Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

FNN模型正确性测试及测试样本生成

博士研究生 刘洧光

2024年01月21日

问题回溯

- 相关内容
  - 2022.08.23 王若辉《AI测试：历史与发展》
  - 2022.03.12 侯钰斌《神经网络模型的覆盖测试》
  - 2021.11.29 王若辉《深度学习系统安全性测试及测试样本优先级排序》

内容提要

- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - NPC
  - BET
- 特点总结与工作展望
- 参考文献

背景简介

- 预期收获
  - 掌握FNN模型的正确性测试相关知识
  - 了解一种FNN模型的白盒测试方法
  - 了解一种FNN模型的黑盒测试方法

内涵解析与研究目标

- 题目内涵解析（FNN正确性测试及测试样本生成）
  - FNN：前馈神经网络
  - 正确性：模型的输出是否准确和可靠，是模型最主要的功能性需求
  - 测试样本：用于测试模型缺陷的所有样本，包括自然样本和或对抗样本
  - 测试样本生成：特指生成能发现模型正确性问题的测试样本

- 研究目标
  - 面向人工智能领域FNN模型正确性测试
  - 研究测试预言、测试充分性验证、测试样本生成与排序等关键问题
  - 结合覆盖测试、差分测试、模糊测试与突变测试等理论
  - 发现模型的正确性问题并指导模型的修复与再训练

研究背景

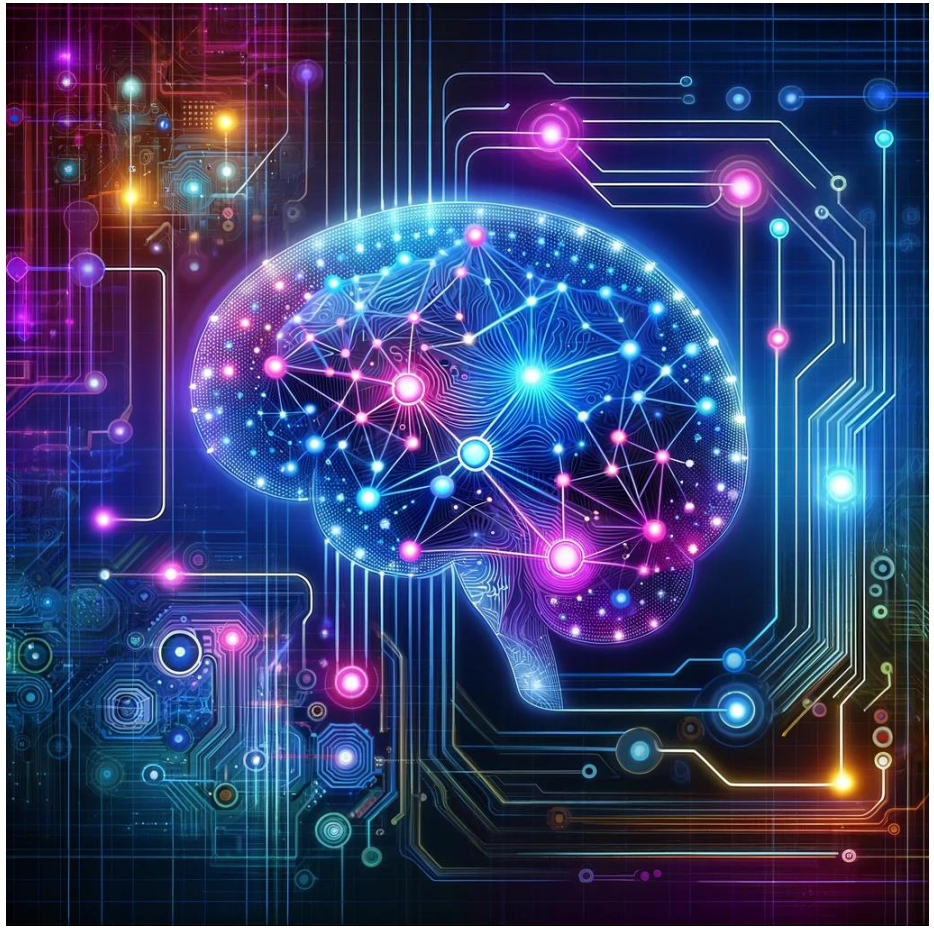
- 研究背景
  - 人工智能技术发展迅速，在图像识别、自然语言处理等领域发挥重要作用
  - FNN模型被应用在医疗诊断、自动驾驶等安全关键领域
  - FNN模型存在过拟合、欠拟合或数据不平衡、数据污染导致的缺陷
  - 这些缺陷将导致人工智能系统在正确性方面出现严重的问题

2022年5月，SpiceJet旗下一架波音737自动驾驶系统失灵，造成17人重伤

2022年8月， 一辆开启了LCC车道居中辅助的小鹏P7在宁波高架行驶过程中与前方一辆检查故障的车辆相撞

研究意义

- 研究意义
  - 发现模型缺陷
    - 测试模型的正确性
    - 及时发现模型的错误决策行为
  - 自动化生成测试样本
    - 高效且自动化生成测试样本，避免手工标记或检查，提升模型正确性测试效率
    - 发现导致模型错误决策的测试样本分布
  - 模型正确性增强
    - 利用测试样本对目标模型进行再训练，以提高目标模型的正确率



发现模型缺陷，自动化生成测试样本，保障模型安全

研究历史

2017

Pei等人首次提出一种针对深度学习系统的白盒测试方法，借鉴了传统软件测试中的覆盖测试和差分测试思想，解决了测试预言问题并生成了测试样本

2018

Ma等人引入突变测试方法：通过突变算子生成突变模型，计算样本在原模型与突变模型上的输出差异，以此来评估样本发现模型缺陷的能力

2020

Lee等人提出一种自适应选择最优神经元的方法，设计了29个神经元特征参数化神经元选择策略，以最大限度地提高NC和TKNC

2022

Wang等人提出了一种黑盒CNN模型测试方法，通过对卷积核添加连续扰动生成发现模型缺陷的测试样本，并且考虑了模型测试的实践场景

2018

Guo等人首次将模糊测试方法引入深度学习系统测试，通过梯度上升算法，最大化模型的神经元覆盖率和模型的错误，生成添加了有限扰动的测试样本

2019

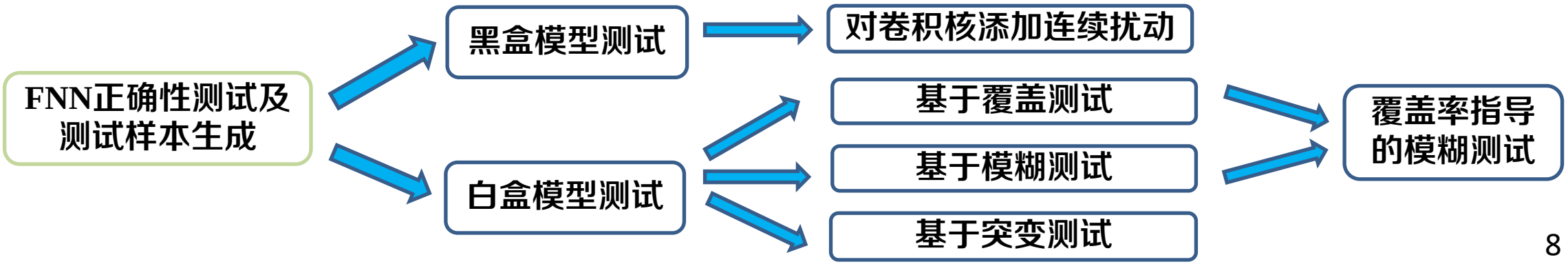
Ma等人在Pei等人提出的神经元覆盖率基础上进行了细化，提出了多种细粒度的神经元覆盖率变体，包括MMNC、TKNC、SNAC和NBC

2022

Xie等人为解决传统神经元覆盖率可解释性不足的问题，从DNN模型中提取决策流图，并提出了基于结构的神经元路径覆盖率和基于激活的神经元路径覆盖率

2023

Jin等人对目前的神经元覆盖准则提出了质疑，通过合作博弈论选择可兴奋的神经元，用PSO优化算法激活更多的兴奋神经元



知识基础　两个测试问题

- 测试预言问题
  - 基本概念：测试预言（Test Oracle）是测试结果的参照物，用于验证目标模型的实际输出是否符合预期
  - 难点：FNN模型通过训练样本学习样本模式，利用神经元权重和非线性激活函数来表征其行为，使其决策逻辑难以被预测
- 测试充分性问题
  - 基本概念：测试充分性是评判现有测试是否具有良好的故障检测性能，对现有测试活动提供的客观置信度度量
  - 难点：传统软件工程的代码覆盖率无法直接应用于深度学习模型，需要提出新的覆盖率指标




目前这两个问题已经基本解决

知识基础　四种测试方法

- 变形测试（解决测试预言问题）
  - 构造一组等式/不等式关系
    - 例：对一张图片进行翻转、仿射、更改对比度，模型前后的预测结果不变
    - 例：在训练数据中注入噪声，模型的正确率下降了

- 差分测试（解决测试预言问题）
  - 使用多个功能相似的模型预测同一样本，对比不同模型之间的输出差异

训练多个模型开销过大

  - 对同一样本添加肉眼不可区分的微小扰动，对比同一模型前后的输出差异

知识基础　四种测试方法

- 覆盖测试（解决测试充分性问题）
  - 使用神经元覆盖率以及各种细粒度变体（见附录）衡量模型的测试充分性
    - 例：传统神经元覆盖率（Neuron Coverage，NC）

$$NC(T,x)=\frac{|(n|\forall x\in T,out(n,x)>t)|}{|N|}$$

    - 其中，$|N|$代表模型的神经元总数，$out(n,x)$是神经元n对于某一测试样本x的输出值，t是预设的神经元激活阈值，T是测试样本全集
- 突变测试（解决测试充分性问题）
  - 人为引入一系列突变规则，制造出一些被称为“突变体”的突变模型/数据
    - 例：随机删除模型的一个隐藏层
    - 例：随机错误标注一部分训练数据
  - 评估测试样本集合的有效性，以及测试人员和工具对模型中潜在缺陷的发现能力

以覆盖率为指导的白盒模糊测试成为主流方法

知识基础　模型正确性测试与对抗样本攻击的区别

- 模型正确性测试
  - 目的：发现一个目标模型的更多缺陷
    - 白盒方法试图实现高神经元覆盖率
    - 黑盒方法通过多样化发现的错误诱导输入的标签来探索不同的决策边界
  - 使用训练数据生成的测试样本对模型微调可以提高正确率
- 对抗样本攻击
  - 目的：成功地制作一个样本落入特定或任意分类结果
    - 对抗样本也可以算是一种测试样本
    - 目前主流的测试样本生成方法和对抗样本很相似
  - 使用训练数据生成的对抗样本对模型微调会降低正确率，提高鲁棒性

两者目的不同，效果不同，但方法相似

研究现状

- 基于覆盖率指导的白盒模糊测试方法
  - 已经证明，神经元覆盖率的提高并不能发现模型的更多缺陷
  - 神经元的激活阈值基于经验设定，导致测试的不确定性
  - 在维护种子列表时需要反复更新神经元覆盖率，开销巨大
- 测试样本生成方法
  - 测试样本通过计算梯度生成，更像是对抗样本的生成方法
  - 生成的测试样本与极端自然样本之间特征分布差距明显
  - 测试样本生成的效率较低，数量较少，无法很好地扩充现有的测试集

需要更科学的方法，生成更自然的样本

car


face

truck

【TOSEM】

NPC: Neuron Path Coverage via Characterizing Decision Logic of Deep Neural Networks

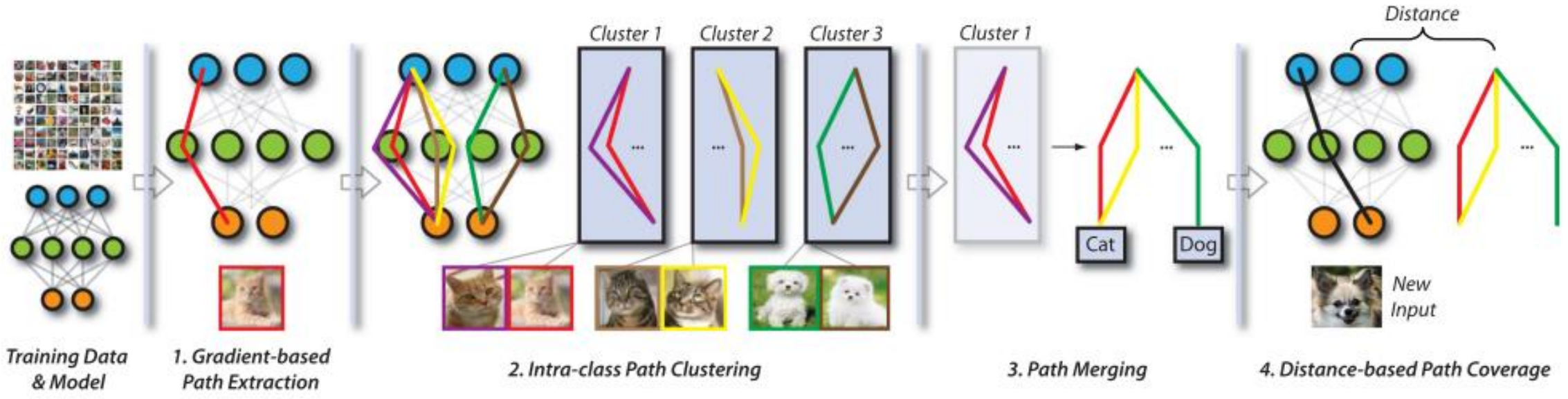
NPC TIPO

T	目标	提出具有可解释性的DNN结构覆盖标准
I	输入	待测模型*1个，训练集数据全集
P	处理	1. 路径提取：基于LPR提取CDP 2. 路径抽象：类内CDP聚类，抽象合并CDP 3. 路径覆盖率：根据测试数据计算SNPC和ANPC
O	输出	抽象决策逻辑CDP*n组，覆盖率指标*2个

P	问题	现有的神经元覆盖指标可解释性低，覆盖率与决策逻辑之间的关系未知
C	条件	前馈神经网络、白盒分类模型
D	难点	1. 如何提出具有较强可解释性的覆盖率指标 2. 构建模型的决策逻辑流图
L	水平	TOSEM 2022（SCI一区）

NPC 算法原理图

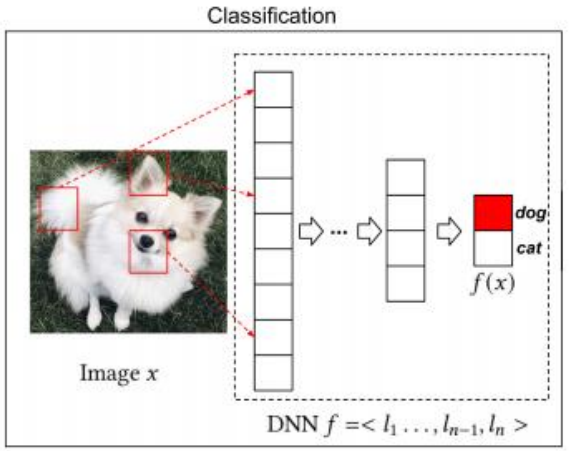
- 算法原理图
  - 步骤1：基于逐层相关性传播原理（LPR）提取关键决策路径（CDP）
  - 步骤2：类内关键决策路径聚类
  - 步骤3：关键决策路径聚类合并抽象
  - 步骤4：计算基于控制流（SNPC）和数据流（ANPC）的神经元路径覆盖率



- 逐层相关性传播（Layer-Wise Relevance Propagation，LRP）
  - 算法思想：一种通过计算输入的关键特征来解释决策的有效方法
  - 算法原理
    - 正向计算输出

输入样本　置信度矩阵

$$f(x) = argmax_y O_n$$



    - 反向计算相关性

输入的维度

分类器f对于输入样本x在类y上的输出

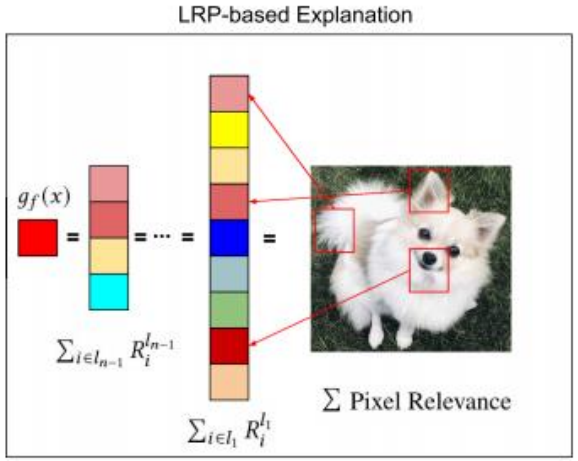
输入层l_1(如图像中的像素)中维度i的相关性

神经元i在第l_{n-1}层的相关性

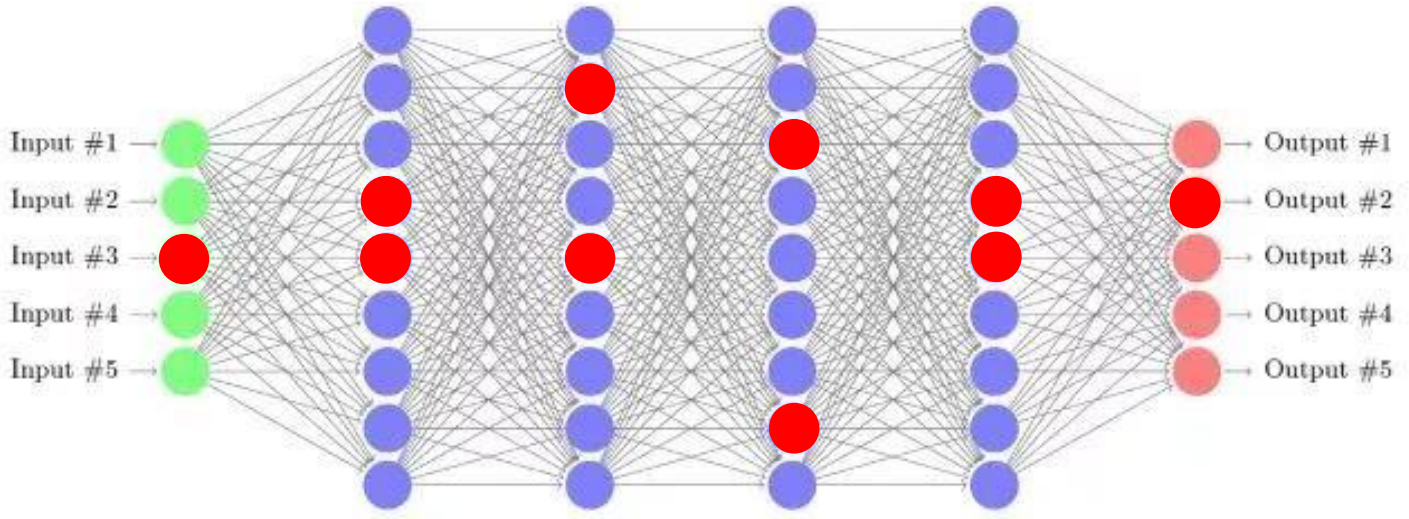
$$g_{f(x)} = \sum_{i=1}^{d} R_i^{l_1}$$

$$g_{f(x)} = R_y^{(l_n)} = \sum_{i \in (l_{n-1})} R_i^{l_{n-1}}$$

$$R_j^{(l_{m+1})} = \sum_{j \in (l_{m+1})} R_{i \leftarrow j}^{(l_m, l_{m+1})}$$



- 关键决策路径构建（a -Critical Decision Path ）
  - 神经元选择表达式：$(\forall j \epsilon s_i . R_j^{l_i} > 0) \cap (\sum_{j \epsilon s_i} R_j^{l_i} > a \cdot g_f(x))$
    - a为每层所选择的神经元百分比个数
    - s_i是第i层上具有最大相关性的一组关键神经元
  - 所选择的神经元构成了关键决策路径：$p =< s_1, \ldots, s_n >$



具有最大关联值的关键神经元

非关键神经元

创新分析　CDP路径抽象

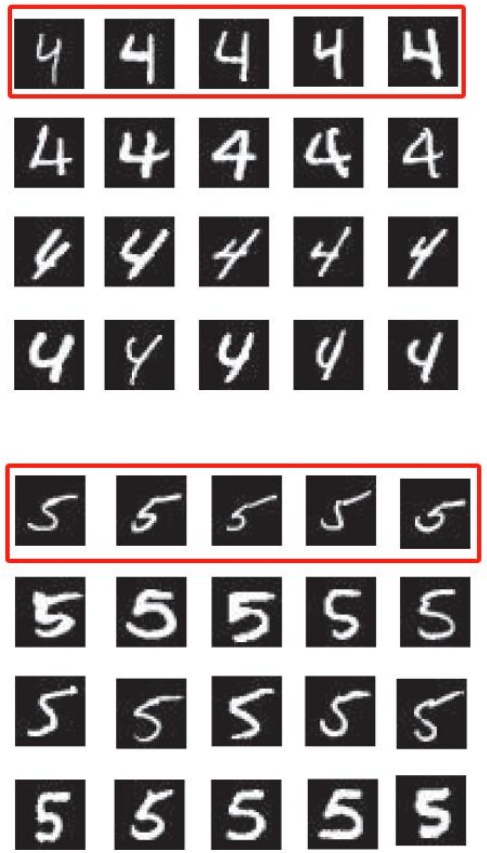
- 抽象关键决策路径（CDP）
  - 原因：计算所有CDP开销巨大；相似类别具有相似的CDP
- 两阶段抽象法
  - 类内路径聚类
    - 同一聚类中的样本具有更相似的决策逻辑
    - K-Means聚类算法
    - 对于每个样本，获得一个表示DNN所有神经元状态的向量
  - 路径合并抽象
    - 计算神经元权值：$w_t = \frac{\left|\left\{x \middle| x \epsilon C_j^y \cap t \epsilon s_i^x\right\}\right|}{\left|C_j^y\right|}$

第j个聚类中的训练样本

输入x的CDP中第i层的关键神经元集合

    - 合并抽象CDP：$\hat{p}_\beta = \langle \hat{s}_{1,\beta}, \cdots, \hat{s}_{n-1,\beta} \rangle$，其中$\hat{s}_{i,\beta} = \{t | t \epsilon \hat{s}_i \cap w_t > \beta\}$

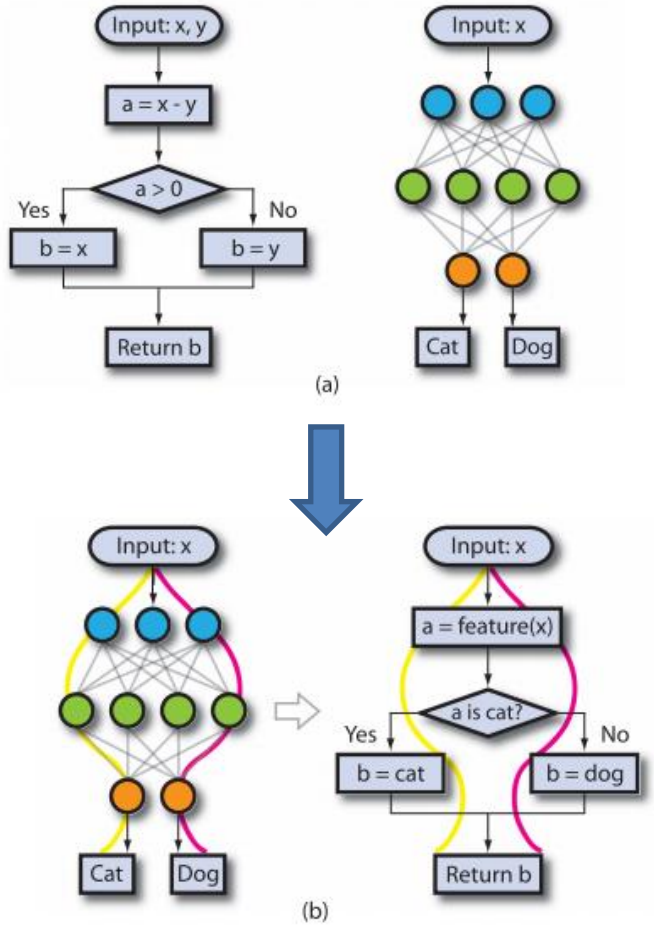
阈值



创新分析　路径覆盖率

- 神经元路径覆盖（Neuron Path Coverage）
  - 基于结构的神经元路径覆盖：SNPC → 主要考虑控制流
  - 基于激活的神经元路径覆盖：ANPC → 主要考虑数据流
  - 从理论上讲，深度神经网络中的路径数量可以是无限的
- SNPC的计算
  - 逐层计算相似度：$J_{p_{x,\hat{p}}^{l}}{}^{l} = \frac{s_l^x \cap \hat{s}_l}{s_l^x \cup \hat{s}_l}$，$p_x$和$\hat{p}$是两条不同的CDP
  - 离散化分为m个覆盖桶：$b_{x,\hat{p}}^{l} = b_i \text{ if } J_{p_{x,\hat{p}}^{l}}{}^{l} \in (\frac{i-1}{m}, \frac{i}{m}]$
  - 覆盖率计算：$SNPC(X) = \frac{\left|\left\{b_{x,\hat{p}}^{l} \middle| \forall x \epsilon X, \forall \hat{p} \epsilon G_{f(x)}, \forall l \epsilon f\right\}\right|}{n \cdot k \cdot |\hat{p}| \cdot m}$

总类数　类$f(x)$的簇数　总层数　桶的总数



创新分析　路径覆盖率

- SNPC采用Jaccard相似度来计算输入x的CDP与抽象CDP之间的结构相似度
  - 覆盖率越高，深度神经网络探索的逻辑就越多样化
  - SNPC只考虑控制流，在某些情况下可能是粗糙的
- ANPC的计算
  - 从聚类簇中选择与输入x最相似的训练样本：$x' = argmax_{x' \epsilon C_j^{f(x)}} J_{p_x, p_{x'}}$
  - x与x'在第l层的距离：$D_{x,x'}^{l} = \left\| A\left(x, \hat{p}^{l}\right) - A\left(x', \hat{p}^{l}\right) \right\|$　抽象路径p的第l层神经元的激活值
  - 离散化分为m个覆盖桶：$d_{x,\hat{p}}^{l} = b_i \text{ if } D_{x,x'}^{l} \in \left(U \cdot \frac{i-1}{m}, U \cdot \frac{i}{m}\right]$
  - 覆盖率计算：$ANPC(X) = \frac{\left|\left\{d_{x,\hat{p}}^{l} \middle| \forall x \epsilon X, \forall \hat{p} \epsilon G_{f(x)}, \forall l \epsilon f\right\}\right|}{n \cdot k \cdot |\hat{p}| \cdot m}$

- 数据与模型资源

数据集名称	MNIST	CIFAR-10	SVHN	ImageNet
样本数量	60,000/10,000	50,000/10,000	73,257/26,032	1,000,000+/50,000
特征维度	28*28*1	32*32*3	32*32*3	244*244*3
标签数量	10	10	10	1000→10
所用模型	SADL-1	SADL-2/VGG16	AlexNet	VGG16
模型准确率	99.1%	90.36%/89.17%	94.31%	76.81%

- 对比方法：最先进的覆盖标准
  - DeepXplore [NC] (2017)
  - DeepGauge [KMNC/NBC] (2018)
  - SADL [LSC/DSC] (2019)
  - DeepImportance [IDC] (2020)

- 评价指标

原始预测结果 屏蔽后预测结果

  - 不一致率（Inc）：$\frac{\{x|x\epsilon X \wedge f(x) \neq f_m(x)\}}{|X|}$
  - 类内/类间相似度
  - 输出公正性：$\frac{\sum_{t\epsilon C_k} P_{t=C_k} \log P_{t=C_k}}{\frac{1}{|c|}\log\frac{1}{|c|}}$

- **RQ1：CDP与DNN决策的关系**

Dataset	Model	$\alpha = 0.7$			$\alpha = 0.8$			$\alpha = 0.9$			$\alpha = 1$		
		Width	Inc.C	Inc.NC	Width	Inc.C	Inc.NC	Width	Inc.C	Inc.NC	Width	Inc.C	Inc.NC
MNIST	SADL-1	9.3	89.3	9.6	**12.8**	**95.8**	**1.9**	17.2	97	0	33.3	98.9	1.8
CIFAR	SADL-2	**20.9**	**100**	**0**	27.2	100	0	36.3	100	0	63.4	100	0
	VGG16	10.2	100	12.1	13.3	100	1.5	**17.9**	**100**	**0.1**	33.8	100	0
SVHN	AlexNet	**23.0**	**100**	**2.1**	29.6	100	1.2	38.2	100	0.2	64.6	100	0
ImageNet	VGG16	**41.1**	**99.3**	**12.5**	51.9	99.8	13.9	65.9	99.9	13.2	98.8	99.8	0.2

α	SADL-1			SADL-2			VGG16(CIFAR-10)			AlexNet			VGG16(ImageNet)		
	Width	Intra	Inter	Width	Intra	Inter	Width	Intra	Inter	Width	Intra	Inter	Width	Intra	Inter
0.7	9.30%	57.7%	34.7%	20.9%	48.6%	23.2%	10.2%	86.5%	22.2%	23.0%	47.8%	30.0%	41.1%	54.9%	32.2%
0.8	12.80%	62.8%	39.1%	27.2%	56.5%	41.8%	13.3%	60.9%	26.0%	29.6%	54.5%	36.5%	51.9%	60.8%	41.6%
0.9	17.20%	67.6%	45.3%	36.3%	64.4%	38.0%	17.9%	67.1%	32.0%	38.2%	63.2%	46.1%	65.9%	70.9%	55.8%
1	33.30%	74.5%	56.3%	63.4%	86.5%	62.0%	33.8%	78.3%	56.1%	64.6%	78.3%	75.4%	98.8%	99.9%	97.9%

- **实验结论**
  - **CDP对模型的决策很重要**
  - **相同模型，不同数据集的CDP也不同**
  - **CDP太宽或太窄都会影响模型决策**

Dataset	Model	Inconsistency Rate				
		20%	20%–40%	40%–60%	60%–80%	80%–100%
MNIST	SADL-1	99.13	97.53	92.46	79.73	49.02
CIFAR	SADL-2	9.13	7.65	3.29	2.39	0.58
	VGG16	97.94	84.13	62.42	43.21	22.64
SVHN	AlexNet	99.96	99.45	78.7	42.86	15.95
ImageNet	VGG16	73.22	27.02	15.07	9.91	6.21

- RQ2：路径抽象的有效性

(k, β)	SADL-1			SADL-2			VGG16(CIFAR-10)			AlexNet			VGG16(ImageNet)		
	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC
(1, 0.6)	16.9	93.6	2.3	13.9	99.9	2.4	15.5	99.8	4.3	17.3	99.4	4.9	41.5	99.8	1.7
(1, 0.7)	14.6	89.2	8.4	9.4	99.8	2.4	13.1	99.3	4.3	13.6	99.1	13.9	34.4	99.8	1.7
(1, 0.8)	12.5	78.8	29.9	6.4	99.2	2.4	10.8	98.8	4.3	9.8	96.3	16.3	26.4	99.8	1.7
(1, 0.9)	10.4	73.6	59.8	4.1	90.9	2.4	7.9	100	10.1	5.7	50.9	37.6	17.7	99.8	1.7
(4, 0.6)	16.7	94.5	2.4	14.6	99.9	2.0	15.8	99.9	4.3	**18.5**	**99.5**	**4.4**	54.5	100	2.9
(4, 0.7)	15.4	95	2.5	10.8	99.9	2.0	13.5	99.9	4.8	15.1	99	5.8	**49**	**100**	**1.9**
(4, 0.8)	**14.6**	**94.1**	**3.9**	7.6	99.6	2.0	11.2	99.9	4.8	11.8	96.4	16.2	40.4	100	2.67
(4, 0.9)	12.2	88.1	14.3	4.8	95.3	2.0	8.4	100	9.2	7.6	80	26.7	34.3	100	2.6
(7, 0.6)	16.9	96.3	2.6	14.9	99.9	1.5	15.9	99.8	2.8	18.4	99.4	5.6	64.2	100	4.3
(7, 0.7)	15.9	96.1	3.2	11.2	99.8	1.5	13.7	99.9	3.5	15.2	98.9	6	59.5	100	2.9
(7, 0.8)	14.6	92.5	2.9	7.9	99.7	1.5	11.4	99.9	4.4	12.1	97.6	9.7	51.8	100	3.6
(7, 0.9)	12.5	88.7	9.5	**5.4**	**98.2**	**1.6**	**8.7**	**100**	**4.9**	8	90.5	19	46.7	100	3.1
(10, 0.6)	16.9	95.3	1.7	15.2	99.9	1.1	15.9	99.9	2.0	18.5	99.4	5.2	73.2	100	4.3
(10, 0.7)	16.1	95.1	3.1	11.5	99.8	1.2	13.7	99.9	2.7	15.4	98.9	6.3	70.5	100	4.3
(10, 0.8)	14.6	93.0	4.3	8.3	99.8	1.3	11.5	99.9	3.4	12.4	98.4	8.5	58.3	100	3.9
(10, 0.9)	13.5	89.2	8.8	5.4	98.8	1.3	8.7	99.9	8.8	8.6	92.9	17.7	54.3	100	2.7

k:每个类中的簇数

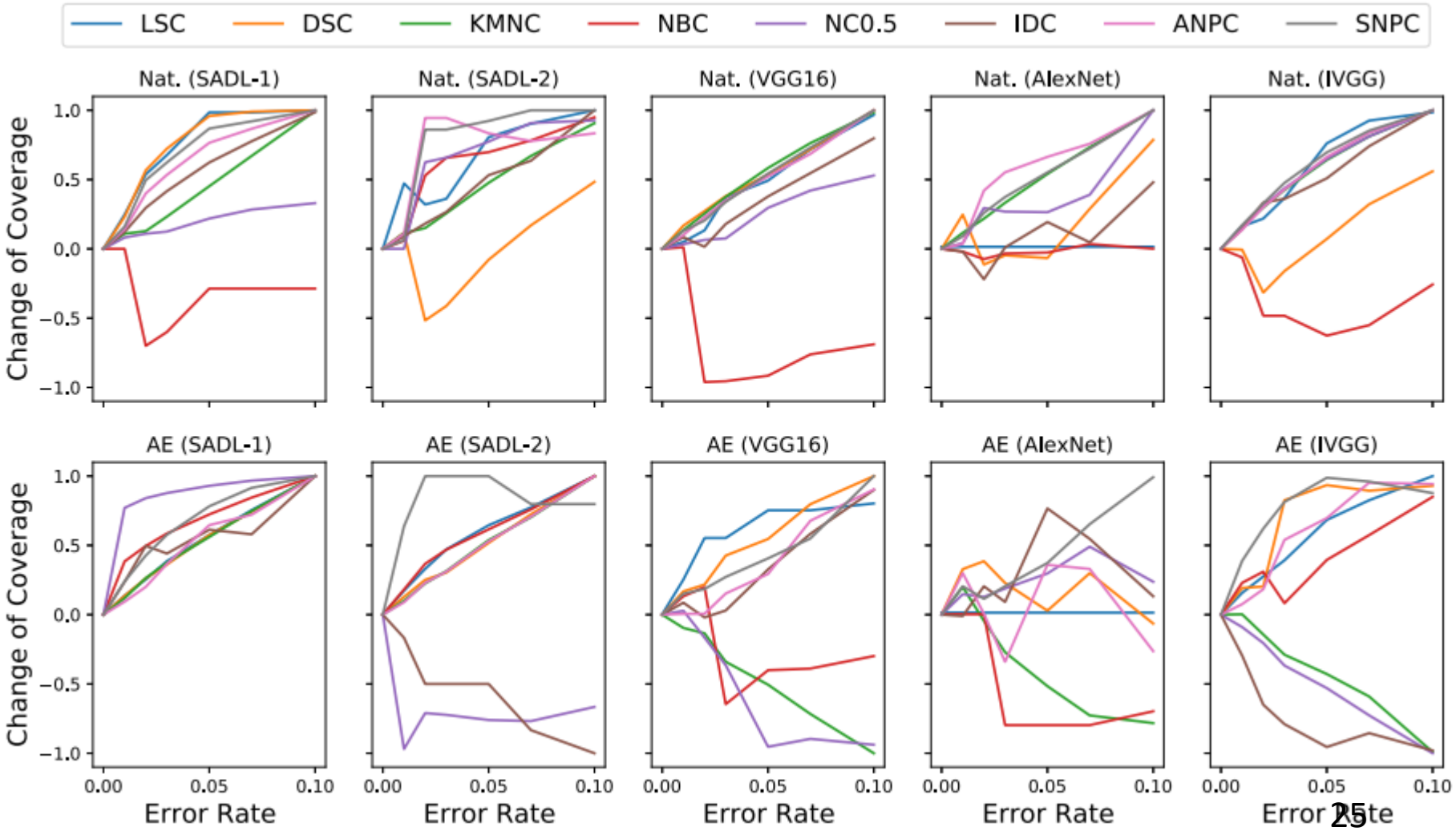
β:神经元选择阈值

- 实验结论
  - 抽象CDP是有效的
  - 在某些情况效果优于具体CDP

Model	#Clus.	Intra_Cla.	Inter_Cla.	Intra_Clus.	Inter_Clus.
SADL-1	4	0.628	0.391	0.708	0.381
SADL-2	7	0.486	0.232	0.524	0.260
VGG16(CIFAR-10)	7	0.671	0.320	0.703	0.316
AlexNet	4	0.478	0.300	0.524	0.298
VGG16(ImageNet)	4	0.549	0.322	0.546	0.331

实验设计　对比实验

- RQ3：缺陷检测的灵敏度
- 实验结论
  - ANPC和SNPC比其他覆盖标准对不同的错误更敏感
  - 不同的覆盖标准可以达到相似的增长趋势，具有一定的相关性



实验设计　对比实验

- RQ4：与输出公正性的相关性（评估覆盖率标准与输出多样性之间的相关性）

		KMNC	NBC	NC(0.0)	NC(0.2)	NC(0.5)	NC(0.75)	LSA	DSA	IDC	ANPC	SNPC
size=100	SADL-1	−0.262	−0.262	−0.300	0.099	0.315	0.001	0.000	−0.638	-0.360	-0.244	**0.723**
	SADL-2	−0.029	−0.029	**0.553**	−0.397	−0.065	−0.111	0.000	0.181	**0.587**	**0.482**	**0.789**
	VGG16	−0.348	−0.348	0.037	−0.031	−0.210	−0.273	0.000	−0.145	0.052	0.012	−0.055
	AlexNet	−0.457	**0.593**	0.000	0.000	0.000	0.000	0.000	−0.320	**0.912**	**0.961**	**0.989**
	Avg.	−0.274	−0.011	0.072	−0.082	0.010	−0.096	0.000	−0.230	**0.298**	**0.601**	**0.612**
size=500	SADL-1	**0.639**	−0.116	0.000	0.000	0.000	0.000	0.000	−0.639	-0.707	**0.340**	**0.870**
	SADL-2	**0.543**	0.041	0.000	0.000	0.000	0.000	0.000	0.330	**0.473**	**0.584**	**0.817**
	VGG16	**0.414**	**0.589**	0.000	0.000	0.000	0.000	0.000	−0.255	**0.210**	**0.141**	**0.821**
	AlexNet	**0.539**	**0.586**	0.000	0.000	0.000	0.000	0.000	−0.170	**0.931**	**0.908**	**0.974**
	Avg.	**0.534**	0.275	0.000	0.000	0.000	0.000	0.000	−0.184	**0.227**	**0.893**	**0.871**

- 实验结论
  - ANPC和SNPC与输出公正性的相关性更强
  - SNPC的相关性强于ANPC
  - 测试用例规模越大，相关性更强

对输出多样性敏感的覆盖标准可以指导生成不同类别的更多样化的测试用例

实验设计　对比实验

- **RQ5：NPC的计算效率**

	ANPC	SNPC	KMNC	NBC	LSC	DSC	NC(0.0)	IDC
SADL-1	10.25(2.60)	8.83(2.60)	3.50	1.92	26.25	59.75	0.67	1.11
SADL-2	39.50(14.45)	47.33(14.45)	26.75	13.67	24.67	166.17	1.33	0.82
VGG16(CIFAR-10)	135.50(35.10)	91.17(35.10)	54.67	28.75	21.00	73.00	2.17	2.81
AlexNet	57.00(13.35)	29.50(13.35)	104.33	53.25	37.08	136.00	1.08	1.21
VGG16(ImageNet)	203.24(59.67)	119.75(59.67)	146.43	88.02	154.27	173.27	30.45	13.21

- **实验分析**
  - **除NC外，所有其他覆盖标准都依赖于训练样本，并需要预处理过程**
  - **预处理只执行一次，且通常离线计算，因此只比较覆盖率的计算时间**
- **实验结论**
  - **ANPC和SNPC采用解释分析，但时间开销并不昂贵**
  - **CDP提取是高效的，它只占覆盖计算总时间开销的一部分**

有效性和效率的权衡

算法总结 NPC

- 算法流程
  - 通过LRP逐层相关性传播提取关键神经元并构建CDP
  - 类内CDP聚类，路径合并抽象
  - 神经元路径覆盖率NPC计算
- 算法优势
  - CDP的构建考虑了训练数据以及类内与类间的区别，具有较好的可解释性
  - NPC考虑了结构流和数据流，并且拥有更高的输出公正性
- 算法不足
  - 仅适用于分类任务，不适用于回归任务
  - 对模型有要求：仅支持前馈神经网络，不支持循环神经网络（RNN等）
  - 对于大规模多分类任务效率低下

【ISSTA】
BET: Black-Box Efficient Testing for Convolutional Neural Networks

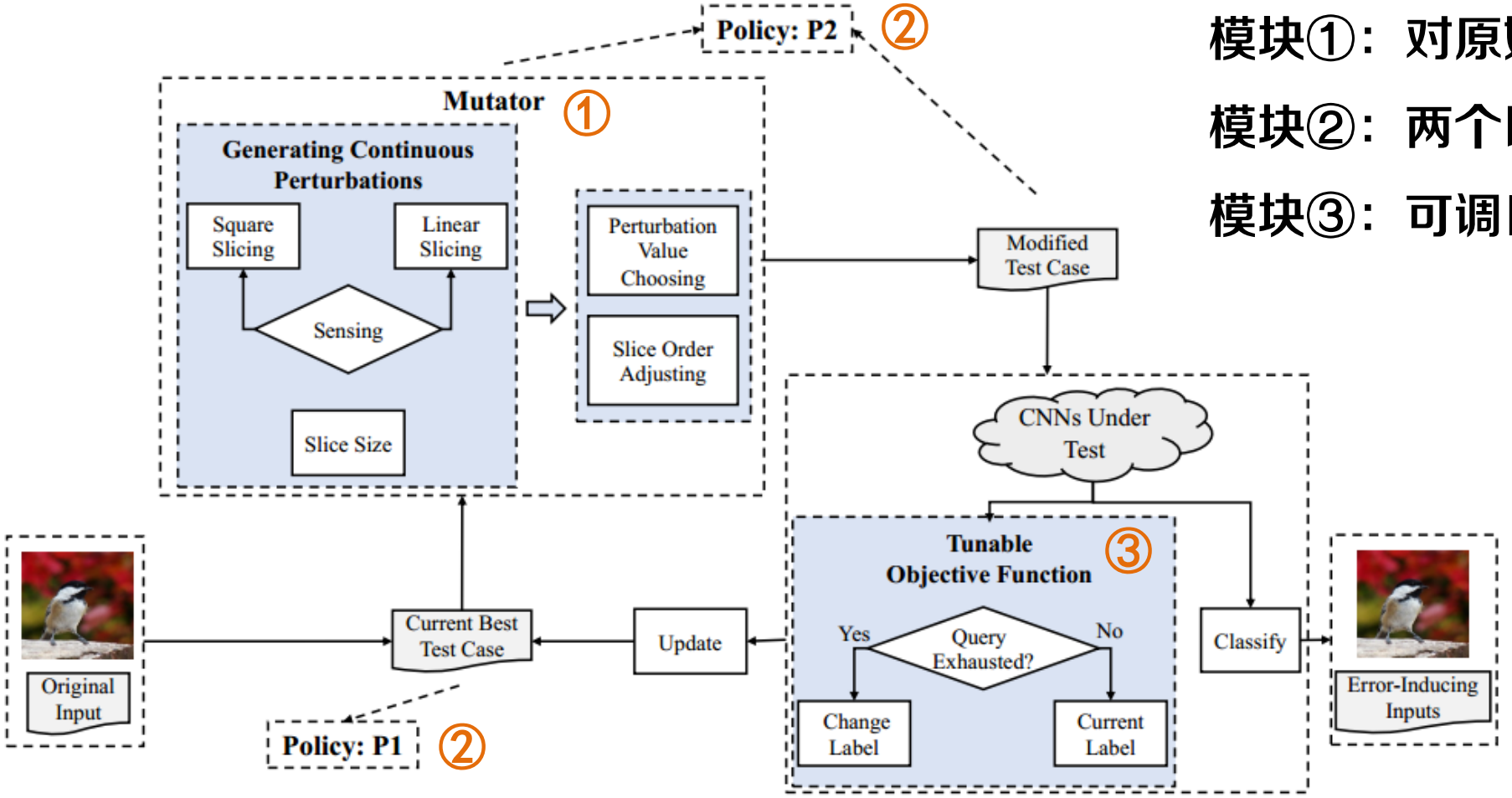
BET TIPO

T	目标	测试黑盒模型并高效地生成诱导性输入
I	输入	待测黑盒CNN模型*1个，训练集数据*n个
P	处理	1.通过以效率为中心的策略选择最佳测试用例 2.对测试用例添加连续扰动，以诱导CNN模型发生错误 3.通过可调目标函数计算得分，并根据得分继续迭代 4.输出能够使模型出错的诱导性输入
O	输出	能够发现模型缺陷的诱导性输入*N个

P	问题	实际应用场景往往无法提供模型的白盒信息 黑盒方法缺乏对模型的内在了解，生成测试用例效率低下
C	条件	CNN黑盒模型
D	难点	1.如何对黑盒模型进行测试 2.如何以高效的查询方式找到导致错误的诱导性输入
L	水平	ISSTA 2022（CCF-A）

BET 算法原理图

- 算法原理图



模块①：对原始测试用例进行变异迭代

模块②：两个以效率为中心的策略

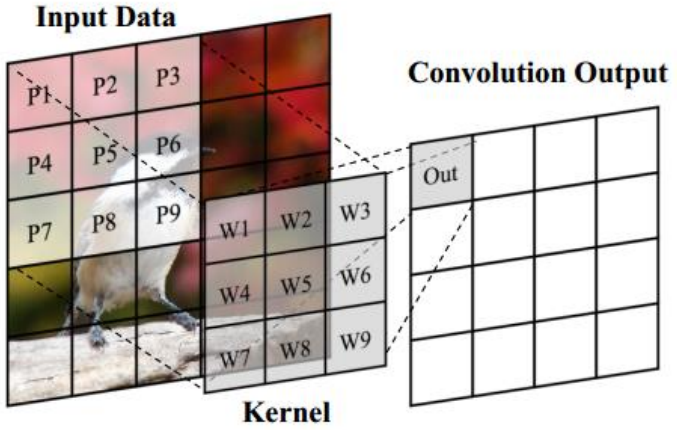
模块③：可调目标函数指导生成

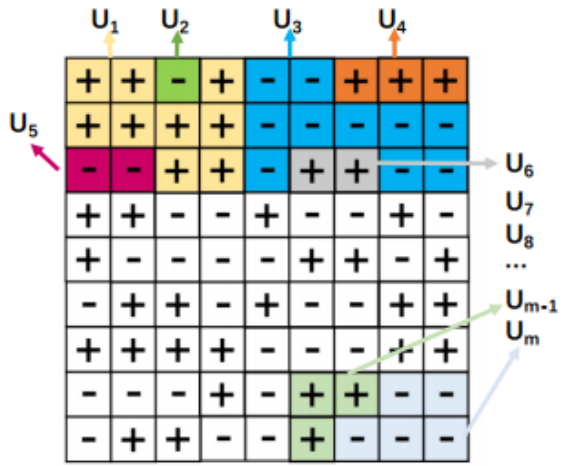
- 必要知识
  - 卷积核的连续扰动：卷积核可以分成多个连续的区域，其中对应的权重具有相同的符号（+或−），在每个区域具有相同符号的扰动称为连续扰动
  - CNN的脆弱性：CNN容易受到连续扰动的影响
  - 卷积运算的输出公式：$\widehat{Out} = W \cdot \hat{P} = \sum_{i=1}^{n}(w_i \times \hat{p}_i)$
  - 最大化模型决策差异：$Dif = |W \cdot \hat{\delta}| = \sum_{i=1}^{n}(w_i \times \hat{\delta}_i)$

区域

扰动

- 注意！！！
  - 黑盒模型不可能完美匹配未知核以产生连续扰动
  - 但仍可以近似匹配卷积核，有效地进行黑盒测试





构建可调目标函数

创新分析　可调模板函数　以效率为中心的策略

- 可调目标函数（支持对标签进行优先级排序）
  - 差分测试（DT）：识别出能够使目标模型与其他模型的预测差异最大化的输入
    - 可调目标函数：$DOF(x) = C(x)[l_i] + \sum_{i=1}^{n} \left| C(x)[l_0] - \widehat{C}_i(x)[l_0] \right|$
    - 其中$C(x)[l_i]$表示对目标模型的全面探索，l_i动态调整以探索模型的不同决策边界
    - $\sum_{i=1}^{n} \left| C(x)[l_0] - \widehat{C}_i(x)[l_0] \right|$旨在找到差分模型与目标模型之间的分歧
  - 单模型测试（ST）
    - 可调目标函数：$SOF(x) = C(x)[l_i]$
- 以效率为中心的策略
  - 规则1：每次迭代中只保留一个当前的最佳测试用例
  - 规则2：每次迭代中只有一个修改过的测试用例会被导出

随机贪婪策略：先随机选择，后贪婪搜索

创新分析　切片并添加连续扰动

- Mutator模块变异迭代
  - 切片大小
    - 在将测试用例分割成片时，需要指定片大小（与扰动大小相同）
    - 在测试了较大的部分之后，继续将测试用例分割成较小的部分
    - 切片大小不会随着当前最佳种子而改变
  - 切片顺序
    - 切片形状近似匹配卷积核的连续区域(SZ)
    - 切片形状先正方形后直线形
    - 优先考虑那些尚未遍历到剩余片的头部的邻居片
  - 添加扰动：对每个字节的扰动限制为$\{+\varepsilon, -\varepsilon\}$

```
while not query budget exhausts do
    curBest = input;
    size = SLICESIZE;
    sCount = len(input) / size;
    slices = slices_ori;
    while size >= STEP do
        // split the test case into multiple slices of the
           same size
        slices = SplitEqually(slices, size);
        size = size / STEP;
        sCount = sCount * STEP;
        curBest, Flag = IterSlices(slices, sCount, curBest);
        if Flag == True then
            break;

Function IterSlices(slices, sCount, curBest):
    for i ∈ range(sCount) do
        modified-test-case = AddPerturbations(curBest, slices[i]);
        isErr, objScore, label = QueryCNNs(modified-test-case);
        if isErr then
            E = E ∪ modified-test-case ;
            D = D ∪ label ;
        if objScore >Tunable-Object-Func(curBest) then
            curBest = modified-test-case;
            Reorder(slices[i+1:]);
        if query budget for current label exhausts then
            Tunable-Object-Func changes label;
            return curBest, True;
    return curBest, False;
```

实验设计　数据资源

- 数据与模型资源

Dataset	Model	8-bits	16-bits
CIFAR-10	VGG16	VGG16-q8	VGG16-q16
	ResNet18	ResNet18-q8	ResNet18-q16
Tiny-ImageNet	VGG16	VGG16-q8	VGG16-q16
	ResNet56	ResNet56-q8	ResNet56-q16
ImageNet	VGG19	VGG19-q8	VGG19-q16
	ResNet50	ResNet50-q8	ResNet50-q16

- 对比方法
  - DiffChaser [黑盒] (2019)
  - DeepXplore [白盒] (2017)
  - DLFuzz [白盒] (2018)
  - Adapt [白盒] (2020)
  - DeepGini [白盒] (2020)
- 评价指标
  - *Err–Num*：使模型发生错误的诱导性输入个数
  - *Label–Num*：每个测试用例平均发现的错误标签
  - *Success rate(SR)*：测试用例能够找到相应的错误诱导输入的百分比
  - *Inact–Rate*：用于评价诱导输入的有效性

实验设计　对比实验

- 对比实验：黑盒差分模型测试的评估

Dataset	Model	Err-Num		Label-Num		SR (%)	
		BET	DiffChaser	BET	DiffChaser	BET	DiffChaser
CIFAR-10	VGG16/VGG16-q8	**10296.2**	1779.4	**5.8**	2.6	**100.0**	100.0
	VGG16/VGG16-q16	**2475.8**	110.0	**4.6**	2.4	**100.0**	100.0
	ResNet18/ResNet18-q8	**7136.2**	6.7	**6.2**	0.8	**100.0**	33.6
	ResNet18/ResNet18-q16	**280.6**	0.2	**2.0**	0.1	**96.5**	1.8
Tiny-ImageNet	VGG16/VGG16-q8	**3153.5**	231.4	**11.4**	1.6	**100.0**	100.0
	VGG16/VGG16-q16	**142.8**	12.8	**2.2**	0.1	**100.0**	34.0
	ResNet56/ResNet56-q8	**860.2**	106.5	**10.2**	1.8	**100.0**	51.6
	ResNet56/ResNet56-q16	**88.4**	3.4	**3.8**	0.6	**100.0**	14.8
ImageNet	VGG19/VGG19-q8	**304.0**	15.6	**31.4**	3.3	**100.0**	43.5
	VGG19/VGG19-q16	**185.2**	0.8	**9.2**	0.7	**96.8**	11.4
	ResNet50/ResNet50-q8	**8611.4**	360.2	**29.4**	4.8	**100.0**	72.4
	ResNet50/ResNet50-q16	**409.8**	48.8	**26.6**	0.6	**94.2**	33.2

- 实验结论
  - BET能够生成更多使模型分类错误的诱导性输入
  - BET能够发现模型的更多错误，探索更多的决策边界
  - BET对每一个测试用例几乎都能够找到诱导性输入

实验设计　对比实验

- 对比实验：黑盒单体模型测试的评估

Table 5: Evaluation of Err-Num in single model testing scenarios.

Dataset	Model	BET	ADAPT	DLFuzz-Best	DLFuzz-RR	DeepXplore
CIFAR-10	VGG16	**16264.4**	13131.4	12143.6	11427.8	10782.6
	ResNet18	**14272.2**	11681.3	11532.4	11738.8	8804.2
Tiny-ImageNet	VGG16	**8843.4**	6546.2	4896.8	4696.8	3542.3
	ResNet56	**9016.2**	8437.6	6867.8	5432.8	4982.8
ImageNet	VGG19	**7059.8**	3154.5	142.7	1031.7	208.6
	ResNet50	**7467.4**	5409.5	524.7	1199.0	2116.0

Table 6: Evaluation of Label-Num in single model testing scenarios.

Dataset	Model	BET	ADAPT	DLFuzz-Best	DLFuzz-RR	DeepXplore
CIFAR-10	VGG16	**9.0**	8.3	4.0	6.9	7.4
	ResNet18	**8.9**	8.9	2.6	7.9	8.1
Tiny-ImageNet	VGG16	**48.2**	41.8	3.4	16.8	12.7
	ResNet56	**49.6**	36.6	2.9	5.8	4.6
ImageNet	VGG19	**49.8**	27.1	1.3	7.5	2.6
	ResNet50	**34.7**	3.8	0.5	1.1	1.4

Table 7: Evaluation of SR (%) in single model testing scenarios.

Dataset	Model	BET	ADAPT	DLFuzz-Best	DLFuzz-RR	DeepXplore
CIFAR-10	VGG16	**100.0**	100.0	96.0	100.0	100.0
	ResNet18	**100.0**	100.0	100.0	100.0	100.0
Tiny-ImageNet	VGG16	**100.0**	93.2	89.3	91.8	90.1
	ResNet56	**100.0**	95.4	91.2	90.3	85.6
ImageNet	VGG19	**100.0**	100.0	52.4	92.4	40.2
	ResNet50	**100.0**	94.0	36.6	54.8	62.4

Table 8: Evaluation of Inact-Rate (%) in single model testing scenarios.

Dataset	Model	BET	ADAPT	DLFuzz-Best	DLFuzz-RR	DeepXplore
CIFAR-10	VGG16	**0.0**	48.4	42.1	49.8	57.9
	ResNet18	**0.0**	22.4	3.8	13.1	18.9
Tiny-ImageNet	VGG16	**0.0**	93.2	89.2	91.9	91.8
	ResNet56	**0.0**	91.7	84.2	88.8	94.5
ImageNet	VGG19	**0.0**	100.0	52.4	92.4	40.2
	ResNet50	**0.0**	94.0	36.6	54.8	62.4

- 实验结论
  - BET能够生成更多使模型分类错误的诱导性输入
  - BET能够发现模型的更多错误，探索更多的决策边界
  - BET对每一个测试用例几乎都能够找到诱导性输入

实验设计　其他实验

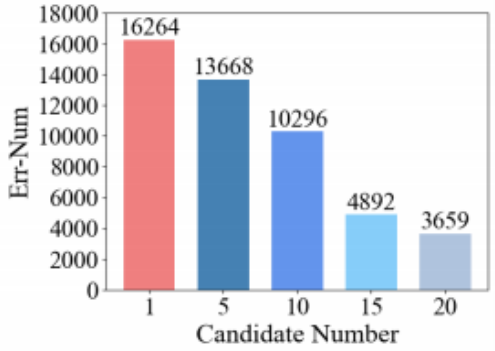
- 通过BET提高CNN的准确性
- 实验设置
  - 仅在CIFAR-10和Tiny-ImageNet上实验
  - 在训练集中各随机混合500个诱导性输入
  - 诱导性输入由训练集中的测试用例生成
  - 对目标CNN进行5次微调
- 实验结论
  - BET产生的误差诱导输入可以将目标模型的准确率提高2 ~ 3%
  - 只使用相同设置下的训练数据集对目标模型进行微调，准确率保持不变甚至下降
  - 使用由测试集生成的诱导性输入微调最高可以提升6%正确率

Dataset	Model	Top-1 Acc (%)	Top-5 Acc (%)
Fine-tune with the error-inducing inputs of BET			
CIFAR-10	VGG16	83.26→85.58	N/A
	ResNet18	85.34→87.03	N/A
Tiny-ImageNet	VGG16	51.37→54.41	75.10→75.74
	ResNet56	46.42→47.88	72.34→72.66
Fine-tune with the corresponding clean training set			
CIFAR-10	VGG16	83.26→83.43	N/A
	ResNet18	85.34→84.78	N/A
Tiny-ImageNet	VGG16	51.37→51.94	75.10→75.21
	ResNet56	46.42→45.48	72.34→72.38

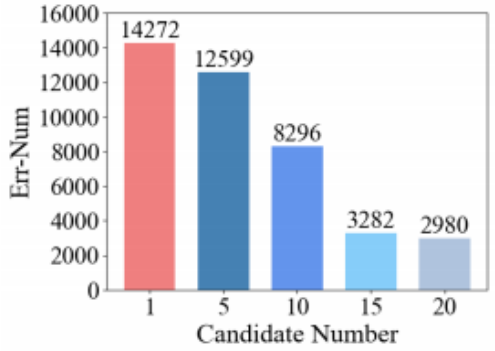
使用对抗样本微调增加鲁棒性，使用诱导性输入微调增加准确率

实验设计　消融实验

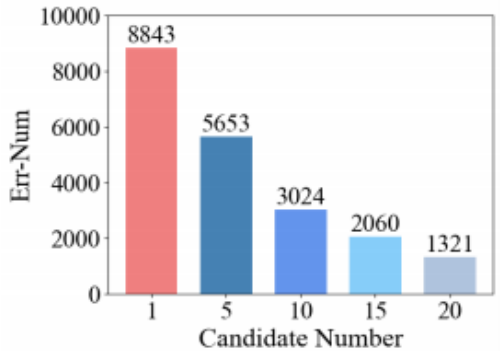
- 消融实验：验证以效率为中心策略的有效性



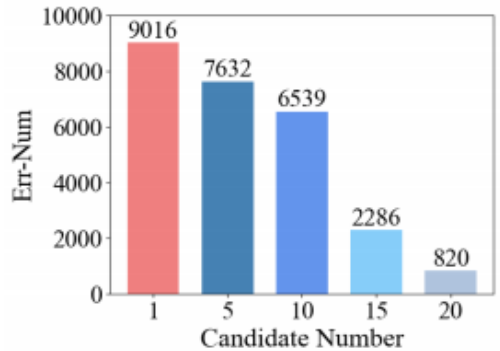
a) CIFAR-10-VGG16



b) CIFAR-10-ResNet18

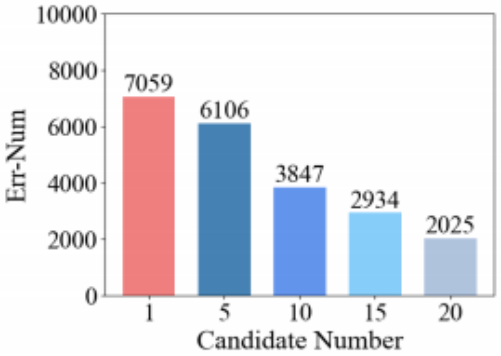


c) Tiny-ImageNet-VGG16

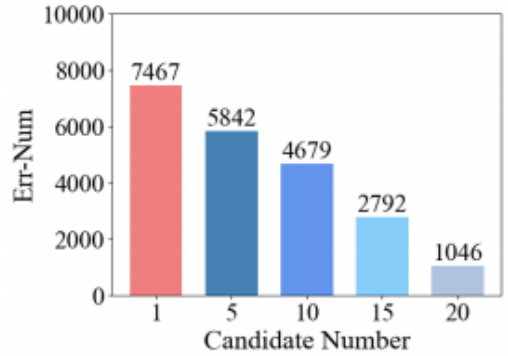


d) Tiny-ImageNet-ResNet56

- 实验结论
  - 以效率为中心的策略(候选数为1)的BET比所有其他比较的候选数执行得更好
  - 在相同的查询预算下，以效率为中心的策略确保BET能够以高效的查询方式找到导致错误的输入



e) ImageNet-VGG19



f) ImageNet-ResNet50

算法总结 BET

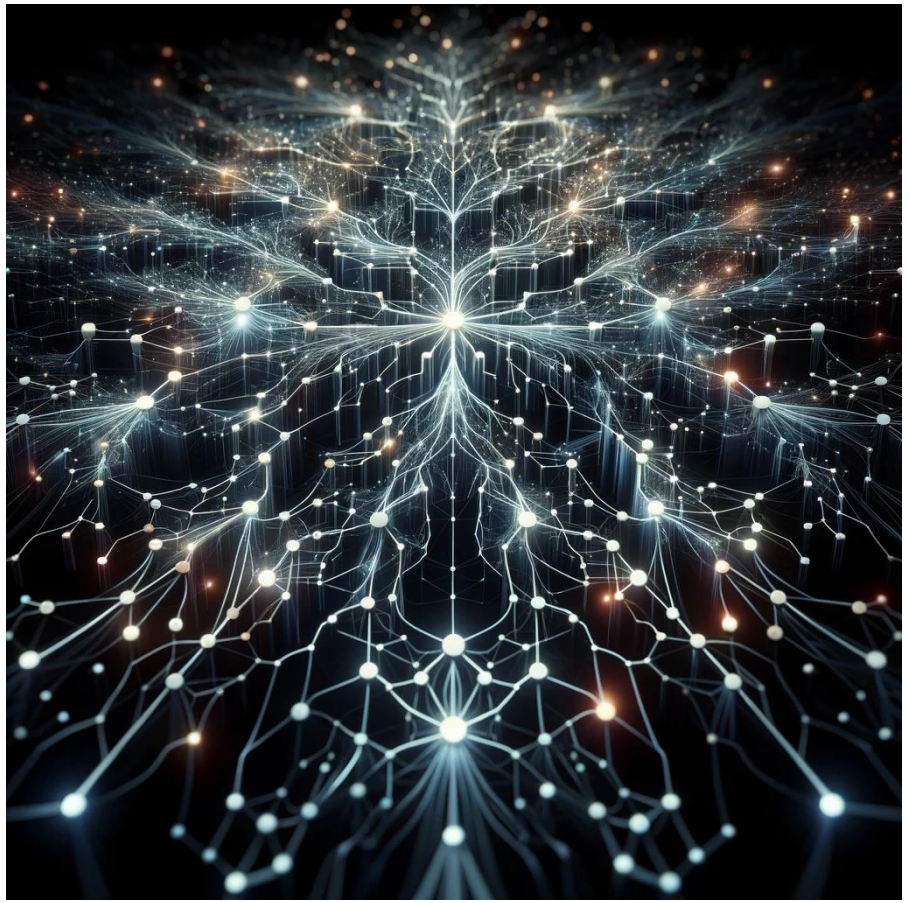
- 算法流程
  - 通过以效率为中心的策略选择最佳测试用例
  - 使用可调目标函数指导测试过程，保存得分较高的扰动
  - 通过添加连续扰动生成测试用例
- 算法优势
  - 适用范围不局限于图像领域，对其他领域的CNN模型都适用
  - 更贴近于实际的黑盒测试场景，即测试人员无法获得模型的（全部）内部信息
  - 同时优于目前的黑盒和白盒方法
- 算法不足（改进方向）
  - 缺乏新的度量来反映黑盒测试的目标模型的内部状态
  - 目前无法在白盒场景中扩展BET的使用
  - 添加扰动时未考虑测试用例的测试预言是否改变

特点总结与未来展望

特点总结与未来展望

- NPC
  - 提出具有可解释性的神经元覆盖率指标
  - 使用训练样本客观地提取模型的决策路径，而非人工基于经验设定阈值
  - 暂未使用所提出的覆盖率指标指导测试样本生成
- BET
  - 考虑黑盒实际测试场景，且效果优于现有的白盒测试方法
  - 忽视了测试预言问题，所生成的测试样本可能更倾向于对抗样本
  - 没有衡量测试充分性的定量指标
- 未来发展
  - 现实的测试场景下多为黑盒，在现有黑盒方法上提出一些定量的指标
  - 使用新的覆盖率指标进行测试样本生成工作
  - 支持测试模型的多样性（特定领域模型、RNN模型、大模型）

- 预期收获
  - 掌握FNN模型的正确性测试相关知识
    - 2个测试问题：测试预言问题和测试充分性问题
    - 4个测试方法：变形、差分、覆盖、突变测试
  - 了解一种FNN模型的白盒测试方法：NPC
    - LPR：逐层相关性传播，选择关键神经元
    - CDP：构建关键神经元路径并抽象路径
    - SNPC：基于控制流的神经元路径覆盖
    - ANPC：基于数据流的神经元路径覆盖
  - 了解一种FNN模型的黑盒测试方法：BET
    - 卷积核的连续扰动思想：CNN的脆弱性
    - 随机贪婪搜索策略：先随机选择，后贪婪搜索



参考文献

[1] Xie X, Li T, Wang J, et al. NPC: N euron P ath C overage via Characterizing Decision Logic of Deep Neural Networks[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2022, 31(3): 1-27.

[2] Wang J, Qiu H, Rong Y, et al. BET: black-box efficient testing for convolutional neural networks. Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis[C]. New York, NY, United States: ACM, 2022: 164-175.

[3] Pei K, Cao Y, Yang J, et al. DeepXplore: Automated Whitebox testing of deep learning systems. Proceedings of the 26th Symposium on Operating Systems Principles[C]. New York, NY: ACM, 2017: 1-18.

[4] Ma L, Juefei-Xu F, Zhang F, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering[C]. New York, NY: ACM, 2018: 120-131.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！

附录　细粒度神经元覆盖率

- k节神经元覆盖率（KMNC）
  - 计算公式

$$KMNC(k) = \frac{|\{S_i^n | \exists x \in T: \phi(x,n) \in S_i^n\}|}{k}$$

  - 公式说明
    - 神经元的输出位于区间$[low, high]$，将区间$[low, high]$分成相等的k个区段
    - 当深度学习系统运行测试集T中的一个测试样本x时，神经元的输出$\phi(x,n)$位于一个区段S_i中，则区段S_i被覆盖

附录　细粒度神经元覆盖率

- $Top-k$神经元覆盖率（TKNC）
  - 计算公式

$$TKNC(k)=\frac{\left|U_{x\in T}\left(U_{1\le i\le l}top_k(x,i)\right)\right|}{N}$$

  - 公式说明
    - 对于给定的测试输入x和同一层上的神经元n_1和n_2，如果神经元输出值$\phi(x,n_1)>\phi(x,n_2)$，则表示n_1比n_2更活跃。$top_k(x,i)$表示给定一个x,第i层中值最大的k个神经元。
    - $TKNC(k)$测量了每一层中曾经是最活跃的k个神经元的数量，定义为每层top_k神经元总数与模型中神经元总数之比

附录　细粒度神经元覆盖率

- 神经元边界覆盖率（NBC）
  - 计算公式

$$NBC(k) = \frac{\left|\sum_{i=1}^{N}(U_i + L_i)\right|}{2N}$$

  - 公式说明
    - 不同于$KMNC(k)$，$NBC(k)$的目标是覆盖边界，即$(low - k\sigma, low]$和$(high, high + k\sigma]$
    - 其中σ是在训练过程中的神经元输出的标准偏差，k是用户定义的参数。设共有N个测试样本，U_i和L_i分别表示样本i的输出落在上边界和下边界

附录　细粒度神经元覆盖率

- 强神经元激活覆盖率（SNAC）
  - 计算公式

$$SNAC(k) = \frac{\left|\sum_{i=1}^{N} U_i\right|}{N}$$

  - 公式说明
    - $SNAC(k)$是$NBC(k)$的一个特例，只考虑了上边界的情况
    - 其中σ是在训练过程中的神经元输出的标准偏差，k是用户定义的参数。设共有N个测试样本，U_i和L_i分别表示样本i的输出落在上边界和下边界