Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

面向深度学习软件库的动态漏洞挖掘方法

网络安全

硕士研究生　刘力源

2022年7月3日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 1. 了解深度学习软件库作用和功能
  - 2. 了解深度学习软件库缺陷或漏洞现状
  - 3. 了解最新的深度学习软件库漏洞挖掘方法

背景简介

- 深度学习软件库/框架/组件
  - 框架：集成完整深度学习开发功能的平台
    - TensorFlow、Theano、CNTK、Mxnet、Pytorch
  - 组件&软件库：各个功能模块
    - 框架内各功能模块、pandas、numpy

背景简介

- 深度学习系统典型架构

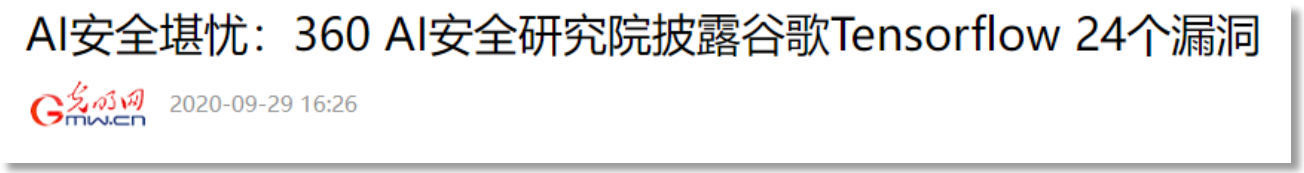
背景简介

- 深度学习框架/软件库本质
  - 高级编程语言开发的代码，意味着存在缺陷的可能
- 深度学习系统安全
  - 绝大多数研究着眼于模型与数据安全
    - 对抗样本
    - 后门攻击
- 当前AI框架缺乏严格的安全测试

Tensorflow 被爆新增71个漏洞 360 AI安全实验室再次立功

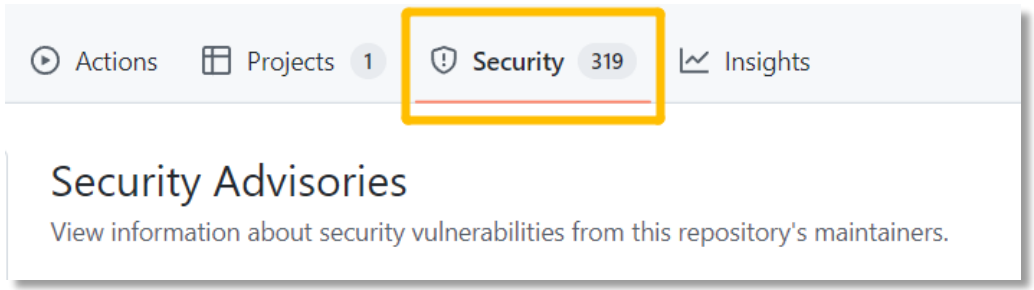
证券市场红周刊
2021-09-23 16:12 | 鲲鹏计划获奖作者,证券市场红周刊官方帐号
关注

近日，三六零(601360.SH，下称“360”) AI安全实验室对谷歌Tensorflow进行安全评测，共新发现71个漏洞，目前均已被谷歌确认并修复。据悉，在不到一年的时间里，360 AI安全实验室已经累计发现并帮助谷歌修复Tensorflow漏洞（CVE）98个，其中高危、严重漏洞共24个，在数量上均位列全球首位。



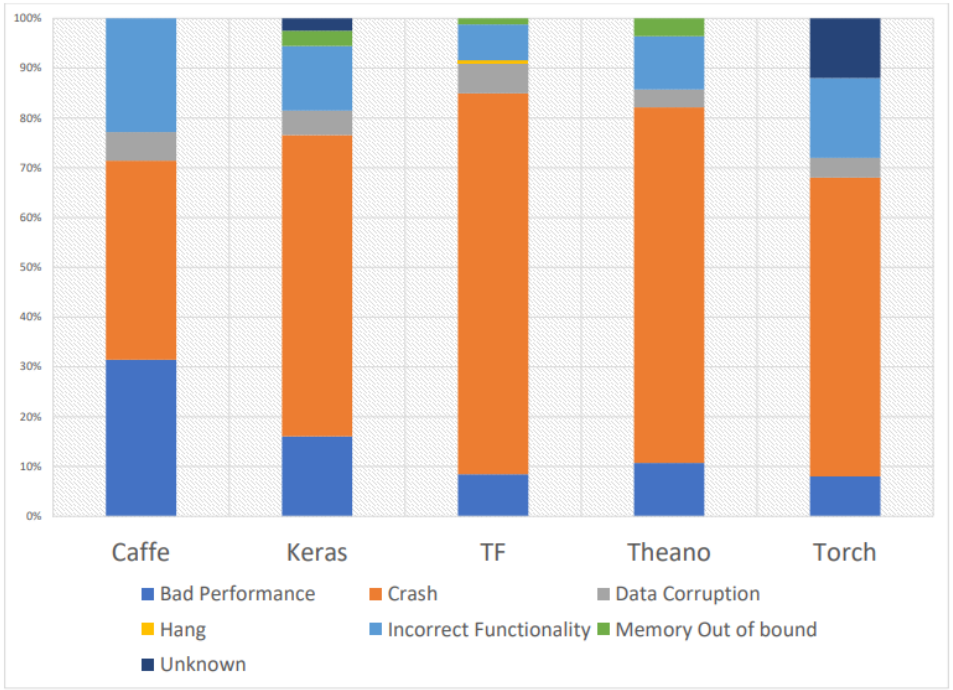
AI安全堪忧：360 AI安全研究院披露谷歌Tensorflow 24个漏洞

光明网 GMW.CN 2020-09-29 16:26



背景简介

- 深度学习软件库缺陷或漏洞现状
  - 缺陷根本原因：代码缺陷
  - 缺陷影响（2019）
    - 性能不佳（Bad Performance）
    - 程序崩溃（Crash）
    - 数据损坏（Data Corruption）
    - 挂起（Hang）
    - 错误功能（ Incorrect Functionality）
    - 内存资源耗尽（Memory Out of Bound）
  - 安全威胁
    - 漏洞利用，构造溢出数据实施攻击，拒绝服务、控制系统
    - 模型决策错误，图像识别、语音识别



基本概念

- 漏洞示例
  - 漏洞影响：程序非正常退出
  - 漏洞分析
    - 输入特征：包含较大的负数值的张量
    - 缺少对输入参数input的安全检查，导致底层库“拒绝服务”

```
import tensorflow as tf

input = tf.constant(-3.5e+35, shape=[10,19,22], dtype=tf.float32)
block_shape = tf.constant(-1879048192, shape=[2], dtype=tf.int64)
paddings = tf.constant(0, shape=[2,2], dtype=tf.int32)
tf.raw_ops.SpaceToBatchND(input=input, block_shape=block_shape, paddings=paddings)
```

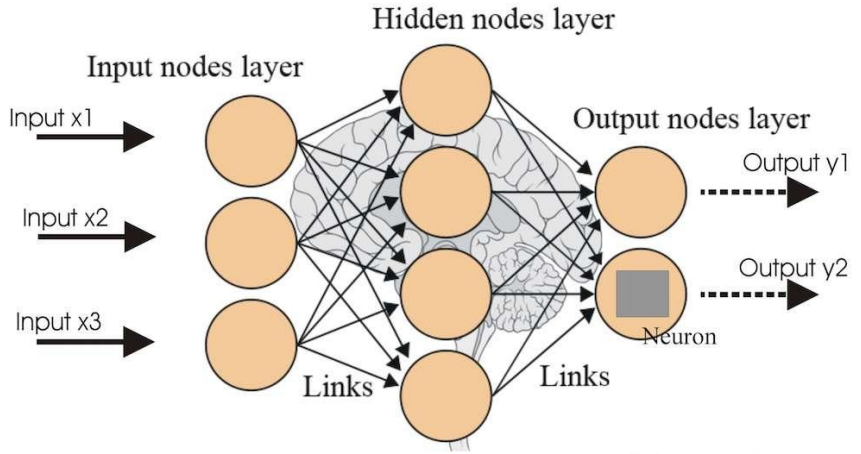
```
2022-07-02 18:33:43.451139: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the foll
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-07-02 18:33:44.375705: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 3969 MB memory:  -> device: 0,
2022-07-02 18:33:44.579817: F tensorflow/core/framework/tensor_shape.cc:396] Check failed: size >= 0 (-1585267068834414592 vs. 0)

Process finished with exit code -1073740791 (0xC0000409)
```

基本概念

基本概念

- 漏洞与漏洞挖掘
  - 漏洞：系统设计、实现或操作和管理中的缺陷或弱点
  - 静态挖掘：分析程序的词法、语义等，结合数据流、控制流信息
  - 动态挖掘：执行待测对象，模糊测试、符号执行
    - 核心问题：构建测试输入，调用功能
- 应用程序编程接口（Application Programming Interface，API）
  - 实现各类子功能得到期望的输出
  - 深度学习软件库调用各API来实现子功能
    - 数据预处理
    - 模型搭建
    - 模型训练
    - ……



基本概念

- 动态测试深度学习软件库
  - 构建API输入
    - 构造能够调用函数、类的输入
    - 局限性：输入符合指定编程语言（如Python）的语法规范

```
tf.raw_ops.Conv2D(
    input, filter, strides, padding, use_cudnn_on_gpu=True, explicit_paddings=[],
    data_format='NHWC', dilations=[1, 1, 1, 1], name=None
)
```

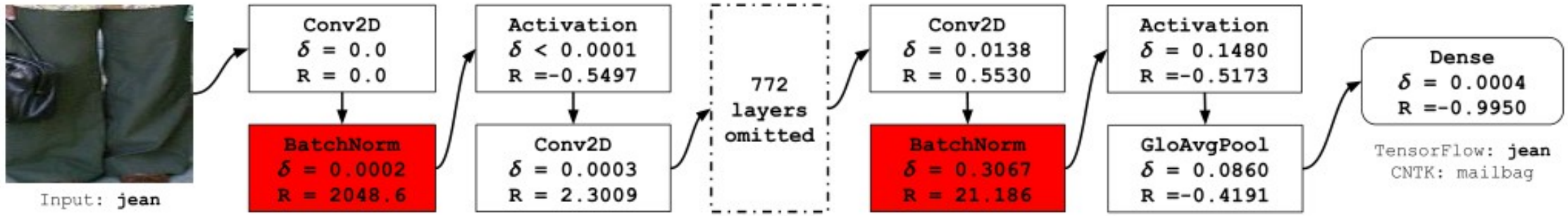
  - 构建模型
    - 模型测试/应用、模型训练，以调用底层库
    - 局限性：如何确定测试预言；缺陷定位与分析

$$x^4 + x^3 + x^2 + 1 = ?$$

深度学习复杂算法更加复杂

基本概念

- 差分测试（CRADLE 2019）
  - 增加测试预言
    - 比较同一个模型在框架A和框架B上输出的置信度差异
  - 局部化缺陷位置
    - 模型运行过程中记录每一层过高的误差变化率
  - 局限性
    - 30个输入模型，且现有公开模型经过多次测试和调整
    - 极小的差异也可能由框架缺陷引起



算法原理

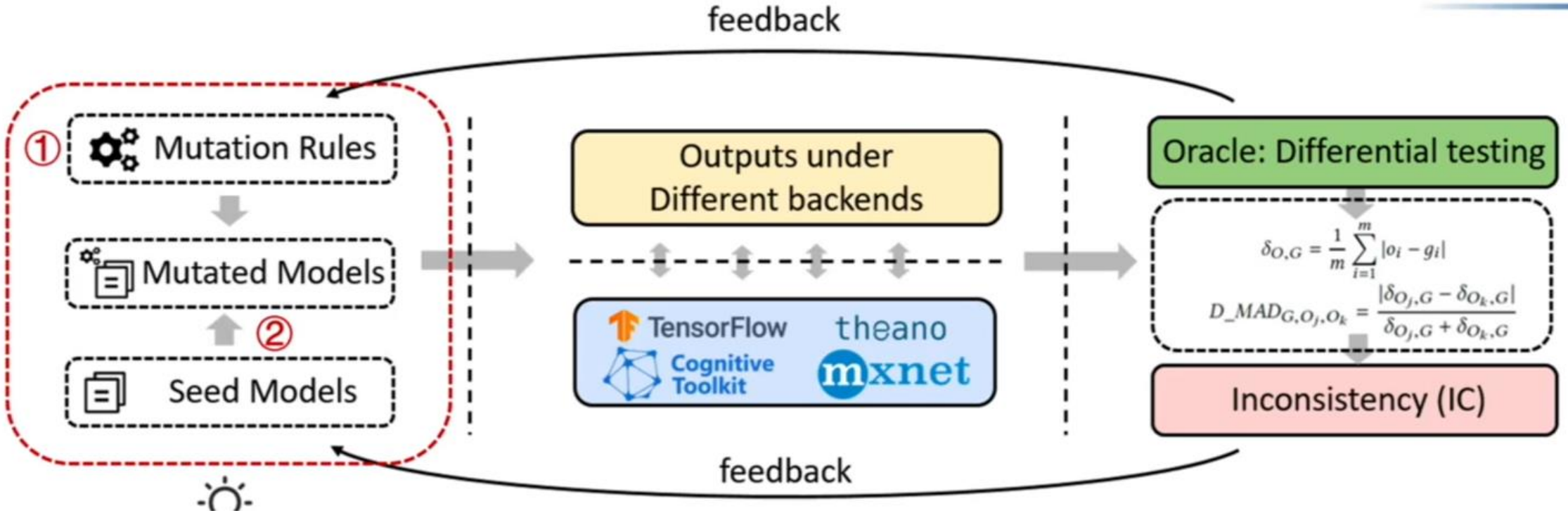
算法原理-LEMON

T	深度学习软件库漏洞挖掘
I	公开深度学习模型
P	1.选取种子模型进行突变； 2.基于四种软件库测试原模型与突变模型； 3.计算模型输出的差异值 4.更新模型与突变规则权值 5.选取和突变新模型
O	突变后的新模型

P	放大模型基于不同软件库的输出差异
C	模型测试/应用任务；差分测试
D	如何确定“有价值”模型和突变规则
L	2020 CCS A类会议

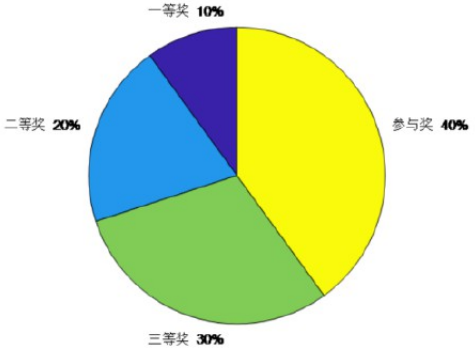
算法原理

- 算法流程图


feedback
①
Mutation Rules
Mutated Models
②
Seed Models
Outputs under
Different backends
TensorFlow
theano
Cognitive
Toolkit
mxnet
Oracle: Differential testing
$\delta_{O,G} = \frac{1}{m}\sum_{i=1}^{m}|o_i - g_i|$
$D\_MAD_{G,O_j,O_k} = \frac{|\delta_{O_j,G} - \delta_{O_k,G}|}{\delta_{O_j,G} + \delta_{O_k,G}}$
Inconsistency (IC)
feedback

算法原理

- 模型选取
  - 轮盘赌算法：起源于一个以人性命为赌注、左轮手枪为赌具的游戏
    - 根据适应度$f(x_i)$计算累积概率$q(x_i)$



$$p(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N} f_j} \qquad q(x_i)=\sum_{j=1}^{i} p(x_j)$$

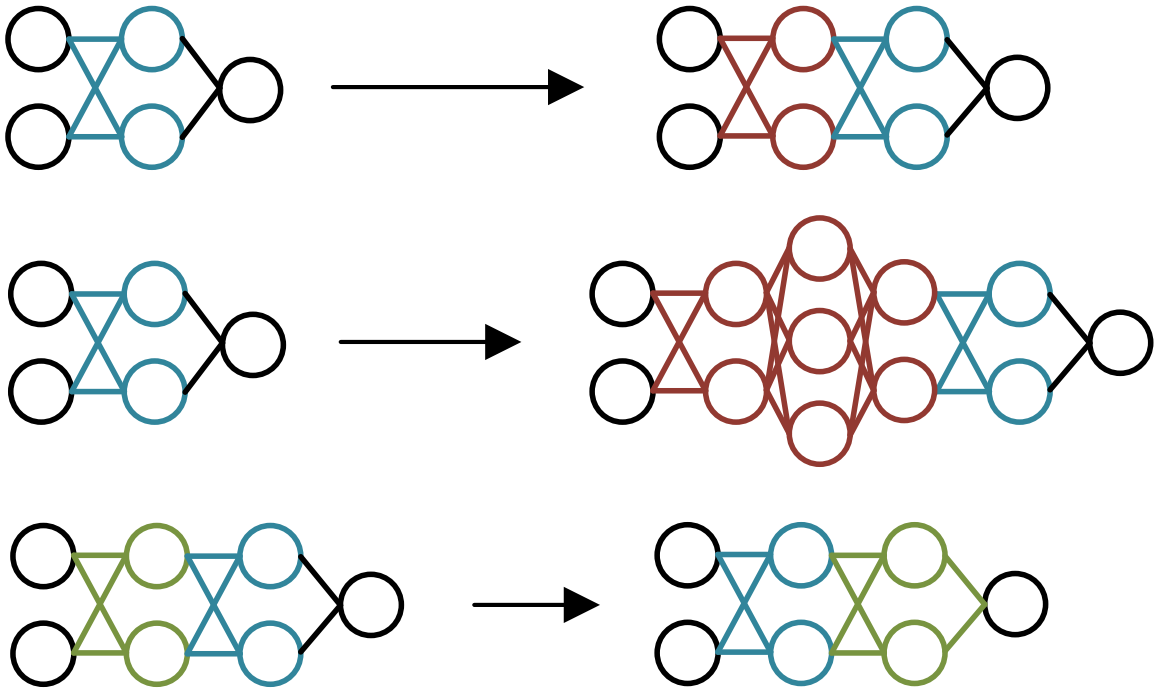
    - 随机生成$m=[m_1,m_2,\dots,m_n](\in(0,1))$
    - 若$q(x_i)>m[i]$，选中个体$x(i)$；若$q(x_i)<m[i]$，比较$m[i+1]$与$x(i+1)$
  - 应用：更大概率选取使用次数更少的模型

$$score_i=\frac{1}{c_i+1} \qquad p_i=\frac{score_i}{\sum_{k=1}^{r} score_k}$$

其中p_i为每个模型选取的概率， c_i为模型被使用的次数

算法原理

- 模型突变
  - 12种突变规则
    - 输入输出形状一致：层删除、层交换、层复制、层添加、多层添加



层删除	激活函数替换
层交换	高斯模糊
层复制	权值重组
层添加	神经元翻转
多层添加	神经元阻断
激活函数删除	神经元替换

算法原理

- 模型突变
  - 累积不一致

$$ACC(M) = \sum_{i=1}^{n} \sum_{j,k=1}^{m} D_{MAD_G,O_{ji},O_{ki}} \ (k > j)$$

$$D_{MAD_G,O_j,O_k} = \frac{|\delta_{O_j,G} - \delta_{O_k,G}|}{|\delta_{O_j,G} + \delta_{O_k,G}|} \qquad \delta_{O,G} = \frac{1}{m} \sum_{i=1}^{m} |o_i - g_i|$$

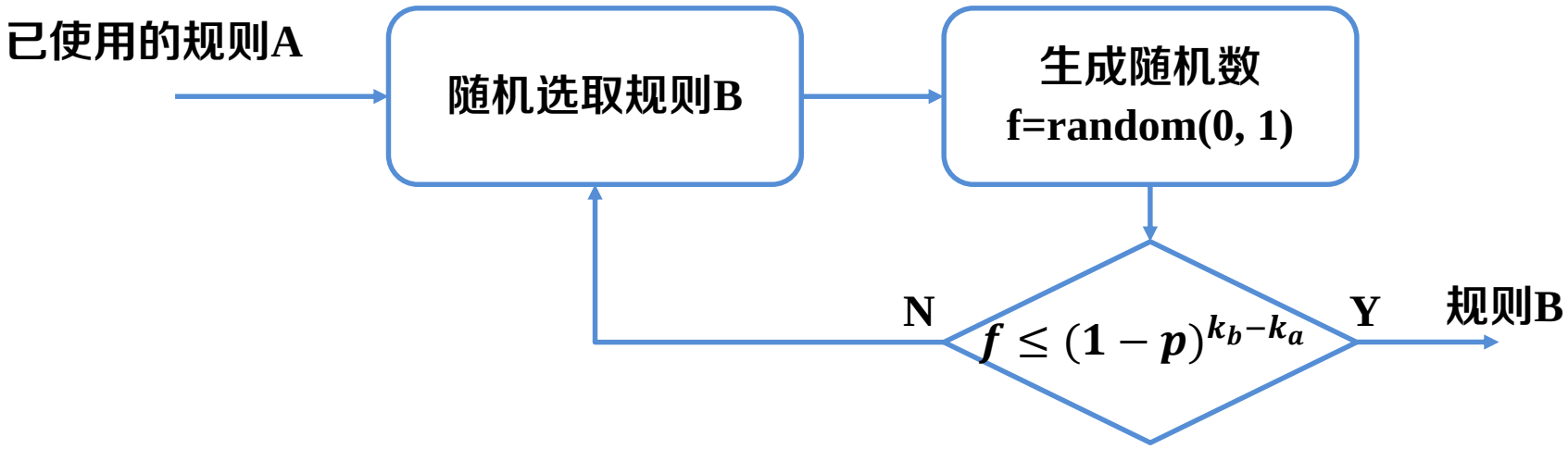
其中深度学习库$\{L_1, L_2, \dots, L_m\}$，输出测试集$\{I_1, I_2, \dots, I_n\}$

  - 若$Acc(M_m) > Acc(M_s)$，将M_m加入种子池中。
  - 突变策略：突变规则的选取受到突变规则之前行为的影响

算法原理

- 突变策略
  - 引入MCMC：根据当前突变规则A对下一个突变规则B进行抽样
    - $Ratio(MU)$为该规则放大累积不一致的次数，基于该次数排序得到k（排名）和被随机选中概率p；
    - 设置概率分布为几何分布；

$$P_s(X=k)=(1-p)^{k-1}p \qquad P_a(MU_b|MU_a)=\frac{P_s(MU_b)}{P_s(MU_a)}=(1-p)^{k_b-k_a}$$



实验分析

- 实验设置
  - 初始模型与数据集
    - CNN*10，LSTM*2
  - 测试对象
    - TensorFlow、CNTK、Theano、Mxnet
  - 实验指标
    - 不一致数量（一个模型四个底层库）
    - 检测出的缺陷数量
- 缺陷数量
  - 13*IC
  - 6*Crash
  - 4*NaN
  - 1*performance

Library	#IC Bugs	#Crash	#NaN	#Total
TensorFlow	4	0	1	5
Theano	3	0	1	4
CNTK	2	0	0	2
MXNet	4	6	2	12
Keras	1 performance bug			24

模型	数据集
AlexNet	CIFAR-10
LeNet5	Fashion-MNIST
LSTM-1	MNIST
LSTM-2	Sine-Wave
ResNet50	Stock-Price
MobileNetV1	ImageNet
InceptionV3	ImageNet
DenseNet121	ImageNet
VGG16	ImageNet
VGG19	ImageNet
Xception	ImageNet

实验分析

- 模型突变
  - 突变机制显著放大不同框架的差异值

Lib Pair	E1			E2			E3			E4			E5		
	V_M	V_I	$\Uparrow_{rate}$	V_M	V_I	$\Uparrow_{rate}$	V_M	V_I	$\Uparrow_{rate}$	V_M	V_I	$\Uparrow_{rate}$	V_M	V_I	$\Uparrow_{rate}$
TF↔TH	0.83	0.31	166.45%	0.60	0.13	348.90%	0.64	0.14	357.30%	0.60	0.19	207.76%	0.59	0.15	282.14%
TF↔CN	0.70	0.34	104.71%	0.62	0.20	213.44%	0.59	0.23	160.12%	0.61	0.25	141.33%	0.60	0.24	152.47%
TH↔CN	0.84	0.39	118.30%	0.69	0.26	162.04%	0.74	0.29	152.77%	0.76	0.31	142.99%	0.74	0.33	123.45%
TF↔MX	0.85	0.55	54.08%	0.82	0.50	62.72%	0.76	0.52	47.84%	0.75	0.49	55.57%	0.80	0.49	64.12%
TH↔MX	0.92	0.71	29.16%	0.79	0.44	81.10%	0.90	0.65	37.15%	0.94	0.74	27.06%	0.81	0.57	42.11%
CN↔MX	0.83	0.45	85.06%	0.81	0.38	112.94%	0.77	0.44	72.51%	0.82	0.54	51.84%	0.80	0.46	72.94%

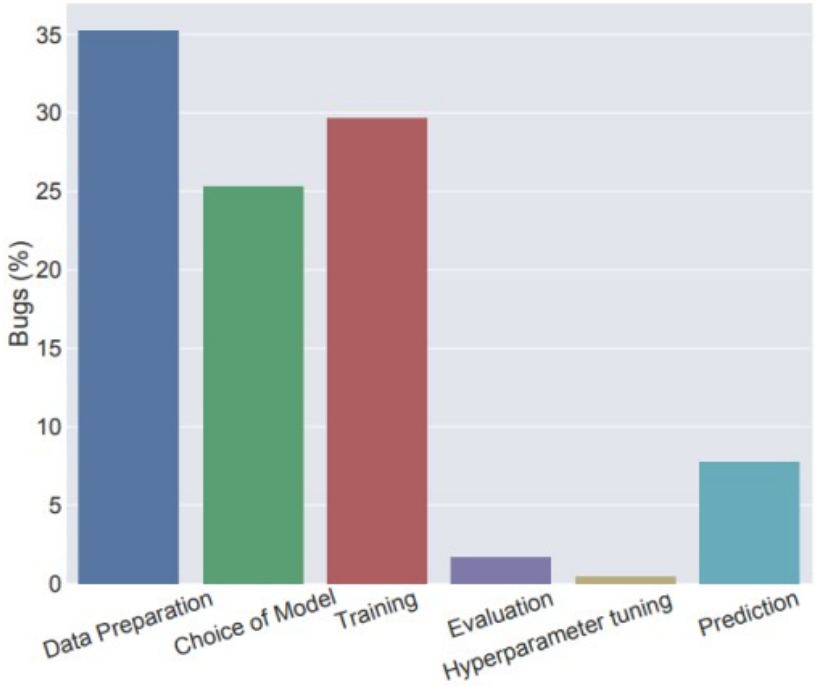
- 模型突变&突变策略
  - 启发式的模型生成策略

  VS随机模型生成
  - 发现更多不一致数量

Lib Pair	# Only mutated		# Only initial		$\Uparrow_{rate}$	
	LEMON	$LEMON_r$	LEMON	$LEMON_r$	LEMON	$LEMON_r$
TF↔TH	136	108	0	0	49.21%	28.83%
TF↔CN	181	133	0	0	69.79%	35.45%
TH↔CN	114	123	0	0	81.29%	57.16%
TF↔MX	371	269	5	55	27.88%	14.79%
TH↔MX	317	231	5	55	35.89%	27.02%
CN↔MX	382	319	5	55	54.17%	41.34%

实验分析

- 优势
  - 模型突变提供大量测试输入
  - 突变策略收集反馈信息，发现更多和数值损坏相关的缺陷
- 劣势
  - 模型预测/测试只占深度学习库较少的比例
    - 模型突变对扩大库代码覆盖的作用微乎其微
  - 缺陷定位需要大量手动工作



算法原理

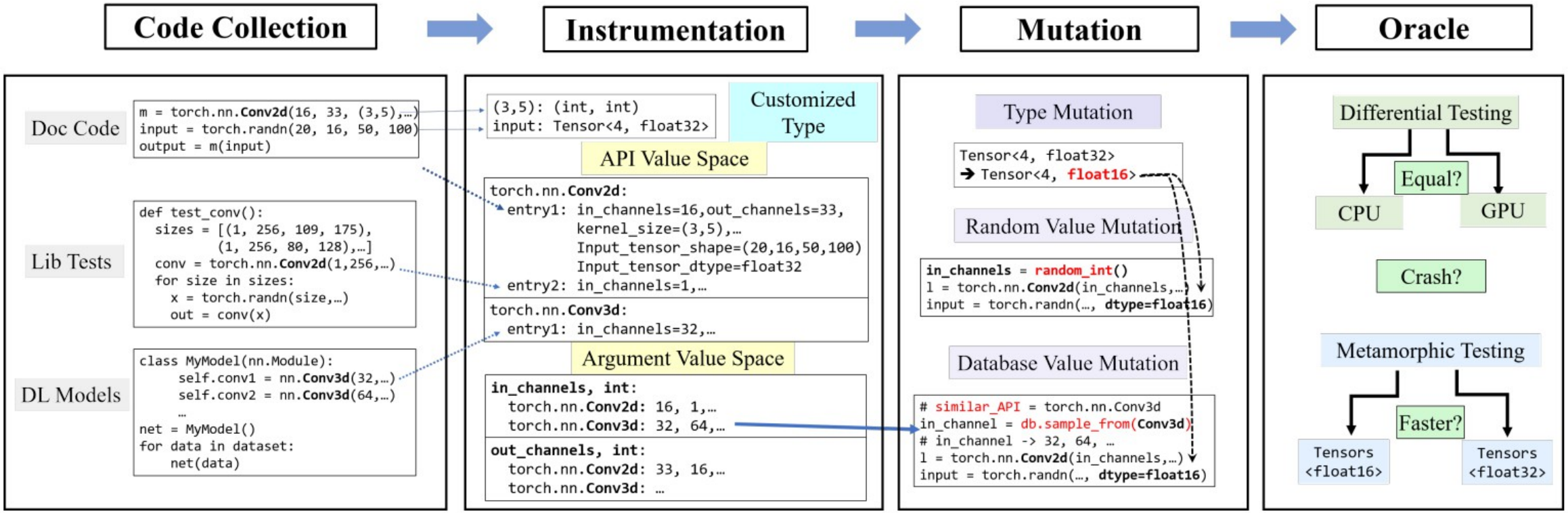
算法原理——FreeFuzz

T	深度学习软件库漏洞挖掘
I	说明书代码片段；开发者测试样例；公开深度学习模型
P	1. 运行上述开源代码 2. 设计插桩工具记录运行时API的输入 3. 生成输入的值空间和参数空间 4. 基于随机值和数据库值突变输入 5. 基于三种测试预言监测API动态执行
O	深度学习软件库漏洞

P	深度学习软件库API参数具有多样的数据格式
C	API基于Python语言；测试对象TensorFlow、Pytorch
D	如何获取已有输入值
L	2022 ACM A类会议(ICSE)

算法原理

- 算法流程图
  - 代码收集、插桩、参数突变、测试语言

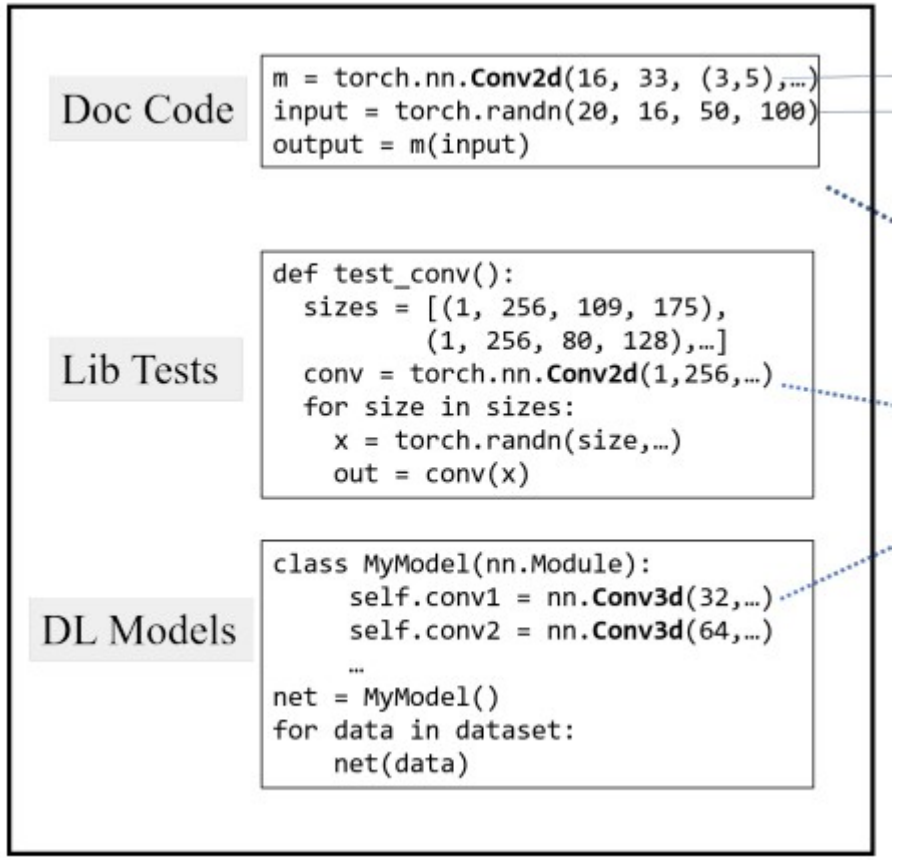


算法原理

- 代码收集
  - 说明书代码
    - 部分API说明书提供代码片段用于说明

```
>>> # non-square kernels and unequal stride and with padding and dilation
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2), dilation=(3, 1))
>>> input = torch.randn(20, 16, 50, 100)
>>> output = m(input)
```

  - 库测试
    - 定性：软件库开发者编写的测试样例
    - 定量：1420 for Pytorch、1493 for TensorFlow
  - 深度学习模型
    - 定性：常用公开模型，可用于训练和测试
    - 定量：102 for Pytorch 、100 for TensorFlow



算法原理

- 插桩
  - 意义 or 作用：获取API输入信息
  - 定制类型空间
    - 记录数据类型、维度等参数详细信息

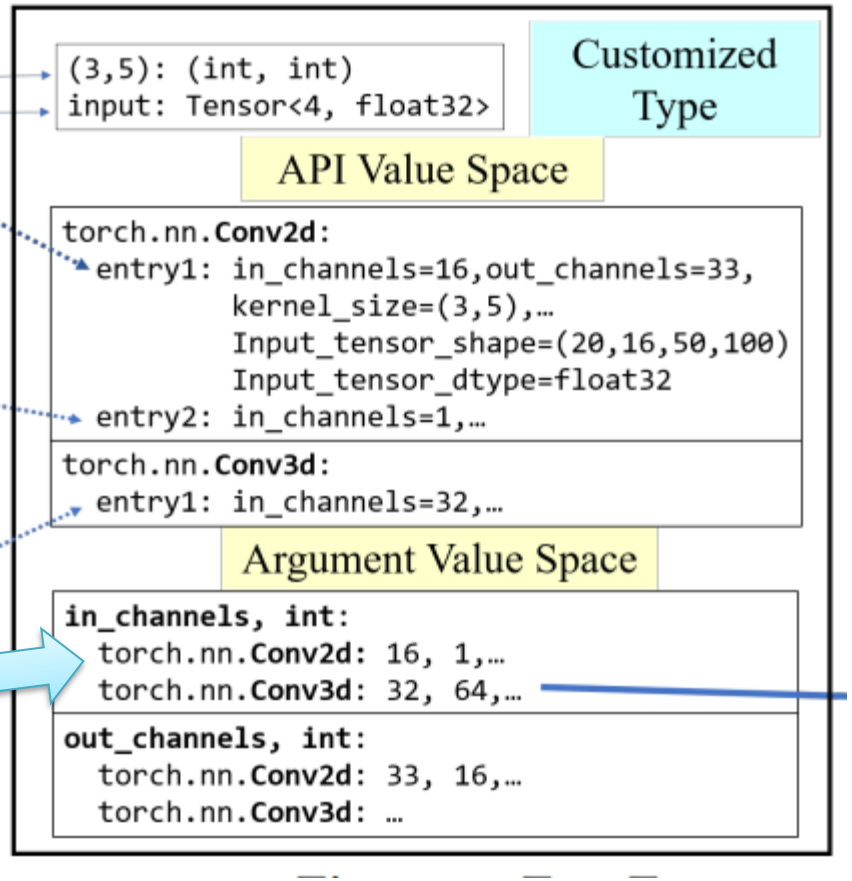
Stride=(2,1)
<class 'tuple'>

→ (int, int)

torch.randn(20, 16, 50, 100)
<class 'torch.Tensor'>

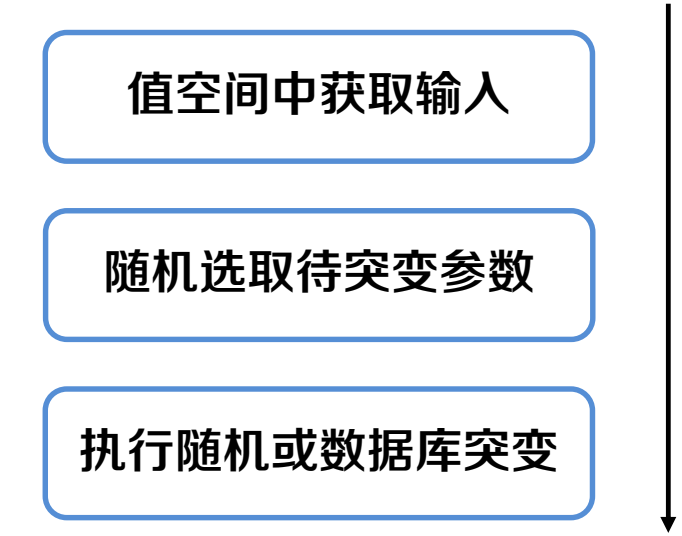
→ Tensor<4, float32>

  - API值空间
    - API参数列表及取值
  - 参数值空间
    - 参数在不同API上的取值



算法原理

- 输入突变
  - 意义 or 作用：生成大量输入用于API测试
  - 数据类型突变
    - 维度、数值类型、原始类型、元组、列表
  - 数值突变
    - 随机突变
    - 数据库值突变

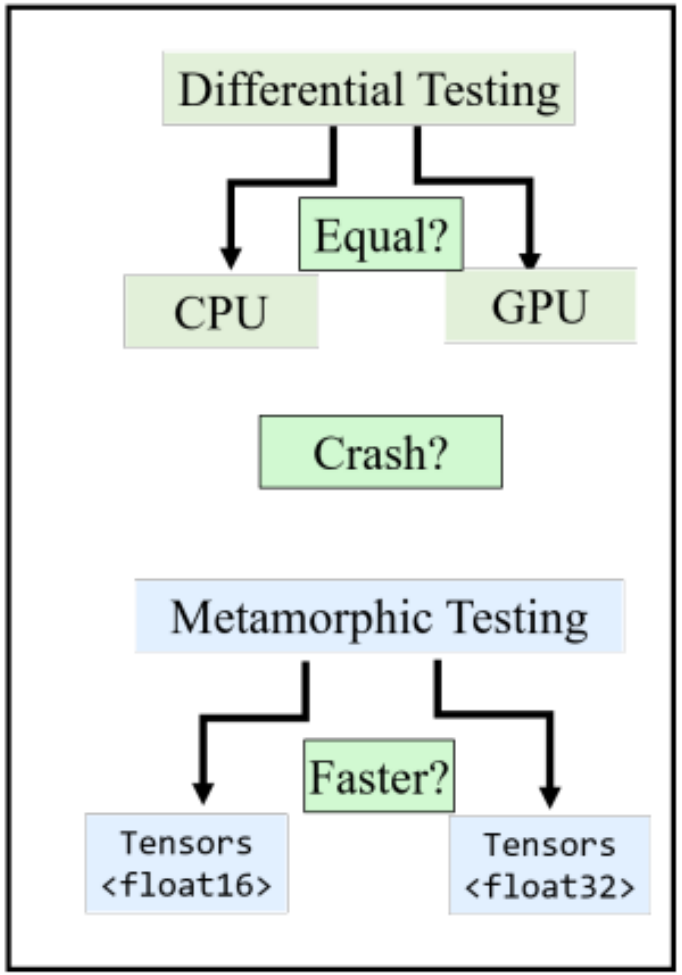


张量维度突变	Tensor<mutate(N), DT>
张量数值类型突变	Tensor<N, mutate(DT)>
原始类型突变	int/bool/float/str
元组突变	(type_mutate(int), …)
列表突变	[type_mutate(int), …]

随机/数据库张量形状	Tensor<mutate(Shape),DT>
随机/数据库张量数值	Tensor<N, mutate(Value)>
随机/数据库原始类型	int/bool/float/str
随机/数据库元组	(type_mutate(int), …)
随机/数据库列表	[type_mutate(int), …]

算法原理

- 测试预言
  - 错误计算缺陷
    - CPU和GPU执行模式下API输出的差异
  - 崩溃
- 实验指标
  - API覆盖数量
  - 值空间
  - C/C++代码覆盖行数
  - 缺陷检测数量



实验分析

- 覆盖
  - 测试版本：Pytorch1.11.0、TensorFlow2.4.0
  - 对比方法
    - LEMON(2020)
    - CRADLE(2019)
  - API覆盖、输入空间覆盖和代码覆盖均呈现良好效果
- 缺陷成果
  - 49个缺陷，38个已证实，21个新版本已修复

	FreeFuzz PyTorch				FreeFuzz TensorFlow			
	Doc	Test	Model	All	Doc	Test	Model	All
# API	427	176	145	470	486	216	269	688
# VS	1259	3383	10898	15532	1810	6879	36638	45269
Line Cov.	39272	30476	26292	42425	33906	31293	34790	39575

	FreeFuzz (tf1.14 full)	LEMON	CRADLE
# API	313	30	59
# VS	9338	1808	2893
Line Cov.	33389	29489	28967

	FreeFuzz	FreeFuzz				Confirmed (Fixed)
		-TypeMu	-RandMu	-DBMu	-AllMu	
PyTorch	28	13	24	26	5	23 (7)
TensorFlow	21	20	5	20	2	15 (14)

实验分析

- 缺陷示例
  - Pytorch，Conv3d

```
1  from torch.nn import Conv3d
2  x = torch.rand(2,3,3,3,3)
3  Conv3d(3,4,3,padding_mode='reflect')(x) # Crash
```

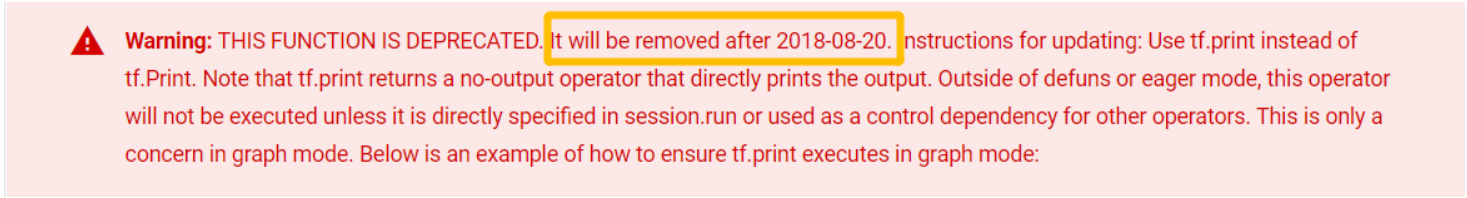
  - Pytorch，MaxUnpoold

```
1  m_gpu = torch.nn.MaxUnpool2d(2,stride=2).cuda()
2  m_cpu = torch.nn.MaxUnpool2d(2,stride=2)
3  tensor = torch.rand(1, 1, 2, 2)
4  indices = torch.randint(-32768,32768,(1, 1, 2, 2))
5  gpu_result = m_gpu(tensor.cuda(), indices.cuda())
6  cpu_result = cpu(tensor, indices) # only crash on CPU
```

优劣分析

优劣分析

- 优势
  - 开辟两种动态漏洞挖掘路线，即Model-Fuzz和API-Fuzz
  - API-Fuzz实现更高功能覆盖和代码覆盖
- 劣势
  - Model-Fuzz面临覆盖率的“硬伤”
  - FreeFuzz基于现有社区开源代码获取输入
    - 功能覆盖和代码覆盖依然受到限制
    - 版本迭代带来重复的输入构建工作



应用总结

- 应用领域
  - 形成AI框架安全性测试或漏洞挖掘体系

- 未来的发展
  - 获取API语法先验知识，直接构建输入，实现更高覆盖率
  - “漏洞” 输入空间引导，发现更多潜在缺陷

参考文献

[1] ISLAM M J, NGUYEN G, PAN R, et al. A Comprehensive Study on Deep Learning Bug Characteristics[J].,2019:510-520.

[2] WANG Z, YAN M, CHEN J, et al. Deep Learning Library Testing via Effective Model Generation[J].,2020:788-799.

[3] WEI A, DENG Y, YANG C, et al. Free Lunch for Testing: Fuzzing Deep-Learning Libraries from Open Source[J]. CoRR,2022,abs/2201.06589.

[4] https://github.com/tensorflow/tensorflow/security/advisories

[5] TensorFlow API Versions | TensorFlow Core v2.9.1 (google.cn)

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！