174-1-1)。
作者简介: 张浩严(2001—),男,硕士研究生,主要研究方向:计算机视觉,机器学习。
通信作者: 吕文涛(1982—),男,博士,副教授,主要研究方向:工业互联网,数字图像处理。Email:alvinlwt@zstu.edu.cn。
收稿日期: 2023-07-31

了蜕变测试的概念。目前，蜕变测试在作为深度学习模型测试重要手段的同时[4]，也与错误定位技术相结合，从而对深度学习模型可疑神经元和污染特征进行定位解释。以上方法主要利用蜕变关系满足与否以及神经元或特征是否覆盖进行定位计算。然而，蜕变关系满足并不代表深度学习模型是正确的。

综合以上分析论述，本课题拟结合蜕变测试和所生成的解释，对现有的错误定位技术进行优化，对造成深度学习模型错误行为的根源进行精确的定位计算。

1　基于解释-蜕变测试的深度学习模型错误定位原理

本节主要分别介绍基于解释-蜕变测试的深度学习模型错误定位技术所包含的蜕变测试、积分梯度法和 SBFL 错误定位技术原理。

1.1　蜕变测试

蜕变测试目前被广泛应用于测试 DNN，测试过程主要包括 4 个阶段[5]：筛选原始测试用例；构造蜕变关系；基于蜕变关系将原测试用例转化成衍生测试用例；用原始与衍生测试用例两次执行程序，判断程序的正确性。测试过程中最重要的就是构造蜕变关系，蜕变关系领域主要包括蜕变关系的识别与选择[4]。

1.2　积分梯度解释方法

假设有一个代表神经网络的函数 $F:R^n \to [0,1]$，令 $x \in R^n$ 作为输入，$x' \in R^n$ 作为基线输入。对于图像有关网络，基线输入通常选择黑色图像[6-7]。

考虑在 R^n 中的一条从输入 x 到 x' 的直线路径，计算路径上所有点的梯度，积分梯度是通过累积各点梯度得到的。对于输入 x 和基线 x' 沿着第 i 维的积分梯度定义如下，

$$IntegratedGrads_i(x) ::= (x_i - x_i') \times \int_0^1 \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \mathrm{d}\alpha \tag{1}$$

其中，$\frac{\partial F(x)}{\partial x}$ 表示 $F(x)$ 沿着第 i 维的梯度。

当应用于图像分类任务时，积分梯度法可以提供各个像素对于模型输出的贡献度，具体步骤如下：首先，选择一张基线图片，通常是一张全黑的图片，然后将该基线输入视为模型输入的起点。接着，将目标图片与基准点之间的路径进行分段采样，得到一系列的中间图像，其中每幅中间图像都是由当前图像和基准点按照一定比例融合得到的。对于每幅中间图像，计算该图像与基准点之间的梯度，并乘以中间图像与目标图片之间的权重，从而得到每个像素对于模型输出的贡献度。将所有中间图像的贡献度累加起来，得到每个像素总的贡献度值。

1.3　可疑神经元定位

1.3.1　错误定位原理

本文将神经网络当作“白盒”，采用基于频谱的错误定位技术（SBFL）对网络内部各个实体进行可疑值计算。SBFL 是一种基于程序覆盖率信息的缺陷定位技术，通过分析程序执行时每个语句被执行的次数，计算语句的异常得分，从而确定可能存在缺陷的语句位置。使用 SBFL 检测神经网络可疑神经元时可使用神经元激活次数替代语句执行的次数，从而得到神经元对应的 SBFL 可疑指标。

1.3.2　错误定位公式

SBFL 错误定位技术的一个重要研究领域是风险评估公式的有效性，风险评估公式的选择对实验的结果具有重要影响[8]。本实验使用 Tarantula、Ochiai 和 Jaccard 公式，进行风险评估。

（1）Tarantula 公式。具体如下：

$$\frac{ef}{ef + nf} \Big/ \left(\frac{ef}{ef + nf} + \frac{ep}{ep + np}\right) \tag{2}$$

（2）Ochiai 公式。具体如下：

$$\frac{ef}{\sqrt{(ef + nf)(ef + ep)}} \tag{3}$$

Jaccard 公式。具体如下：

$$\frac{ef}{ef + nf + ep} \tag{4}$$

其中，ef 表示激活该神经元，但违反了蜕变关系的蜕变测试组的数量；nf 表示未激活该神经元，并违反了蜕变关系的蜕变测试组的数量；ep 表示激活该神经元，并满足了蜕变关系的蜕变测试组的数量；np 表示未激活该神经元，且满足了蜕变关系的蜕变测试组的数量。

2　深度学习模型潜在错误检测算法

本节主要介绍使用蜕变测试结合积分梯度法对 PyTorch 预训练的 ResNet50、VGG16、Inception－V3 潜在错误是否存在进行检测，使用 ImageNet2012 的验证集作为测试输入，包含 50 000 张图片。

2.1　深度学习模型蜕变测试

2.1.1　蜕变关系构建

现有的基于蜕变测试 DNN 测试大多都是通过突变图像来测试图像分类模型，突变图像大多是保留语义的突变[7]，可使用如下蜕变关系构建蜕变数据集。

(1)像素级突变:通过为图像的所有像素添加相同的常数,从而改变图像的亮度。本实验中采用了提亮 1.2 倍的效果,以保证视觉上的一致,实验效果如图 1、图 2 所示。

图 1 原始测试样例

Fig. 1 Original test sample

图 2 提亮后的蜕变测试样例

Fig. 2 Metamorphic testing sample after brightening

(2)仿射变换:通过将像素矩阵转置,可实现图像旋转 90°的效果;通过将原图像划分为若干个子区域,使用每个子区域的平均像素值代替该子区域,可实现将图片缩小的效果。本实验采取图像旋转 90°、图片缩小为原先 1/2 的效果,实验效果分别如图 3~图 6 所示。

图 3 原始测试样例

Fig. 3 Original test sample

图 4 旋转后的蜕变测试样例

Fig. 4 Metamorphic testing sample after rotating

图 5 原始测试样例

Fig. 5 Original test sample

图 6 缩小后的蜕变测试样例

Fig. 6 Metamorphic testing data after reducing

2.1.2 基于蜕变测试的深度学习模型错误检测

蜕变测试被应用于测试 DNN 时,不需要预先知道数据对应真实标签,只需要将原始测试样例与蜕变测试样例的预测结果进行对比,两者相同即代表该测试样例通过,两者不同代表该测试样例没有通过测试。在本实验中,将通过蜕变测试的样例归类为 Satisfy 组,将未通过蜕变测试的样例归类为 Violate 组,并将其作为 SBFL 错误定位方法的 2 组输入。蜕变测试结果见表 1~表 3。

表 1 ResNet50 模型蜕变测试结果

Table 1 The metamorphic testing results of ResNet50 model

操作	Satisfy 个数	Violate 个数
旋转	39 006	10 994
缩小	42 698	7 302
提亮	47 569	2 431

表 2 VGG16 模型蜕变测试结果

Table 2 The metamorphic testing results of VGG16 model

操作	Satisfy 个数	Violate 个数
旋转	23 447	26 553
缩小	37 419	12 581
提亮	45 850	4 150

表 3 Inception-V3 模型蜕变测试结果

Table 3 The metamorphic testing results of Inception-V3 model

操作	Satisfy 个数	Violate 个数
旋转	25 966	24 034
缩小	40 805	9 195
提亮	46 883	3 117

然而在蜕变测试通过时,并不代表该测试用例在模型上的预测是没有问题的。本实验中,使用

satisfy 组的原始测试用例与蜕变测试用例通过 2.2 节中使用的方法生成解释并判断解释是否相同，从而确定模型是否存在隐藏缺陷（见表 4）。

表 4　判断是否有隐藏缺陷

Table. 4　Determine if there are hidden defects

	预测相同；解释相同	预测相同；解释不同	预测不同；解释不同	预测不同；解释相同
是否有缺陷？	没有	有	有	NA

2.2　深度学习模型潜在错误检测

视觉概念是人类在感知世界时所捕获的基本感知单位[7]，在图像上表示有意义的语义实例。积分梯度法使用像素贡献度标记图像的每个像素，并根据像素贡献度生成灰度热力图。由于越小的像素贡献度对结果的作用越小，本实验将由最大类间方差法得到的阈值作为支持像素与非支持像素的分类阈值，贡献度高于阈值的记为支持像素，低于阈值的记为非支持像素，并将支持像素的值设为 0（灰度图中 0 代表黑色），非支持像素的值设为 255（灰度图中 255 代表白色）。此后使用打开和关闭的操作将处理过的解释图像进行连接，从而形成视觉概念。在计算机视觉领域，通常通过计算 *IoU* 的方式来量化 2 个区域的重叠程度，本实验借鉴 Yuan 等学者[7]的结论将阈值 0.2 作为判定 2 个区域是否相同的依据，*IoU* 高于 0.2 记为解释相同，反之记为解释不相同。

当 Satisfy 组的原始测试用例以及蜕变测试用例的对应解释不同时，表示在该蜕变关系下，模型是存在隐藏缺陷的。

2.2.1　蜕变测试解释生成

本实验使用 Captum 库的积分梯度法生成样例通过模型时的像素贡献度，根据像素贡献度生成图片通过该模型时的灰度热力图。效果图如图 7、图 8 所示。

图 7　测试样例

Fig. 7　Test sample

图 8　解释图像

Fig. 8　Image after explaining

2.2.2　最大类间方差法

最大类间方差法（OSTU）[9-10]是一种用于图像阈值分割的算法，其基本思想是在图像灰度图中寻找一个合适的阈值，该阈值将图像分为 2 个类别，使得同一类别内的方差尽可能小，不同类别之间的方差尽可能大。

2.2.3　打开与关闭操作

首先介绍图片处理操作中的侵蚀与膨胀操作。为简单起见，使用真和假表示支持像素和非支持像素。

图片侵蚀的原理是将图像中每个像素的值与其周围邻域内的像素进行比较，若该像素所处邻域内所有像素的值都为 1，则该像素值不变；否则，将该像素值设为 0。这样可以消除或缩小图像中小的亮区域或暗区域，从而使边界更加清晰。

图片膨胀的原理是将图像中每个像素的值与其周围邻域内的像素进行比较，若该像素所处邻域内存在一个像素的值为 1，则该像素值为 1；否则，将该像素值设为 0。这样可以填充或扩大图像中小的空洞或断裂区域，从而使图像更加连续。

侵蚀和膨胀是图像处理中常见的 2 种操作，且都使用一个内核滑动输入图像，得到一个输出图像，本实验采用的内核大小为 3×3。基于此，再遵循这 2 个基本操作的顺序，可以得到以下 2 个高级操作。

图片关闭操作是对图片先进行膨胀后、再进行侵蚀，打开操作则相反。为了将支持像素转换为视觉概念，图片首先应用关闭，然后应用打开，即可得到对应视觉概念，效果图如图 9、图 10 所示。

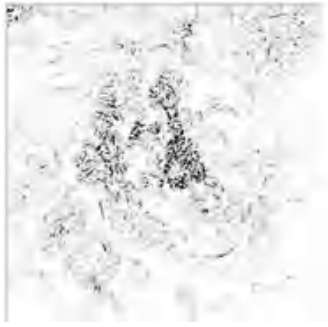

图 9　解释图像

Fig. 9　Image after explaining

图 10 视觉概念

Fig. 10 Visual concept

2.2.4 *IoU* 计算

IoU(Intersection over Union)是一种用于衡量目标检测算法性能的评价指标。其计算方法为:将模型预测出的边界框和真实标注的边界框进行比对,计算二者交集面积与并集面积之间的比例。本实验用其量化 2 个区域的重叠程度,设 A 为区域一,B 为区域二,*IoU* 计算公式如下:

$$IoU = \frac{A \cap B}{A \cup B} \tag{5}$$

理论上,*IoU* 的取值范围为[0,1],且越接近 1,表示模型的性能越好,因为预测结果与真实结果的重合度越高。在本实验中,将 *IoU* 阈值设置为一个特定的值 0.2[7],用以判定 2 张解释图片是否相同。当 *IoU* 大于等于阈值时,认为解释相同;反之,则认为解释不同,*IoU* 计算结果见表 5 ~ 表 7。

表 5 ResNet50 模型 *IoU* 计算结果

Table 5 The *IoU* calculation results of ResNet50 model

操作	*IoU* < 0.2 个数	*IoU* < 0.2 百分比
图片旋转	37 460	0.96
图片缩小	16 668	0.39
图片提亮	2 688	0.05

表 6 VGG16 模型 *IoU* 计算结果

Table 6 The *IoU* calculation results of VGG16 model

操作	*IoU* < 0.2 个数	*IoU* < 0.2 百分比
图片旋转	22 486	0.95
图片缩小	18 323	0.48
图片提亮	1 476	0.03

表 7 Inception-V3 模型 *IoU* 计算结果

Table 7 The *IoU* calculation results of Inception-V3 model

操作	*IoU* < 0.2 个数	*IoU* < 0.2 百分比
图片旋转	25 919	0.99
图片缩小	32 401	0.86
图片提亮	3 721	0.07

从以上数据得出结果,在 3 个蜕变关系中,图片旋转的检错能力较强。

3 深度学习模型错误定位算法

本节主要使用 SBFL 错误定位技术,实现对 ResNet50、VGG16、Inception-V3 模型的故障定位。

3.1 可疑神经元定位

本小节主要介绍对各个模型如何选取需要故障定位的神经元或参数。

3.1.1 ResNet50 模型错误定位

ResNet[9](Residual Network)是近年来非常流行的深度神经网络模型之一,主要用于图像分类和目标检测等任务。在本节中,将介绍如何使用 SBFL 技术对 ResNet50 进行错误定位。

ResNet50 由 50 个卷积层组成,其中每个卷积层都包含多个残差块。在残差块中,输入信号会经过一条直接连接,并与输出信号相加,最终得到残差信号。因为神经元的激活需要经过激活函数的作用,所以本实验对可疑神经元定位时将 Bottleneck 模块(PyTorch 设计的 ResNet50 模型中每个 Bottleneck 模块的输出都经过了 ReLU 激活函数的作用)的输出作为神经元是否激活的依据,值为 0 代表未激活,值大于 0 代表已经激活。

本实验中使用 PyTorch 训练好的 ResNet50 模型,同时使用 SBFL 技术来定位模型中的错误。SBFL 技术可以通过收集测试用例的执行结果和神经元覆盖信息,来计算每个代码行的可疑度值,进而确定可能出错的神经元,本实验的流程步骤具体如下。

(1)收集测试用例的执行结果:将蜕变测试得到的 Satisfy 组和 Violate 组作为 SBFL 技术的输入,并分别记录每个神经元的激活次数。

(2)收集神经元激活信息:对于 ResNet50 模型,本实验通过自定义 *hook* 函数,并且对于 ResNet50 模型的 Bottleneck 模块注册 *hook* 函数进而得到该模块的输出,收集到神经元的激活信息。

(3)计算可疑度值:根据 Satisfy 组和 Violate 组得到的神经元激活次数,可以使用 SBFL 技术计算每个神经元的可疑度值。本实验采用了 Tarantula、Ochiai、Jaccard 三种可疑度计算公式。

(4)确定可疑神经元位置:根据计算得到的可疑度值,确定可能出错的神经元。一般来说,可疑度值越高的神经元越有可能是错误的。

通过以上步骤,就可以使用 SBFL 技术对 ResNet50 模型进行错误定位,确定可能存在问题的神经元。

3.1.2　VGG16 模型错误定位

VGG16 是一个深度卷积神经网络模型，具有 16 个卷积层，其中包括 13 个卷积层和 3 个全连接层。

同样地，本实验对可疑神经元定位时将 ReLU 函数的输出作为神经元是否激活的依据，值为 0 代表未激活，值大于 0 代表已经激活。本实验通过自定义 *hook* 函数，并且对于 VGG16 模型的 ReLU 层注册 *hook* 函数进而得到该层的输出，收集到神经元的激活信息。其余步骤与 ResNet50 模型一致。

3.1.3　Inception-V3 模型错误定位

Inception-V3 是一个深度卷积神经网络模型，由 Google 团队在 2015 年提出，用于图像分类和目标检测等任务。模型具有 22 层，其中包括 13 个卷积层和 9 个全连接层。Inception-V3 模型的特点是采用了 Inception 模块，即在同一层中使用不同尺寸的卷积核，以捕获不同大小的特征。

由于 Inception-V3 模型中没有激活函数，本实验对该模型的错误定位以参数是否为 0，作为是否覆盖的判断标准。参数为 0 代表未覆盖该参数，反之，则代表该参数被覆盖。Inception - V3 的 BasicConv2d 模块注册自定义的 *hook* 函数，进而得到该层的输出，其余步骤同 ResNet50 模型。

3.2　实验结果及分析

本实验采用了 Tarantula、Ochiai、Jaccard 三种可疑度计算公式，由于模型每层的输出是一个四维数组，故而选择以四维向量的形式表示各个不同的神经元或参数，部分实验结果见表 8~表 10，其中排序代表神经元或参数在本层实体而非整个模型中的可疑度排序。

表 8　ResNet50 模型错误定位部分结果

Table 8　Partial results of ResNet50 model error location

层数	神经元	Tarantula 公式		Ochiai 公式		Jaccard 公式	
		可疑度	排序	可疑度	排序	可疑度	排序
0	(0, 0, 0, 0)	0.490 839 772	743 328	9.72E-06	743 328	0.171 342 66	588 404
0	(0, 0, 0, 1)	0.490 834 139	743 358	9.72E-06	743 357	0.164 690 29	609 950
0	(0, 0, 0, 2)	0.490 465 898	745 094	9.71E-06	745 094	0.157 132 20	633 311
0	(0, 0, 0, 3)	0.488 544 481	752 596	9.65E-06	752 596	0.155 491 08	637 838
0	(0, 0, 0, 4)	0.492 099 306	737 342	9.76E-06	737 342	0.157 507 75	632 164
0	(0, 0, 0, 5)	0.493 885 111	729 129	9.81E-06	729 129	0.158 688 62	628 484
0	(0, 0, 0, 6)	0.492 344 997	736 194	9.76E-06	736 194	0.157 490 94	632 217
0	(0, 0, 0, 7)	0.490 805 415	743 484	9.72E-06	743 487	0.156 924 14	633 933
0	(0, 0, 0, 8)	0.491 068 215	742 217	9.72E-06	742 217	0.157 230 29	633 001

表 9　VGG16 模型错误定位部分结果

Table 9　Partial results of VGG16 model error location

层数	神经元	Tarantula 公式		Ochiai 公式		Jaccard 公式	
		可疑度	排序	可疑度	排序	可疑度	排序
0	(0, 0, 0, 0)	0.482 580 75	2 945 592	9.67E-06	2 945 592	0.379 315 76	1 801 368
0	(0, 0, 0, 1)	0.506 818 95	955 244	1.01E-05	955 242	0.525 538 56	2
0	(0, 0, 0, 2)	0.506 779 80	956 930	1.01E-05	956 931	0.525 495 65	3
0	(0, 0, 0, 3)	0.506 269 90	980 554	1.01E-05	980 557	0.524 575 95	10
0	(0, 0, 0, 4)	0.506 221 10	983 138	1.01E-05	983 140	0.524 959 50	6
0	(0, 0, 0, 5)	0.505 967 26	997 120	1.01E-05	997 121	0.523 960 95	25
0	(0, 0, 0, 6)	0.506 014 90	994 337	1.01E-05	994 336	0.524 238 05	18
0	(0, 0, 0, 7)	0.506 648 66	962 580	1.01E-05	962 582	0.524 884 05	7
0	(0, 0, 0, 8)	0.506 699 74	960 306	1.01E-05	960 306	0.525 096 83	4
0	(0, 0, 0, 9)	0.507 006 70	947 667	1.01E-05	947 668	0.525 603 40	1

表 10 Inception-V3 模型错误定位部分结果

Table 10 Partial results of Inception-V3 model error location

层数	参数	Tarantula 公式		Ochiai 公式		Jaccard 公式	
		可疑度	排序	可疑度	排序	可疑度	排序
0	(0, 0, 0, 0)	0.502 620 94	213 246	1.01E-05	213 246	0.465 740 56	242 910
0	(0, 0, 0, 1)	0.502 588 00	213 551	1.01E-05	213 551	0.465 536 86	244 262
0	(0, 0, 0, 2)	0.502 284 80	216 157	1.00E-05	216 157	0.464 972 67	247 535
0	(0, 0, 0, 3)	0.501 874 10	218 766	1.00E-05	218 766	0.464 277 90	249 698
0	(0, 0, 0, 4)	0.501 859 07	218 849	1.00E-05	218 849	0.464 201 00	249 895
0	(0, 0, 0, 5)	0.501 770 30	219 316	1.00E-05	219 316	0.463 909 12	250 417
0	(0, 0, 0, 6)	0.501 906 04	218 602	1.00E-05	218 603	0.463 876 78	250 469
0	(0, 0, 0, 7)	0.502 059 20	217 657	1.00E-05	217 657	0.464 028 72	250 198
0	(0, 0, 0, 8)	0.502 084 00	217 482	1.00E-05	217 482	0.464 030 53	250 196
0	(0, 0, 0, 9)	0.502 232 50	216 505	1.00E-05	216 505	0.464 109 70	250 072

从仿真实验内容来看,Tarantula 和 Ochiai 公式的结果排序基本近似或者相同,而 Jaccard 公式与前两者相比差距较大。Tarantula 公式和 Ochiai 公式都是基于以下假设:一个程序元素(如一行代码)在出错版本中引起错误的概率远高于未出错版本中的概率。这 2 个公式的主要区别在于对未被执行的程序元素进行处理的方式不同。Jaccard 公式与这 2 个公式不同,是将测试用例出错信息和程序元素覆盖信息组合在一起来计算错误定位公式,并且相较于前两者的计算公式考虑的因素更少、更简单。所用的计算方式是将出错测试用例涉及的所有程序元素相对于所有程序元素的并集求交集。因此,在某些情况下,当测试用例涉及的程序元素数量较少时,Jaccard 公式的结果可能与 Tarantula、Ochiai 公式的结果有很大不同。这表明使用错误定位技术时需要考虑不同的指标,并选择适合特定问题的指标来获得更好的结果。由此得出实验结论,在深度学习模型错误定位中,Tarantula 和 Ochiai 公式的效果要优于 Jaccard 公式。

4 结束语

本文针对深度学习模型预测难以解释、缺陷检测和错误定位面临挑战的问题,引入了蜕变测试的概念,并结合深度学习可解释性的方法对此进行了研究讨论。具体来说,本文选取了图片旋转、缩小、提亮三种蜕变关系生成蜕变数据集,并分别对 ResNet50、VGG16 和 Inception-V3 模型进行蜕变测试。通过对每个蜕变关系的 Satisfy 组和 Violate 组进行可解释性分析,本文判断了模型是否有隐藏缺陷以及蜕变关系的检错能力,并得到了神经元或参数可疑排序。本文的研究具有重要意义,可以为深度学习模型的下一步改进提供理论依据和实验支持。

参考文献

[1] 纪守领, 李进锋, 杜天宇, 等. 机器学习模型可解释性方法, 应用与安全研究综述[J]. 计算机研究与发展, 2019, 56(10): 2071-2096.

[2] LINARDATOS P, PAPASTEFANOPOULOS V, KOTSIANTISs S. Explainable AI: A review of machine learning interpretability methods[J]. Entropy, 2020, 23(1): 18.

[3] CHEN T Y, TSE T H. New visions on metamorphic testing after a quarter of a century of inception[C]//Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2021: 1487-1490.

[4] 田甜, 杨秀婷, 王安轼, 等. 蜕变测试研究进展及其在并行程序测试中的研究展望[J]. 软件学报, 2023 ,34 (1):130-149 .

[5] CHEN T Y, KUO F C, LIU Huai, et al. Metamorphic testing: A review of challenges and opportunities[J]. ACM Computing Surveys (CSUR), 2018, 51(1): 4.

[6] SUNDARARAJAN M, TALY A, YAN Qiqi. Axiomatic attribution for deep networks[J]. arXiv preprint arXiv, 1703.01365, 2017.

[7] YUAN Yuanyuan, PANG Qi, WANG Shuai. Unveiling hidden DNN defects with decision-based metamorphic testing[C]//37th IEEE/ACM International Conference on Automated Software Engineering. New York: ACM, 2022: 1-13.

[8] XIE Xiaoyuan, CHEN T Y, KUO F C, et al. A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2013, 22(4): 31.

[9] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2016: 770-778.

[10] OTSU N. A threshold selection method from gray-level histograms[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1979, 9(1): 62-66.