计算机科学
COMPUTER SCIENCE

基于神经元覆盖指标的测试用例生成优化研究

肖子勤, 史涯晴, 曲豫宾

引用本文

肖子勤, 史涯晴, 曲豫宾. 基于神经元覆盖指标的测试用例生成优化研究[J]. 计算机科学, 2025, 52(11): 339-348.

XIAO Ziqin, SHI Yaqing, QU Yubin. Research on Optimization of Test Case Generation Based on Neuron Coverage Index [J]. Computer Science, 2025, 52(11): 339-348.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240900006

基于神经元覆盖指标的测试用例生成优化研究

肖子勤　史涯晴　曲豫宾
陆军工程大学指挥控制工程学院　南京 210007
(1561294988@qq.com)

摘　要　深度神经网络(Deep Neural Networks,DNNs)已在诸多领域实现广泛应用,因其复杂性和不确定性,对其进行测试显得尤为重要。传统的测试方法过于依赖单一指标,无法全面揭示深度神经网络的完整行为模式。因此,需综合考量不同的覆盖指标,以便更全面地评估模型性能。结合6种多粒度的深度神经网络覆盖指标,优化模糊测试的变异策略和种子选择等步骤,生成高质量且高覆盖率的测试用例。在MNIST和CIFAR10数据集上对4种不同复杂性的模型进行实验,将原始训练集和新生成的有效测试用例合并用于重训练模型,以提高分类准确率。实验结果显示,该方法可以显著提高覆盖率,并通过自适应重训练优化模型提高了分类准确率。

关键词:神经网络;图像分类;模糊测试;变异策略;测试用例生成
中图分类号　TP311.5

Research on Optimization of Test Case Generation Based on Neuron Coverage Index

XIAO Ziqin,SHI Yaqing and QU Yubin
College of Command and Control Engineering,Army Engineering University,Nanjing 210007,China

Abstract　DNNs have been widely applied in many fields,and testing them is particularly important due to their complexity and uncertainty. Traditional testing methods rely too much on a single indicator and cannot fully reveal the complete behavioral patterns of deep neural networks. Therefore,it is necessary to comprehensively consider different coverage indicators to more comprehensively evaluate the performance of the model. It combines six multi-granularity deep neural network coverage metrics,optimizes the mutation strategy and seed selection steps of fuzzy testing,generates high-quality and high-coverage test cases. Experiments are conducted on four models of different complexities on the MNIST and CIFAR10 datasets. The original training set and newly generated effective test cases are combined for retraining the model to classification accuracy. The experimental results show that this method can significantly improve coverage and classification accuracy by optimizing the model through adaptive retraining.

Keywords　Neural networks,Image classification,Fuzzy testing,Mutation strategies,Test case generation

1　引言

深度神经网络在图像识别、自然语言处理和语音识别等领域得到显著的发展。近年来,研究者提出了一系列用于测试用例充分性的覆盖指标,例如神经元覆盖率(Neuron Coverage,NC)[1],该指标是一种用于评估深度神经网络在输入空间中的覆盖程度的指标。深度神经网络中神经元激活值的分布对于揭示极限行为具有关键意义。受到传统测试覆盖概念的影响,通过分析神经元激活值的统计分布以及连续层之间神经元激活值的变化关系,定义一系列基于神经元激活值的结构化测试覆盖指标,这些指标能够界定测试输入在DNN中的覆盖范围。基于此,提出了一种以覆盖为导向的测试输入生成方法。

然而,由于DNN的复杂性和不确定性,其鲁棒性评估变得尤为重要。传统的测试方法往往只关注模型的神经元覆盖率,而忽视了其他重要的指标,难以全面评估DNN的鲁棒性并提升模型的预测精度。神经元覆盖率类似于传统代码覆盖率,但由于深度学习中大多数规则是从训练数据中学习的,这与传统软件测试方法有差异,因此,代码覆盖率并不是评估DNN的良好指标[2]。模型测试的关键问题在于选择高质量的测试用例,以揭示模型中的潜在缺陷。依赖单一指标进行评价时,模型可能会针对该指标进行过度优化,而忽视其他关键因素。此外,单一指标评价也可能无法充分反映数据集中样本偏差的情况。较高的神经元覆盖率通常意味着网络具有更好的泛化能力和鲁棒性,能够处理更多样化的输入。然而,目前对DNN的测试大多基于单一指标,指标设计及类别划分相对粗糙且有限[3]。主要问题如下:

1)传统的变异策略可能会导致生成的测试样本粗糙且覆盖率不足,因此,生成和选择有效的测试用例是一个亟待解决的关键问题。

到稿日期:2024-09-02　返修日期:2024-11-27
通信作者:史涯晴(cuterabbitlele@qq.com)

2)生成的测试用例是深度学习模型测试中的重要部分之一,充分利用生成的测试用例来提高模型的性能和覆盖率是另一个关键问题。

本文研究的总体框架如图 1 所示,通过优化变异策略和种子选择等方法,以多个覆盖指标作为反馈机制,来优化生成的测试用例,使其在保留语义的同时实现高覆盖率,并进一步利用梯度提升算法来提高覆盖率,最后通过重训练模型增强其鲁棒性。本文提出一套综合方法来提高深度学习模型测试的质量和效率。该方法通过结合优化的变异策略、种子选择、覆盖指标反馈、梯度提升算法以及模型重训练,生成更高质量的测试用例,并最终得到一个更鲁棒的深度学习模型。

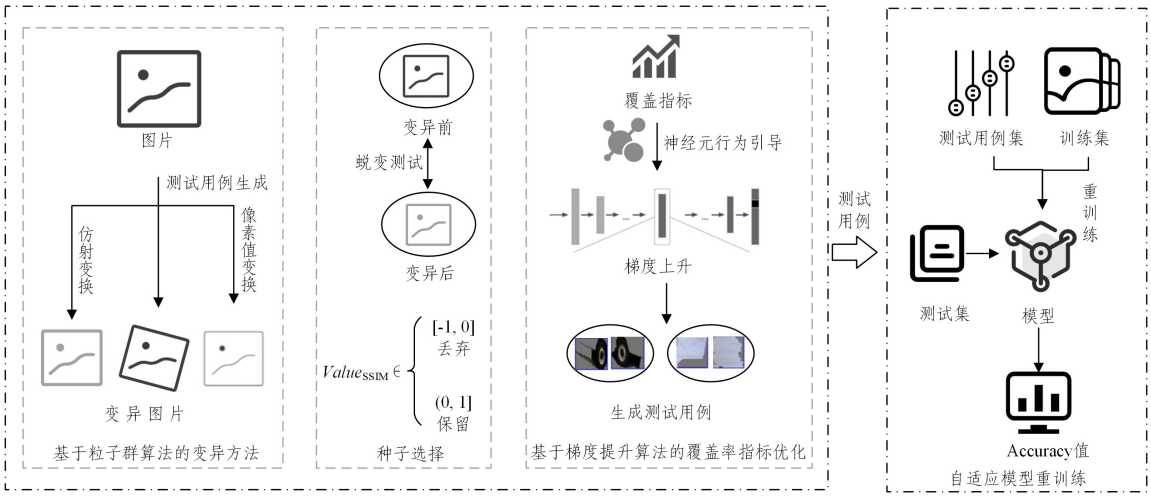


图 1 总体研究框架

Fig. 1 Overall research framework

2 相关工作

目前,深度神经网络测试输入生成方法主要包括两类:1)基于传统软件测试的方法,它通过对给定种子输入进行特定变换来最大化模型覆盖率,从而生成测试输入,这类方法被称为基于覆盖的测试输入生成;2)基于对抗的测试输入生成方法,它利用机器学习和深度学习的概念,通过向原始样本添加小扰动来生成测试输入,以诱发 DNN 错误分类,这类方法包括数据增强、生成和正则化等[4]。

由于深度学习模型结果的不确定性,传统软件测试逻辑无法直接适用。因此,需要通过神经元覆盖率等指标来评估测试用例集的充分性。DeepXplore[5] 是首个针对真实世界 DNN 的白盒测试框架,引入神经元覆盖指标并通过差异测试方法寻找产生高覆盖率的测试输入。为了提高 DNN 模型的鲁棒性,即在存在扰动情况下维持正确性的能力,需要通过对抗鲁棒性测试来确保模型不易受到环境或输入扰动的影响。一些研究提出了专门针对 DNN 的测试框架和方法,例如 DeepHunter[6] 利用覆盖制导模糊测试方法,系统地生成新的测试输入来提高目标覆盖率,并准确评估 DNN 模型质量和检测错误行为。神经元覆盖率通过激活神经元的比例来衡量测试输入对 DNN 内部逻辑的覆盖程度,包括 KMNCov, NBCov, SNACov 等不同覆盖标准。Ma 等[7] 扩展了神经元覆盖的概念,提出多粒度测试标准 DeepGauge,通过统计不同神经元的输出并动态分配激活阈值的方法,提高对对抗性样本的捕捉能力。此外,还有一些研究专门针对循环神经网络(Recurrent Neural Network, RNN)的测试覆盖准则,例如状态级别和转换级别的覆盖准则,捕捉其动态的状态转换行为,并通过自动化测试框架 DeepCruiser[8] 来生成大规模测试输入。为了更细粒度地分析单个测试输入的充分性,Yi 等[9] 提出了 SADL 框架,引入意外充分性和意外覆盖两种指标,通过比对测试输入与训练数据的差异来表示意外值。

白盒测试框架 DeepCT[10] 受到传统组合测试方法的启发,提出了一系列测试覆盖准则及相应测试用例生成方法。它们对神经元输出进行离散化,并基于组合测试思想定义了两种 t-way 覆盖标准,还提供了基于约束求解的测试用例生成方法。该方法使用两个 DNN 模型,在 MNIST 数据集上与随机测试方法进行对比实验。结果表明,相比于随机测试方法,DeepCT 在两个测试覆盖标准上均获得了更高的覆盖率,同时检测到更多的对抗性样本。

综上所述,当前 DNN 测试覆盖指标的研究主要集中在神经元覆盖率及其扩展、对抗性测试方法,以及面向特定网络结构如 RNN 的覆盖准则等方面。这些研究为提升 DNN 模型的鲁棒性和测试充分性提供了重要的理论基础和实践方法[11]。

3 基于覆盖指标的测试用例生成优化方法

本文通过神经元覆盖指标引导变异操作生成新的测试用例,结合随机变异和基于规则的变异方法,优化种子变异策略,并利用梯度上升、学习率调度及数据增强等手段提高覆盖率。同时,使用多个覆盖标准作为反馈指导测试用例生成。最后,对模型进行重训练以调整和改进,提升其测试通过率和性能指标。模糊测试是一种通过生成大量随机数据来应对 DNN 输入范围广泛问题的方法[12]。它生成大量随机输入以测试系统故障,使用由指定约束组成的覆盖指标来随机改变输入,并通过快速近似最近邻算法来计算测试覆盖率。模糊测试通过变异生成新的测试用例,并采用多个覆盖标准作为条件引导测试用例生成。然而,模糊测试的局限在于,它不能

保证达到测试目标。

模糊测试通过变异产生新的保留语义的测试集，利用多种覆盖标准作为反馈，从多角度指导测试生成。模糊测试框架包括种子的选择、种子变异方法及覆盖引导准则。初始种子用于生成后续测试用例，变异种子则是初始种子经过变异算子新生成的数据。在预训练模型的基础上，种子选择策略的多样性和复杂性对测试结果有着直接的影响[13]。种子选择策略包括随机选择和基于概率选择。1）随机选择作为一种基础而广泛使用的策略，通过随机函数返回一个介于一定范围内的数值来选取状态。这种方式简单易行，能够在没有先验知识的情况下进行状态或路径的选择，适用于初始任务[14]。2）基于概率选择是指如果种子被变异多次，则降低该种子被选择的概率，从而达到生成更多不同测试用例的目的。该策略通过降低多次变异种子的选择概率来生成更多不同测试用例。结合状态的历史信息、路径发现的新路径数量以及未处理路径等因素，通过计算状态分数来决定下一个状态的选择。使用神经元覆盖指标作为反馈引导，指导种子的筛选。变异方法包括图像仿射变换和像素值变换，通过图像变异函数限制变异范围，保证语义不变。覆盖引导准则采用多个神经元覆盖准则作为反馈指引种子筛选，进一步优化种子优先级排序，使得其最终在相同的时间内达到更高的神经元覆盖率。模糊测试优化方法研究框架如图 2 所示。

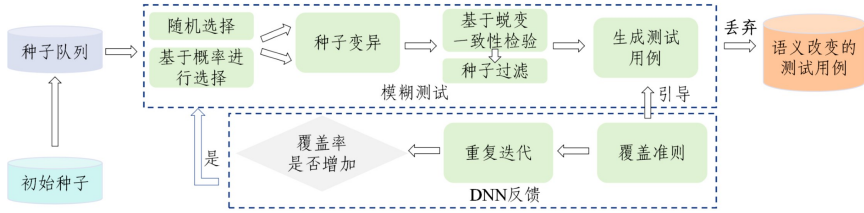


图 2 模糊测试优化方法研究框架

Fig. 2 Framework of fuzzy testing optimization method

3.1 覆盖指标

本文融合 DeepXplore，DeepHunter 以及 DeepGauge 中提出的 NC，TKNC，BKNC，SNAC，NBC 和 KMNC 指标展开研究，涵盖 6 个不同粒度的测试准则，以进一步指导测试生成过程。

相关度量指标定义如下：设 $N=\{n_1,n_2,\cdots\}$ 是 DNN 的一组神经元，设 $T=\{x_1,x_2,\cdots\}$ 为一组测试输入，$\phi(x,n)$ 表示函数，函数代表在给定测试输入 $x\in T$ 时，$n\in N$ 为神经元的输出。深度神经网络可被划分为主要功能区（Major Function Region）和边界区（Corner-case Region）[15]。其中，主要功能区为输入与训练数据分布相近就能很容易触发的神经元覆盖的部分。对于神经元 n，high 和 low 代表边界值，其分别来自对训练集的分析。边界区为输入与训练数据分布相近但很难触发的神经元覆盖的部分。$[low_n,high_n]$ 表示主要功能区的区间。对于测试输入 $x\in T$，若 $\exists n\in N:\phi(x,n)\in(-\infty,low_n)\cup(high_n,+\infty)$，则说明 DNN 落在边界区 $(-\infty,low_n)\cup(high_n,+\infty)$。如图 3 所示，深度神经级别分为主功能区域和边界区，针对主功能区域提出 k 区域神经元覆盖；针对边界区，提出神经元边界覆盖和强活跃神经元覆盖，对于层级别提出 top-k 神经元覆盖。

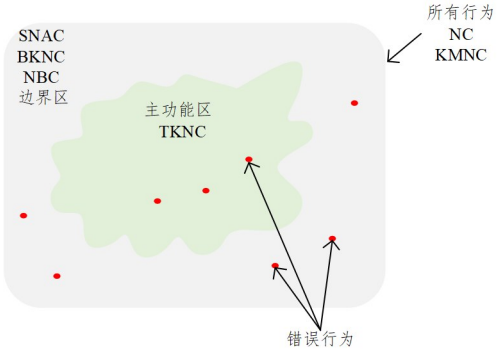


图 3 覆盖指标分类图

Fig. 3 Distribution of coverage indicators

1）神经元覆盖率

神经元覆盖率统计在测试集输入下，激活值超过给定阈值的神经元占比，计算方式为满足条件的神经元数量与神经元总数的百分比。因此，将单个输入的覆盖率定义为：设 D 为训练后的 DL 模型，由 N 个神经元组成，$activate(n,x)$ 为真的充要条件是：当 x 作为输入传递给 D 时，神经元 n 被激活。由此得到式（1）：

$$NCov(T,x)=\frac{|\{n\mid\forall x\in T,\phi(x,n)>t\}|}{|N|} \tag{1}$$

其中，T 代表已有测试样本的集合，x 代表单个测试样本，n 代表单个神经元，$\phi(x,n)$ 代表在该测试样本作用下该神经元的激活值，t 代表给定阈值，N 代表模型内的神经元总数。

2）K 多节神经元覆盖率

将神经元的输出由原来的“激活”与“非激活”两种简单状态进一步细化为由训练阶段输出的[最小值，最大值]定义的取值区间，并将该区间等分成若干小段，然后输入测试数据，统计每个神经元的输出落在哪些小段中，最后计算出被覆盖的小段占总段数的比例。K 多节神经元覆盖（K-Multisection Neuron Coverage，KMNC）[6-7] 把每个神经元的输出从最低到最高切出 k 个部分，每个神经元 n 的 K 多节覆盖表示为：

$$\frac{|\{S_i^n\mid\exists x\in T:\phi(x,n)\in S_i^n\}|}{k} \tag{2}$$

神经元区间覆盖率相对更复杂，需要记录当前测试的深度学习模型在输入已有测试样本集合之后所有神经元的激活值的分布，然后将最大值与最小值之间的区域均等地划分为给定数量的区间，计算有激活值分布的区间数量与区间总数的比例，得到单个神经元区间覆盖率，最后统计所有神经元的区间覆盖率的平均值，得到模型的神经元区间覆盖率。进一步将 DNN 的 K 多节覆盖表示为：

$$KMNCov(T,k)=\frac{\sum_{n\in N}|\{S_i^n\mid\exists x\in T:\phi(x,n)\in S_i^n\}|}{k\times|N|} \tag{3}$$

其中，T 代表已有测试样本的集合，k 代表给定的区间数量，n 代表单个神经元，S_n^i 代表单个神经元的激活区间，$\phi(x,n)$ 代表在该测试样本作用下该神经元的激活值，N 代表模型内的神经元总数。

3）神经元边界覆盖率

神经元边界覆盖（Neuron Boundary Coverage，NBC）定义为，被覆盖边界情况与边界情况总数的比率，用于度量给定测试输入集 t 覆盖边界区域（上边界和下边界值）的情况，UpperCornerNeuron 为边界区的上界，LowerCornerNeuron 为边界区的下界，分别表示为：

$$LowerCornerNeuron=\{n\in N\mid \exists x\in T:\phi(x,n)\in(-\infty,low_n)\} \tag{4}$$

$$UpperCornerNeuron=\{n\in N\mid \exists x\in T:\phi(x,n)\in(high_n,+\infty)\} \tag{5}$$

4）强神经元激活覆盖

强神经元激活覆盖（Strong Neuron Activation Coverage，SNAC），对于输出超过最大值的神经元，将其视为过分活跃的神经元，这种神经元在模型内传递有用的决策信息，被称为“强神经元”，用于衡量给定测试输入 t 覆盖边界的情况，其计算式为：

$$SNACov(T)=\frac{|UpperCornerNeuron|}{|N|} \tag{6}$$

统计输出在区间外的神经元占神经元总数的比例，计算式为：

$$NBCov(T)=\frac{|UpperCornerNeuron|+|LowerCornerNeuron|}{2\times|N|} \tag{7}$$

5）top-k 高活跃神经元的比例（Top-k Neuron Coverage，TKNC）

网络层级覆盖率进一步拓展到考虑单个网络层内的所有神经元，给定数值 k，当前测试的深度学习模型在输入单个测试样本之后，如果某个神经元的激活值位于其所在的网络层内所有神经元激活值大小的前 k 位，那么就认为该神经元满足激活条件。将测试集输入待测模型后，统计每层前 k 个激活值最高的神经元组成的集合规模，并与该层神经元总数进行比较分析。当输入已有的测试样本集合之后，可以计算满足激活条件的神经元的数量与网络层内神经元总数的比例，得到单个网络层的覆盖率。然后统计所有网络层覆盖率的平均值，得到模型的网络层级覆盖率，其计算方式为 $top_k(x,i)$，表示给定测试输入 x 下，第 i 层中激活值最大的 k 个神经元。最后计算每层中 top-k 的神经元个数与神经元总数之比，计算式为：

$$TKNCov(T,k)=\frac{|\bigcup_{x\in T}(\bigcup_{1\leqslant i\leqslant l}top_k(x,i))|}{|N|} \tag{8}$$

其中，T 代表已有测试样本的集合；k 代表给定的top-k数值；x 代表单个测试样本；$top_k(x,i)$代表在输入单个测试样本的条件下，激活值位于所在网络层前 k 位的神经元的集合；N 代表模型内的神经元总数。

6）Back-k 低活性神经元的比例

Back-k 低活性神经元覆盖率（Back-k Neuron Coverage，BKNC）即在测试集输入模型后，各神经网络层中激活强度位于后 k 位的神经元数量与该层神经元总数的比值。当输入已有的测试样本集合之后，可以计算满足激活条件的神经元的数量与网络层内神经元总数的比例，得到单个网络层的覆盖率。然后统计所有网络层覆盖率的平均值，得到模型的网络层级覆盖率，其计算方式为 $back_k(x,i)$，表示给定测试输入 x 下，第 i 层中激活值最小的 k 个神经元。最后计算每层中 Back-k 的神经元个数与神经元总数之比：

$$BKNCov(T,k)=\frac{|\bigcup_{x\in T}(1\bigcup_{i\leqslant 1\cup l\geqslant i}back_k(x,i))|}{|N|} \tag{9}$$

3.2 测试用例生成方法

3.2.1 基于粒子群算法的变异方法优化

本文利用变异算子对图像进行修改，最终保证图像关键特征和类别标签不变。设计的变异算子包括噪声添加、亮度调整、对比度调整、旋转、缩放等，使用的变异算子如表 1 所列。

表 1 变异算子及其作用描述

Table 1 Mutation operators and function description

变异方法	变异算子	作用描述
仿射变换 G	VerticalFlip	模拟垂直翻转
	HorizontalFlip	模拟水平翻转
	Tanspose	模拟转置
	ShiftScaleRotate	模拟平移缩放旋转
	ImageCompression	模拟图像压缩
像素值变换 P	Brightness	模拟亮度变换
	Contrast	模拟对比度变换
	ElasticTransfom	模拟弹性变换
	CLAHE	模拟自适应直方图均衡化
	GaussianBlur	模拟高斯模糊
	RandomGamma	模拟伽马模糊

变异算子与原始的大规模数据集分布相似，每个变异算子都需要定义生成变异图像的蜕变关系。将变异算子应用于训练集生成新的图像数据集，并使用图像识别模型对比变异前后的分类结果。通过不断优化变异算子和图像识别模型，确保变异不改变图像语义，从而提升模型在实际应用中的表现[16]。最后使用测试集评估变异模型的识别准确性，并分析错误注入的程度以评估测试集质量。此方法的局限性在于基本突变算子覆盖的深度学习系统有限，注入的错误可能无法完全代表真实错误。给定图像 i，赋值器生成新图像 i'，从人的角度看，i 和 i' 的语义是相同的。变换手段包括像素值变换与仿射变换，变异操作的流程如图 4 所示。

粒子群优化算法（Particle Swarm Optimization，PSO）是一种基于群体智能的优化方法，它通过模拟鸟群、鱼群等社会生物的行为来解决优化问题。在图像处理中，PSO 用于优化变异过程中的权重和学习因子，从而使变异后的图像更接近原始图像。选择 PSO 进行优化，主要是基于其收敛速度快、全局搜索能力强以及参数设置相对简单的特点。与遗传算法（GA）相比，PSO 在处理高维优化问题时具有更高的效率，因为它通过粒子间的信息共享来加速搜索过程。此外，PSO 的收敛性更容易控制，避免了 GA 中可能出现的早熟收敛问题。基于粒子群优化算法用于搜索最佳参数的函数，使得变异后

的图像与原始图像之间的差异最小，如算法 1 所示。该算法以参数范围、图像、变异函数、神经元覆盖率路径、获取函数和覆盖函数作为输入，通过改进的粒子群算法来增强其性能。加权变异的 WVPSO 算法，结合自适应惯性权重和自适应学习因子，通过算术交叉的变异和自然选择机制的替换策略来提高粒子多样性，然后加入高斯扰动使粒子更容易跳出局部最优解。因此使用粒子群优化算法来寻找图像变异过程中的最佳参数值，是一种有效的策略。通过改进的标准 PSO 算法、加权变异的 WVPSO 算法和其他多种变异策略，可以在保证收敛速度和精度的同时，减少变换后图像与原始图像之间的差异，从而实现更优质的图像处理效果。使用变异算子对输入图像进行变异，生成新变异的图像。Distance 用于计算原始图像和变异后的图像之间的距离，度量两个图像之间的差异，计算原始图像和变异后的图像之间的差异度量，并返回这个值作为适应度值。最后，返回找到的最佳变异参数值。

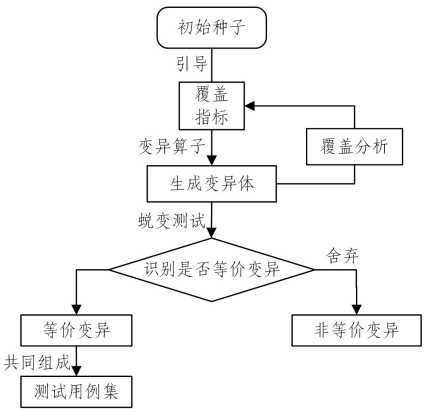


图 4 变异操作流程图

Fig. 4 Flowchart of mutation operation

算法 1 基于粒子群算法的变异方法优化

输入：param_range，label，transformation，coverage_function

输出：覆盖率值，测试用例

1. 初始化

设置粒子群规模 N

2. 对于每个粒子 i(i=1 到 N)：

3. 初始化粒子位置 x_i 和速度 v_i

X_i = 随机初始化位置(param_range)

V_i = 随机初始化速度(param_range)

4. 计算粒子的适应度值 F_i(目标函数的值 distance)

trans_image = transformation(image，x_i)

F_i = coverage_function(trans_image)

5. 将 F_i 设置为粒子的个体最优值 $pBest_i$，X_i 设置为 $pBest_i$ 的位置

$pBest_i = F_i$

$pBest_i\_pos = X_i$

6. 比较 $pBest_i$ 与全局最优值 gBest，如果 F_i 更好，则更新 gBest

如果 i=1 或者 F_i 比 gBest 好：

gBest = F_i

gBest_pos = X_i

7. 迭代过程

对于每一代 t(t=1 到最大迭代次数)：

对于每个粒子 i(i=1 到 N)：

8. 更新变异速度和位置，计算变异速度更新项(通常包括惯性项、个体最优项和全局最优项)

w = 惯性权重

c_1 = 个体学习因子

c_2 = 社会学习因子

r_1 = 随机数(0，1)

r_2 = 随机数(0，1)

$V_i(t+1) = w * V_i(t) + c_1 * r_1 * (pBest_i\_pos - X_i(t)) + c_2 * r_2 * (gBest\_pos - X_i(t))$

9. 更新变异位置

$X_i(t+1) = X_i(t) + V_i(t+1)$

10. 边界处理

确保 $X_i(t+1)$ 在 param_range 范围内

11. 计算新的适应度值

trans_image = transformation(image，$X_i(t+1)$)

$F_i(t+1)$ = coverage_function(trans_image)

12. 更新个体最优和全局最优

如果 $F_i(t+1)$ 比 $pBest_i$ 更优：

$pBest_i = F_i(t+1)$

$pBest_i\_pos = X_i(t+1)$

如果 $F_i(t+1)$ 比 gBest 更优：

gBest = $F_i(t+1)$

gBest_pos = $X_i(t+1)$

13. 返回全局最优解 gBest“最佳变异参数”

3.2.2 蜕变测试检查语义

测试预言(Test Oracle)是软件测试过程中用于验证待测系统是否按照预期正确运行的方法，通过比较实际输出与预期输出来判断测试用例是否通过[17]。本文的测试预言为图像经过一系列变异操作后分类标签是否改变。针对神经网络的测试预言问题，使用蜕变测试(Metamorphic Testing)，通过定义蜕变关系来生成新的测试用例，用于检测变异前后样本语义是否改变。决策边界用于分类问题中，在特征空间内根据不同特征对样本进行分类，不同类型间的分界就是模型针对该数据集的决策边界[18]。通过决策边界可以可视化分类结果，并直接根据样本在特征空间的位置对该样本的类型进行预测。在得到不同的变异算子子集后，需要保证生成的变异样本的决策边界扰动具有多样性。图 5 以二分类任务为例，变异方法与对抗攻击的决策边界的对比展现了其异同。

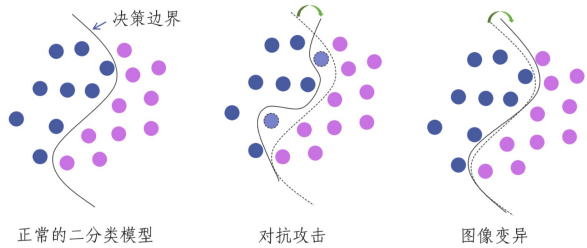


图 5 变异方法与对抗攻击二分类对比

Fig. 5 Comparison of mutation method and adversarial attack binary classification

将原始测试用例(初始种子)进行变异，依据构建的蜕变关系进行图像变异和自动判断 DNN 的输出结果。蜕变测试可以有效地检测和确保图像识别系统在面对图像变异时，仍能保持高准确性。假设 p 是初始种子映射到变异种子的数

学表示，即 $p[i]=o$ 时，假设 f_i 和 f_O 代表对输入和输出域进行特定转换的函数，满足下列关系：

$$\forall i, p[f_i(i)]=f_O(p[i]) \tag{10}$$

针对图像识别任务，定义初始种子和变异种子之间的蜕变关系为：

$$\forall i\in\mathbb{I}\wedge\forall\tau\in\mathbb{T}, DNN[\tau(i)]=DNN[i] \tag{11}$$

其中，$\mathbb{T}$表示像素值变换及仿射变换。基于这种蜕变关系，通过初始种子生成的变异样本被映射到这种蜕变关系中，在后续的测试中通过分析模型的行为差异来验证。

基于所定义蜕变关系中的输出关系，验证程序输出。若蜕变关系未被满足，则放弃该变异图片，其一致性度量如式(12)所示：

$$IB(DNN,\mathbb{I})=\sum_{i\in\mathbb{I}} f(\mid DNN[i]-DNN[\tau(i)]\mid)>\epsilon \tag{12}$$

对于具有蜕变关系的初始种子和变异种子，若模型的输出在某一误差范围内，则该模型的行为是一致的；若超出阈值，则放弃该变异后的样本。由此，通过对比神经网络的输出与真实标签的差距来判断测试是否通过。

如算法 2 所示，它包括变异样本生成、样本选择和结构相似性指数(Structural Similarity Index，SSIM)[19]计算。对于一个 k 分类的任务，输入为类数 k、训练集 TR、训练程序 TP、原始 DNN 模型 m、测试集 TE、变异算子子集 MOG。算法的最终输出为 SSIM 值的集合 DVS，其中 DVS_i 表示第 i_{th} 类别的决策边界扰动多样性，双重筛选机制的目的是确保生成的图像在视觉和语义上都与原始图像保持一定的一致性。第一轮筛选中，计算每个变异样本与原始图像之间的边界距离，如果某个样本的边界距离超过阈值，就认为这个样本与原始图像的差异过大，将其舍弃。在第二轮筛选中，计算每个通过第一轮筛选的变异样本与原始图像的 SSIM 值。如果某个样本的 SSIM 值低于预定的阈值，就认为这个样本可能会改变原始图像的语义，将其筛除。

算法 2 蜕变测试方法筛选变异样本

输入：分类类别数 k，训练数据 TR，训练程序 TP，DNN 模型 m，测试数据集 TE，变异操作子集 MOS

输出：img_list

```
1. 步骤 1：变异样本生成
2. 初始化集合 MUT={};
3. for each operator op in MOS:
4.     if op∈MO_d: // 数据层
5.         使 TR 突变以产生 MUT_1;
6.     else if op∈MO_n: // 神经元层
7.         使 m 突变以产生 MUT_2;
8.     else: // 模型层
9.         使 m 突变以产生 MUT_3;
10.    MUT=MUT_1∪MUT_2∪MUT_3;
11. 步骤 2：样本选择
12.   for i in all classes(1,k):
13.     初始化第 i_th类距离集合 DS_i={};
14.     for each 测试用例 t in TE:
15.       for each mutantm′ in MUT:
16.         if m(t)=i and m′(t)! =i:
17.           计算 dis(t,i,m,m′);
18.           adddis(t,i,m,m′) to DS_i;
19. 步骤 3：SSIM 值计算
20. for i in all classes(1,k):
21.     calculate SSIM(DS_i,MUT);
22.     将 SSIM(DS_i,MUT)加入到 DVS;
23.     output DVS.
24. 步骤 4：样本筛选
25.   if 符合蜕变关系：
26.     将变异后的 item_img 加入到 img_list
27.   end if
28.   output img_list
```

算法 2 主要包含以下 3 个步骤：

步骤 1 变异样本的生成过程。对于 MOS 中的每个变异算子，算法会生成相应的变异样本，并将其添加至 MUT。在执行变异操作时，算法会自动选择与当前变异算子相匹配的目标对象进行修改。

步骤 2 样本选择。从第 i 类($1\leqslant i\leqslant k$)的测试数据 t 进行分类。如果 t 被分类器 m 正确分类，但没有被另一个分类器 m' 正确分类，那么可以认为在 t 上，第 i 类的决策边界出现了扰动。接下来，依据式(13)和式(14)，计算 m 和 m' 的边界距离，并将这个距离值添加到第 i 类的距离集合 $dis(t,i,m,m')$ 中。其中，DS_i 表示所有类别在变异样本集 MUT 上的决策边界距离集合，DS_i 专门包含第 i 类数据的决策边界扰动距离。

进一步，定义 t 在 m 和 m' 之间关于类别 i 的决策边界扰动距离(Decision Boundary PerturbationDistance，DBPD)，如式(15)所示，在对不同变异算子子集进行评估时，对生成样本集的决策边界扰动多样性进行量化。

$$d_1=\left|y_i-\frac{1}{k}\right| \tag{13}$$

$$d_2=\left|y_i'-\frac{1}{k}\right| \tag{14}$$

$$dis(t,i,m,m')=d_1+d_2 \tag{15}$$

步骤 3 SSIM 值计算。对于距离集合 DS_i($1\leqslant i\leqslant k$)，使用 SSIM 来衡量决策边界扰动程度。SSIM 是一种用于评估两幅图像相似度的指标，主要用于衡量图像失真前后的相似性，并广泛应用于图像处理领域。SSIM 基于结构信息的退化来评估图像质量，这种方法更符合人类视觉系统的特性。具体来说，SSIM 通过比较两幅图像的亮度(Luminance)、对比度(Contrast)和结构(Structure)来计算相似度。在计算亮度时，SSIM 使用图像像素值的均值来进行评估。对比度则通过标准差来衡量，其反映了图像像素值的波动幅度。结构的比较则依赖于协方差，这有助于识别场景中物体的结构属性。综合这 3 个因素，SSIM 能够对图像质量进行全面评估。给定两张图像，结构相似性可通过式(16)求出：

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{16}$$

其中，μ_x 是 x 的平均值；μ_y 是 y 的平均值；σ_x^2 是 x 的方差；σ_y^2 是 y 的方差；σ_{xy} 是 x 和 y 的协方差。$c_1=(k_1L)^2$；$c_2=(k_2L)^2$ 是用来维持稳定的常数，L 是像素值的动态范围。$k_1=0.01$，$k_2=0.03$ 时，SSIM 值为 $-1\sim1$。当两张图像一模一样时，

SSIM 的值等于 1。结构相似度指数从图像组成的角度将结构信息定义为独立于亮度、对比度的，反映场景中物体结构的属性，并将失真建模为亮度、对比度和结构 3 个不同因素的组合。具体而言，其用均值作为亮度的估计，标准差作为对比度的估计，协方差作为结构相似程度的度量。

3.3 基于覆盖指标引导的测试用例生成

3.3.1 基于梯度提升算法的覆盖指标优化

采用蒙特卡罗树搜索（Monte Carlo Tree Search，MCTS），结合选择目标特征、变异方式和区域，优化测试的有效性。每次迭代选择一个图像区域进行变异，并反馈测试结果，直至满足停止条件。图 6 展示了基于梯度提升算法的覆盖指标优化的流程。

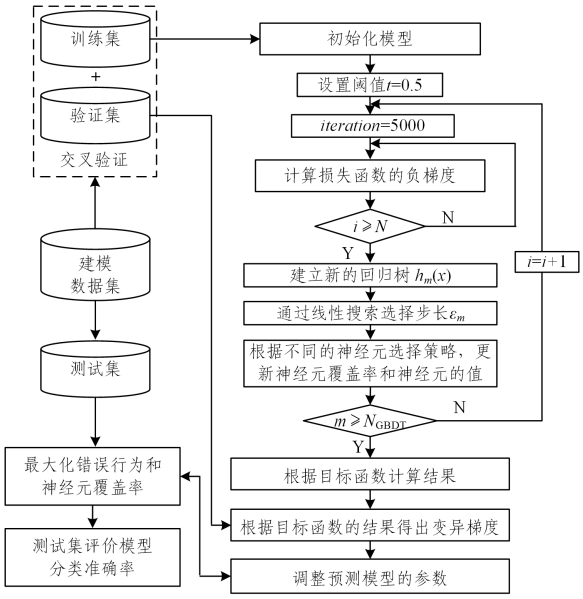


图 6 基于梯度优化的覆盖指标引导生成

Fig. 6 Coverage index guided generation based on gradient optimization

为了增强测试的有效性，将目标特征用于下一步的变异操作，并将测试结果反馈给 MCTS。采用梯度上升法沿着覆盖率的目标函数的梯度方向进行搜索，以找到最大值。每次迭代中，梯度上升法通过步长与梯度的乘积更新参数向量，迭代过程持续进行直到满足最大迭代次数。梯度上升法通过将步长（α）与梯度（$\nabla_w f(w)$）相乘，并将结果与当前的参数向量（w）相加更新参数。这个过程中，步长（α）决定每次迭代的步幅大小。迭代过程将持续进行，直到满足最高迭代次数 5 000 才停止。梯度上升法的迭代计算式为：

$$w = w + \alpha \nabla_w f(w) \tag{17}$$

参数向量（w）被更新为其原始值加上步长（α）乘以目标函数在当前参数下的梯度（$\nabla_w f(w)$）。采用增加多样性的正则化策略，以激活那些以往难以触发的神经元，从而实现神经元的多样化表现。

3.3.2 自适应模型重训练策略

通过预处理新数据，使其与预训练模型的输入要求匹配，对模型进行重训练。微调是一种重要的模型调整技术，它涉及在预先训练好的模型上进行进一步训练，此过程通常依赖于大量与特定任务相关的数据，这些数据用于指导模型更准确地完成特定类型的任务[20]。然而，如果微调数据集不够多样化，则可能限制模型的泛化能力和性能。使用多指标引导生成的测试用例来丰富微调数据集是提高模型性能的有效策略。选择性解冻预训练模型层以微调整个模型，减少迭代次数。在验证和测试集上评估模型性能，并调整参数和超参数以获得最佳性能。自适应模型重训练策略是针对已经训练好的模型，通过一系列方法重新训练模型以提高其适应性和性能的过程。

多指标引导生成高覆盖率的测试用例，增加训练数据的多样性和数量，从而提高模型的泛化能力。根据新数据的特性，调整模型的超参数（如学习率、批量大小、迭代次数等）以优化模型性能。使用增量学习的方法，在保持原有知识的基础上，逐步学习新数据，以避免“灾难性遗忘”现象。

重训练的研究框架如图 7 所示。首先，假设本文将对深度学习模型 m 进行调整与改进，该模型对应的测试用例集为 t。在深度学习模型 m 上，测试用例集 t 的准确率为 p。

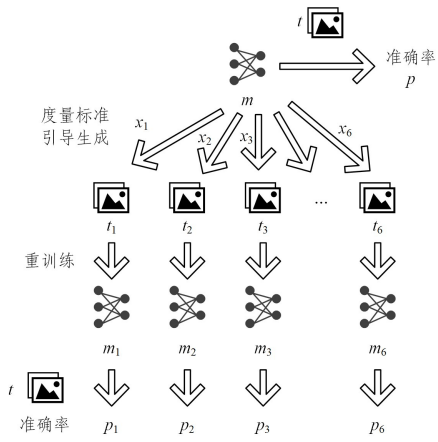


图 7 覆盖指标引导生成及重训练的研究框架

Fig. 7 Technical roadmap for guiding the generation and retraining of coverage indicators

将 NC，TKNC，BKNC，SNAC，NBC 和 KMNC 这 6 种深度神经网络的度量标准定义为 $x_1, x_2, \cdots, x_i, \cdots, x_6$，使用其度量标准引导生成新的测试用例集 $t_1, t_2, \cdots t_i, \cdots, t_6$。使用 t_i 对原深度学习模型 m 进行重新训练，采用常规的优化算法和训练技巧，如随机梯度下降、学习率调整等来训练模型，对模型进行调整和改进。在重训练过程中，对模型进行评估，测量新的性能指标，并与初始性能进行比较。根据评估结果，可以进一步调整和优化训练策略，反复迭代训练过程，以达到更高的性能，从而修复深度学习模型的缺陷。重训练完成后，得到新的模型 $m_1, m_2, \cdots, m_i, \cdots, m_6$。使用测试用例集 t 对各个重训练后的模型 m_i 进行测试，得到 t 在各个模型上的准确率，分别为 $p_1, p_2, \cdots, p_i, \cdots, p_6$。

其次，由覆盖指标 X 引导生成新的测试用例集 T，使用新的测试用例集对原模型 m 进行重训练，以进行调整和改进，得到重训练后的模型 M'。再使用测试用例集 t 对重训练后的模型 M' 进行测试，得到 t 在新模型上的准确率 P。衡量改进程度，研究重点在于测试用例生成和优化，通过神经元行为模式引导生成测试用例，并利用这些用例重训练模型。

在已训练好的网络上进行修改，冻结原网络的部分层；训

练新添加部分，解冻原网络的部分层；联合训练解冻层和新添加部分。由于重训练数据量不大，因此通过正则化 L_1 和 L_2 避免过拟合，限制模型复杂度。L_1 正则化与权重系数的绝对值成正比，L_2 正则化与权重系数的平方成正比（又称权重衰减），神经网络的 L_2 正则化也叫权重衰减（Weight Decay）。范数惩罚项指模型权重参数每个元素的平方和与正的常数的乘积，计算式如下：

$$L(w,b)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(w^{\mathrm{T}}x^{(i)}+b-y^{(i)})^2 \tag{18}$$

其中，w 是权重参数；b 是偏差参数；样本 i 的输入为 $x^{(i)}$，标签为 $y^{(i)}$，样本数为 n。自适应模型重训练策略是一个综合性的过程，通过不断地调整和优化模型的结构、训练策略、损失函数等方面，可以显著提高模型的适应性和性能。

4 实验及结果分析

4.1 实验环境与数据集

本实验在 2 个流行的数据集和 4 个具有不同复杂性的 DNN 上进行，使用多粒度测试标准（神经元级别和层级别），即 6 种深度神经网络的覆盖标准度量方法（NC，TKNC，BKNC，SNAC，NBC 和 KMNC）引导生成新的测试用例集，从而引导生成更高效的测试用例集。本实验在 Windows 10 操作系统上进行，硬件环境为 Intel Core i7-10875H @ 2.30GHz 八核 CPU 和 NVIDIA GeForce RTX 3070 GPU。编程工具为 Python 3.7，深度学习框架为 TensorFlow。实验数据集选用 MNIST 和 CIFAR-10。MNIST 数据集包含 60000张 28×28 像素的手写数字图像用于训练，10000 张用于测试，分为 0－9 共计 10 类。针对 MNIST 数据集，训练 LeNet-5 以及自定义的层数为 3，5，10 的隐藏层模型。CIFAR-10 数据集包含 60000 张 32×32 像素的彩色图像，分为飞机、鸟、狗、马等 10 类，50000 张用于训练，10000 张用于测试。针对 CIFAR-10 数据集，使用 VGG-16 和 ResNet-20 模型。超参数设置方面，神经元覆盖阈值为 0.5，K-多段神经元覆盖中 K 值设为 1000，神经元边界覆盖上下限分别为最大最小激活值加 0.5 倍标准差。训练过程记录覆盖指标变化，保存能增加覆盖率的新测试用例。

4.2 实验结果

实验结果如表 2 所列，在不同数据集和模型上，6 个覆盖准则中的大部分都可以达到较高的覆盖率峰值。NC 指标反映神经元整体活跃程度，KMNC 关注神经元在不同激活区间的分布，NBC 衡量神经元边界覆盖情况，SNAC 关注强激活神经元覆盖比例，TKNC 和 BKNC 分别从高活跃性和低活性角度评价神经元表现。

表 2 指标覆盖率结果

Table 2 Results of indicator coverage

（%）

模型	方法	NC	TKNC	BKNC	SNAC	NBC	KMNC
LeNet4	初始样本	5.8	8.33	9.42	0.86	0.66	0.38
	TensorFuzz	88.6	59.45	47.29	10.58	0.78	5.47
	DeepTest	86.6	48.52	22.45	4.07	5.21	16.48
	本文方法	86.52	74.13	39.64	30.50	10.97	18.05
LeNet5	初始样本	6.59	7.71	6.36	0.53	0.49	0.33
	TensorFuzz	67.83	83.67	13.80	5.14	4.47	55.33
	DeepTest	68.53	83.75	45.57	4.23	4.16	29.58
	本文方法	70.93	80.81	35.04	6.77	3.37	14.75
VGG16	初始样本	10.46	10.21	1.96	0.21	0.21	46.70
	TensorFuzz	18.14	66.16	14.29	23.74	8.82	78.19
	DeepTest	26.47	67.85	2.45	3.71	13.05	55.22
	本文方法	28.44	72.45	2.08	26.57	42.10	64.25
ResNet20	初始样本	40.91	56.91	16.13	0.24	0.21	0.32
	TensorFuzz	51.22	66.16	56.42	11.93	62.33	62.33
	DeepTest	48.87	67.91	48.41	67.91	41.85	41.85
	本文方法	68.49	70.64	99.92	57.46	45.54	38.03

实验发现，SNAC 由于异常点概率较小，覆盖率一直较低；NBC 因神经元值超出训练集边界的情况不普遍，覆盖率也低。对于同一种覆盖准则，覆盖率峰值在不同模型中有差异但不大，模型层数对覆盖率峰值影响较小。然而，单一指标在多个模型和数据集上展示出有效性，但其泛化能力不足。在不同的模型和数据集上，覆盖率的表现不一致。例如，NC 在 LeNet4 和 ResNet20 上覆盖率较高，但在 LeNet5 和 VGG16 上相对较低，且覆盖率提升后模型准确率反而下降。TKNC 作为全局覆盖指标，在所有模型和数据集上表现平稳，而 BKNC 在 ResNet20 上表现突出。

为提高神经元覆盖指标的泛化能力，可以尝试结合多个指标，或者开发新的自适应指标，动态调整测试策略。尽管不同覆盖准则在相同模型下的上升趋势不同，有些覆盖准则的覆盖率随测试用例增加稳定上升到峰值，但各指标的表现具有偶然性，不具备泛化能力，具体结果及原因分析如表 3 所列。

表 3 结果及原因分析

Table 3 Results and cause analysis

结果	原因
在 LeNet4 的 NC 实验中，覆盖率提升效果不佳	覆盖指标阈值的设置方式导致在小型模型上难以有效覆盖深层状态，且覆盖率提升阈值会提前触发，导致难以生成能够提高覆盖率的样本，进而造成奖励反馈机制失灵，最终使变异策略退化为随机变异
在 KMNC 和 TKNC 上，可以稳定提高覆盖率	KMNC 作为一种划分为 K 节的细粒度覆盖度量指标，能够生成更全面的反馈信息，进而更有效地优化和引导变异学习策略的实施
在 SBNC 和 NBC 上初始覆盖率比较低	SBNC 与 NBC 方法的研究重点在于边界区域的神经元活动，特别是那些激活值显著高于主边界阈值的神经元。神经元在主要功能区的激活值超出阈值的现象不多，导致该方法的整体覆盖率偏低

4.3 实验结论

本文方法与其他两种方法相比，NC 指标的提升并不显著，最终覆盖率和 Tensorfuzz[21] 与 DeepTest[22] 都是 88%左右，这可能是因为 NC 指标作为一个全局性的衡量标准，具有更强的泛化能力，因此本文方法可能缺乏针对性。在边界处理方面，针对 SBNC/NBC 这类关注边界神经元的指标，通过特定变异策略克服了初始覆盖率低的局限。本文方法在 KMNC 指标上的表现均高于 Tensorfuzz

和 DeepTest 方法 10%～20%，这表明该方法更适合处理如 KMNC 这类分段式全域指标。SNAC 由于异常点概率较小，覆盖率一直较低，如在 LeNet5 模型上覆盖率都在 10%以下，NBC 因神经元值超出训练集边界的情况不普遍，覆盖率也低。NC 和 TKNC 指标在 LeNet4 模型上获得了较大覆盖率的提升，分别上升了 80.72%及 65.8%。KMNC 指标在 LeNet5 模型上表现较佳，上升了 14.42%。实验表明，尽管具体效果因覆盖指标和模型而异，本文方法在不同模型和数据集上覆盖率普遍提高。这些实验结果充分证明了，本文方法在不同架构的深度学习模型上具有更好的泛化能力和测试有效性。

神经元覆盖指标在 DNN 测试中尤为重要，但其与模型准确性的关系复杂。本实验通过提升覆盖率引导模糊测试，结果显示，尽管覆盖率有所提高，但分类准确率降低。这说明高覆盖率不等于高准确率。后续研究将比较关键和非关键神经元的健壮性，提供开发和测试建议。覆盖指标的对比如表 4 所列。

神经元覆盖率的增加意味着测试范围更广，而更广泛的测试理论上能产生更有效的测试用例，使用这些测试用例对模型进行重训练，可以提高模型的准确率。

表 4 覆盖指标对比

Table 4 Comparison of evaluation indicator methods

覆盖指标	名称	测试数据集	优点	缺点	适用场景
NC	被激活的神经元比例	随机采样数据集	计算简单	只关注是否激活，覆盖面窄	快速预判测试集合是否有效
KMNC	K 节神经元覆盖比例	不同区间的数据集	能区分不同激活区间的覆盖	仅考虑某个区间的神经元	分析测试集在不同激活区间的覆盖效果
NBC	神经元边界覆盖比例	边界样本的数据	聚焦边界情况，反映角落覆盖	仅考虑边界区	分析测试集是否覆盖边界情况
SNAC	强神经元激活覆盖比例	高激活样本的数据	专注覆盖上边界神经元状况	仅考虑上边界，覆盖面窄	分析测试集是否激活高响应区域
TKNC	top-k 高活跃神经元的比例	全量测试集	反映重要神经元的覆盖	k 的取值较难确定	判断重要神经元是否被覆盖
BKNC	top-k 低活性神经元的比例	全量测试集	反映非重要神经元的覆盖	k 的取值较难确定	判断非重要神经元是否被覆盖

4.4 重训练结果分析

针对模型鲁棒性差的问题，对模型进行重训练，以提高准确率。本文使用新生成的测试用例对模型进行重训练，记录重训练后模型的准确率。在重训练之后，准确率均能得到稳步提升，因此重训练是提高模型鲁棒性的有效手段。

重训练结果如表 5 所列。

表 5 重训练后的相对准确率

Table 5 Relative accuracy after retraining

模型指标	LeNet4			LeNet5		
	初始准确率/%	新生成的测试用例个数	重训练后的准确率/%	初始准确率/%	新生成的测试用例个数	重训练后的准确率/%
KMNC	98.87	16759	99.90	98.97	13266	99.94
NBC	98.87	10558	99.97	98.96	10715	99.98
SNAC	98.87	10565	99.98	98.97	10668	99.99
TKNC	96.45	10023	99.97	98.96	10094	100.00
BKNC	98.87	10029	99.98	98.97	10066	100.00
NC	98.80	10014	99.02	98.97	10013	100.00

通过采用多指标引导生成的多元化测试用例来丰富和完善重训练的微调数据集，不仅可以提升模型在特定任务上的表现，还能增强模型对新任务的适应能力和整体的鲁棒性。这种方法的关键在于通过覆盖多种行为模式和评估标准，确保模型在不同方面都达到最佳性能，从而实现一个全面优化且高效的深度学习模型。

结束语 以往相关研究表明，随着网络加深、加宽，传统方法在提高覆盖率时，往往忽视了测试用例分类准确率，导致测试用例质量较差[23]。通过使用本文生成的高质量测试用例重训练模型，提升了模型的分类准确率，也增强了模型的鲁棒性。然而，神经元覆盖指标与分类准确性之间的直接关系尚不明确，不同指标在不同模型上表现出的泛化能力不一致。提高测试用例覆盖率未必能提升准确率，受限于冗余特征和不合适的激活模式等因素，通过组合测试技术、多指标综合考量、自动生成蜕变关系等方法有助于提升测试覆盖率和模型鲁棒性。本文方法在处理大规模数据集和复杂任务时可能存在性能瓶颈，应引入更先进的优化算法，并增强模型的泛化能力。未来研究应聚焦于开发更高效的变异策略和鲁棒性评估框架，以降低计算复杂度并提高泛化能力[24]，同时，应进一步探索不同覆盖指标与模型性能之间的具体联系。

参 考 文 献

[1] YANG Z, SHI J, ASYROFI M H, et al. Revisiting neuron coverage metrics and quality of deep neural networks[C]// 2022 IEEE International Conference on Software Analysis, Evolution and Engineering(SANER). 2022: 408-419.

[2] XIE X, LI T, WANG J, et al. NPC: Neuron path coverage via characterizing decision logic of deep neural networks [J]. ACM Transactions on Software Engineering and Methodology, 2022, 31(3): 1-27.

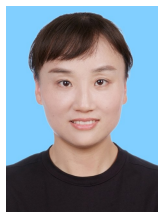
[3] AGHABABAEYAN Z, ABDELLATIF M, BRIAND L, et al. Black-box testing of deep neural networks through test case diversity [J]. IEEE Transactions on Software Engineering, 2023, 49(5): 3182-3204.

[4] FAHMY H, PASTORE F, BRIAND L. HUDD: A tool to debug DNNs for safety analysis [C]// Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings. 2022: 100-104.

[5] PEI K, CAO Y, YANG J, et al. DeepXplore: automated white

box testing of deep learning systems [C]// 26th Symposium on Operating Systems Principles. 2017: 1-18.
[6] XIE X, MA L, JUEFEIXU F, et al. DeepHunter: a coverage guided fuzz testing framework for deep neural networks [C]// 28th ACM SIGSOFT Inter-national Symposium on Software Testing and Analysis. 2019: 146-157.
[7] MA L, XU J F, ZHANG F Y, et al. DeepGauge: multi granularity testing criteria for deep learning systems [C]// 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018: 120-131.
[8] DU X, XIE X, LI Y, et al. Deepcruiser: Automated guided testing for stateful deep learning systems[J]. arXiv: 1812. 05339, 2018.
[9] YI Z B, LI S S, MA J, et al. Towards an Efficient and Robust Adversarial Attack Against Neural Text Classifier[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2022, 36(11): 2253007.
[10] MA L, XU J F, XUE M H, et al. Deepct: Tomographic combinatorial testing for deep learning systems[C]// 2019 IEEE 26th International Conference on Software Analysis, Evolution and Engineering(SANER). IEEE, 2019: 614-618.
[11] GUO H, TAO C, HUANG Z. Multi-objective white-box test input selection for deep neural network model enhancement [C]// 2023 IEEE 34th International Symposium on Software Reliability Engineering. 2023: 521-532.
[12] YUAN Y, PANG Q, WANG S. Revisiting neuron coverage for DNN testing: A layer wise and distribution-aware criterion [C]// 2023 IEEE/ACM 45th International Conference on Software Engineering. 2023: 1200-1212.
[13] KANG D. Bridging fuzz testing and metamorphic testing for classification of machine learning [C]// Proceedings of the 30th IEEE International Conference on Consumer Electronics(ICCE 2022). 2022: 1-2.
[14] LI Z, MA X, XU C, et al. Structural coverage criteria for neural networks could be misleading[C]// 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results(ICSE-NIER). IEEE, 2019: 89-92.
[15] WANG L, XIE X, DU X, et al. DistXplore: Distribution-guided testing for evaluating and enhancing deep learning systems [C]// Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2023: 68-80.
[16] XIAO D, LIU Z, YUAN Y, et al. Metamorphic testing of deep learning compilers [C]// Proceedings of the ACM on Measurement and Analysis of Computing Systems. 2022: 1-28.
[17] HU Q, GUO Y, XIE X, et al. Test optimization in DNN testing: A survey [J]. ACM Transactions on Software Engineering and Methodology, 2024, 1(22): 1-41.
[18] ATTAOUI M O, FAHMY H, PASTORE F, et al. DNN explanation for safety analysis: An empirical evaluation of clustering-based approaches [J]. ACM Transactions on Software Engineering and Methodology, 2023, 10(41): 16-57.
[19] KAUR K, SINGH E J. Reducing SSIM(structural similarity index measure) using improved edge detection technique on grey scale images[J]. International Journal for Research in Applied Science and Engineering Technology, 2020, 8(9): 504-508.
[20] YANG Z, SHI J, ASYROFI M H, et al. Revisiting neuron coverage metrics and quality of deep neural networks [C]// 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering. 2022: 408-419.
[21] ODENA A, OLSSON C, Andersen D, et al. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing[C]// International Conference on Machine Learning. PMLR, 2019: 4901-4911.
[22] TIAN Y, PEI K, JANA S, et al. Deeptest: Automated testing of deep-neural-network-driven autonomous cars[C]// Proceedings of the 40th International Conference on Software Engineering. ACM, 2018: 303-314.
[23] YU J, DUAN S, YE X. A white-box testing for deep neural networks based on neuron coverage [J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 34(11): 9185-9197.
[24] WANG Z, YAN M, LIU S, et al. A Review of Deep Neural Network Testing Research [J]. Journal of Software, 2020, 31(5): 1255-1275.

XIAO Ziqi, born in 1999, postgraduate. Her main research interests include intelligent software testing and deep learning model testing.



SHI Yaqing, born in 1981, Ph.D, professor, master's supervisor, is a member of CCF(No. 49805M). Her main research interest is intelligent software testing.

(责任编辑:何杨)