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

智能化系统的安全测试方法

硕士研究生　王若辉

2021年03月21日

内容提要

- 背景简介
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

内容提要

- 预期收获
  - 1.了解软件测试的意义和基本分类。
  - 2.了解软件测试的两种白盒测试方法—覆盖率测试和突变测试。
  - 3.了解当前智能化系统的安全测试方法。

背景介绍

- 软件测试

  一个测试工程师走进一家酒吧，要了一杯啤酒。

  一个测试工程师走进一家酒吧，要了一杯咖啡。

  一个测试工程师走进一家酒吧，要了0.7杯啤酒。

  一个测试工程师走进一家酒吧，要了-1杯啤酒。

  一个测试工程师走进一家酒吧，要了一份asdfQwer@24dg!&*(@。

  一个测试工程师走进一家酒吧，要了NaN杯Null。

  一个测试工程师走进一家酒吧，要了一杯啤酒‘;DROPTABLE酒吧。

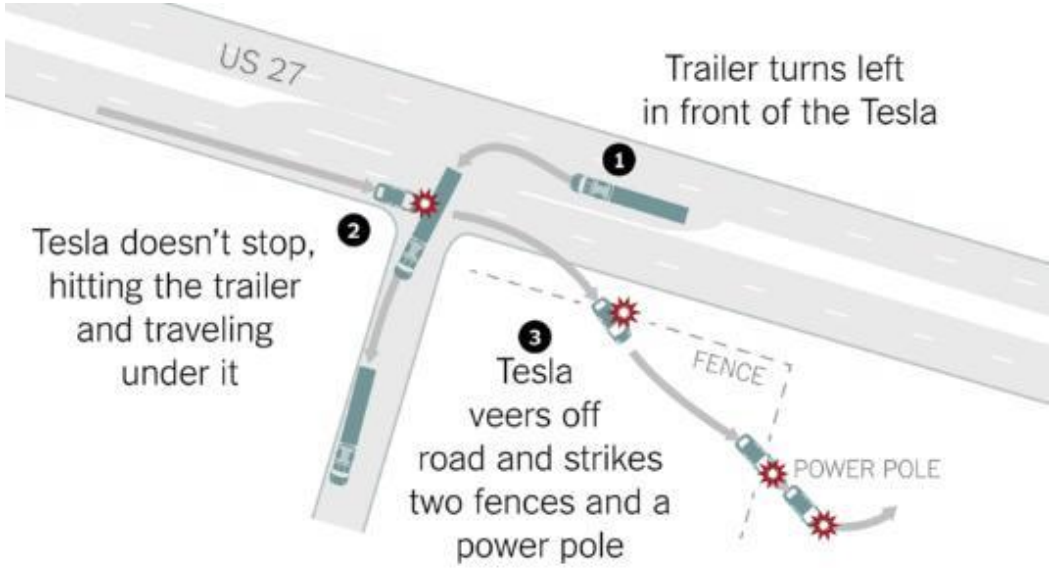
  ……

  最后，测试工程师满意的离开了酒吧。

- 然后一名顾客走进酒吧，点了一份炒饭，酒吧炸了。

背景介绍

- 2016年5月，在佛罗里达的一条高速公路上，一辆开启了 Autopilot 模式的特斯拉发生了车祸撞上了一辆拖车，驾驶员致死。这是特斯拉全球首例死亡事件。
  - 拖车的两个特征“在明亮的天空下呈白色”、“驾驶座高度高” 。
  - 具备该特征的数据并不是特斯拉测试集的一部分，因此在测试过程中从未出现过。



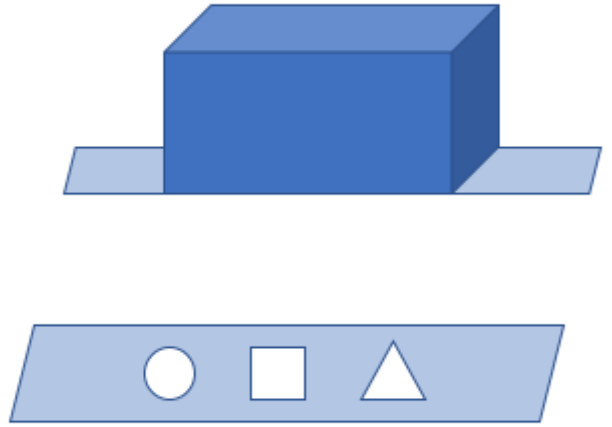
背景介绍

- 测试是程序的执行的过程，目的在于发现错误，测试只能证明有错，不能保证无错。好的测试是能发现错误的测试。
  - 如何尽可能地保证测试的充分性?
  - 如何测试智能化系统?

基本概念

基本概念

- 按测试过程是否查看代码，测试被分为三类
  - 黑盒测试（功能测试）
    - 着眼于程序外部结构，只检查软件是否按照需求说明书的规定正常使用。
    - 仅关心输入和输出。
  - 白盒测试（结构测试）
    - 全面了解程序内部逻辑结构、对所有逻辑路径进行测试。
    - 从检查程序的逻辑着手，得出测试数据。
  - 灰盒测试
    - 介于白盒测试与黑盒测试之间。
    - 对系统的局部认知。



基本概念

- 覆盖率测试，软件测试中常用的白盒测试方法
  - 语句覆盖——每条可执行语句。
  - 判定（分支）覆盖——每个判定表达式。
  - 条件覆盖——判定表达式的每个条件的值。
    - 判定/条件覆盖——每个判定表达式 U 判定表达式的每个值。
    - 多重条件覆盖——每个判定的所有可能的条件取值组合。
  - 路径覆盖——每条可能的路径。

基本概念

- **突变测试**
  - **突变测试通常对程序的源代码或者目标代码做小的改动，并把截然不同的错误行为作为预期。如果测试数据没有觉察到这种小改动带来的错误，就说明这个测试是有问题的。**
  - *A testing methodology in which two or more program mutations are executed using the same test cases to evaluate the ability of the test cases to detect differences in the mutations.* **——IEEE (Std 610.12-1990)**
- **突变测试理论的两个基本假设**
  - **程序员能力假设**（Competent Programmer Hypothesis，CPH）
  - **耦合效应假设**（Coupling Effect，CE）

基本概念

- 示例
  - 现在有一个为患者分配医生的程序
    - 1.读取病人的年龄。
    - 2.如果病人大于14岁，就会分配一个普通医师作为他们的主要医生。

```
1：  input Age;
2：  if Age > 14 do:
3:          Doctor = General Physician();
4：  end if;
```

  - 作为一名测试人员，给出的测试数据集是{0,13,20}。
  - 目的：评估测试数据集的质量。

基本概念

- 示例
  - 对程序进行以下几种突变
    - Mutant 1：if *Age* > 14 do: => if *Age* < 14 do:
    - Mutant 2：if *Age* > 14 do: => if *Age* =14 do:
    - Mutant 3：if *Age* > 14 do: => if *Age* >=14 do:
    - Mutant 4： *Doctor* = General Physician(); => *Doctor* = Wang();

    - Mutant 5: if *Age* > 14 do: => if *Age* %%14 do:
    - Mutant 6: **删除** *Doctor* = General Physician();
    - Mutant 7: if *Age* > 14 do: => if *Age* >14 & *Age* >14 do:

基本概念

- 示例
  - 输出结果（√表示第三行函数执行，×反之，err表示报错）

Test set-Data	Expected Result	Moutant 1	Moutant 2	Moutant 3	Moutant 4	Moutant 5	Moutant 6	Moutant 7
0	×	√	×	×	×	err	×	×
13	×	√	×	×	×	err	×	×
20	√	×	×	√	×	err	×	√

基本概念

- 示例
  - 输出结果（√表示第三行函数执行，×反之，err表示报错）

Test set-Data	Expected Result	Moutant 1	Moutant 2	Moutant 3	Moutant 4	Moutant 5	Moutant 6	Moutant 7
0	×	√	×	×	×	err	×	×
13	×	√	×	×	×	err	×	×
20	√	×	×	√	×	err	×	√

  - 测试数据集“杀死”了1、2、4、5、6，未能“杀死”3、7。
  - 如果测试数据集“杀死”了所有突变体，它就是有效的。

基本概念

- 示例
  - 但是，对于Moutant 5、6 和 7：
    - Moutant 5: if Age > 14 do: => if Age %%14 do:
      - 语法错误突变
    - Moutant 6:删除 Doctor = General Physician();
      - 与输入无关突变
    - Moutant 7:if Age > 14 do: =>if Age >14 & Age >14 do:
      - 等效突变

Moutant 5	Moutant 6	Moutant 7
err	×	×
err	×	×
err	×	√

  - 如果在测试的最后阶段还有一些突变体存在，这意味着它可能是一个无效的突变也可能是数据集不充分。

基本概念

- 将覆盖率测试技术应用于智能化系统
  - DeepXplore (SOSP 2017)
  - 意义：
    - 首次将白盒测试应用于深度学习系统测试。
    - 提出神经元覆盖率作为测试充分性验证标准。
  - 问题：
    - 神经元覆盖率无法表明测试样本集覆盖了 DNN模型的所有可能的输出。
    - 神经元覆盖率强调个体神经元的作用，没有考虑神经元之间的组合关系。

- 将突变测试技术应用于智能化系统
  - ？

算法原理

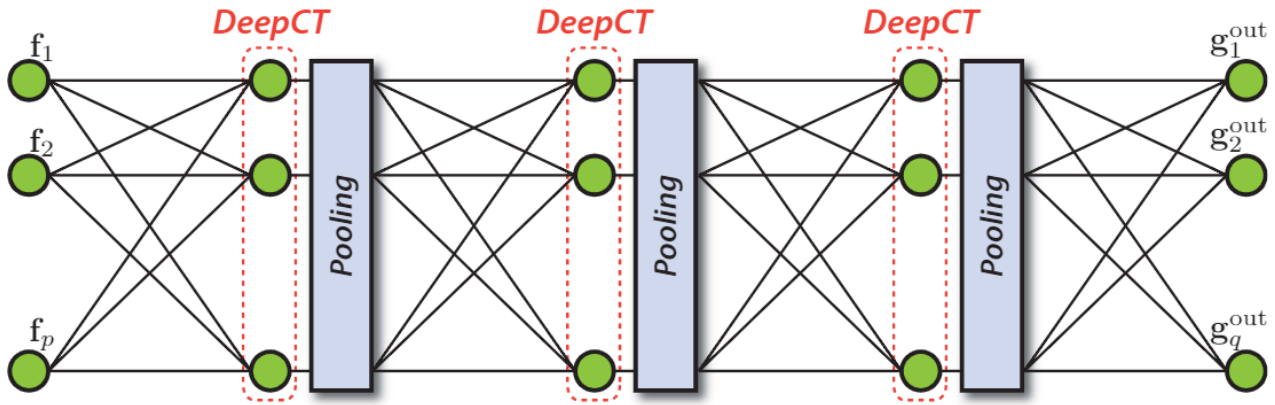
算法原理

T	通过组合覆盖率评估测试充分性，指导生成测试样本
I	DNN模型、种子测试集
P	1. 更新组合覆盖表 2. 寻找未覆盖到的神经元激活配置 3. 随机选择指导生成测试
O	测试样本

P	将组合测试引入神经元覆盖率准则中
C	同层神经元之间存在逻辑关系，共同决定下一层神经元逻辑
D	如何定义一个神经元组合关系的覆盖标准
L	CCF B类会议

算法原理

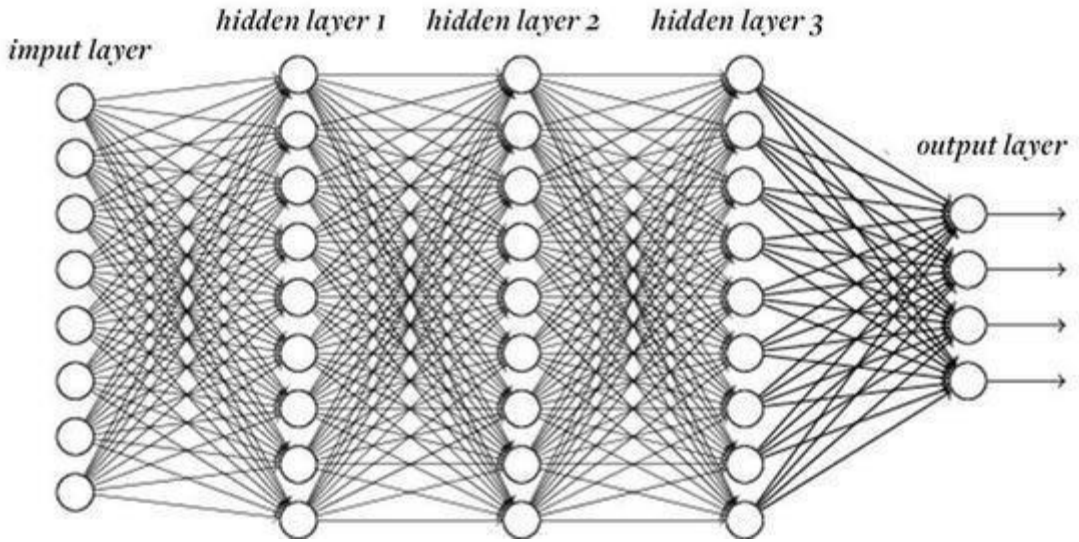
- 层析（Tomographic）
  - 为什么侧重分层进行测试?



  - DL模型各个层中的权重是结构化的、层次化的。
  - 在任何一个中间层，神经元都会合并所有先前的信息，并以更高的抽象级别作为输入数据传递给后续层。
  - 为了检验DNN模型，最好是像计算机断层扫描一样逐层检验，系统地挖掘其潜在的特征空间。

算法原理

- 组合（Combinatorial）
  - 为什么系统检测每层内神经元之间的组合作用?



- 每层中的神经元之间没有直接的（有形的）交互作用。
- 但是神经元之间可能存在逻辑的（无形的）交互作用，其中当前层的神经元共同决定下一层神经元的逻辑。

算法原理

- 神经元激活配置
  - $A(n_i, x) \in 0,1$ 表示神经元n_i在给定输入x下的激活状态。
  - 对于一组神经元$M = \{n_1, n_2, \ldots\ldots, n_k\}$
    - M 的神经元激活配置是元组 $c = (b_1, b_2, \ldots\ldots, b_k)$，其中$b_i = \{0,1\}$ [2] 。
- 神经元t-way组合
  - 给定一个测试集 T 和第 i 层的神经元集合L_i
    - 使用$\Theta(t, L_i)$来表示 L_i 中神经元所有的 t-way 组合。
    - 使用$\Theta_{full}(t, L_i, T)$表示 L_i 的神经元的所有t-way组合的所有神经元激活配置都被 T 完全覆盖。
    - 对于一个t-way组合$\theta \in \Theta(t, L_i)$，用我们用$C(t, \theta, T)$来表示 T 所覆盖的 θ 的神经元激活配置集。

算法原理

- t-way 组合稀疏覆盖

$$\mathrm{SparseCov}(t, L_i, T) = \frac{|\{\theta \in \Theta(t, L_i) | \theta \in \Theta_{full}(t, L_i, T)\}|}{|\Theta(t, L_i)|}$$

  - 达到完全覆盖的组合数与所有组合数之比。

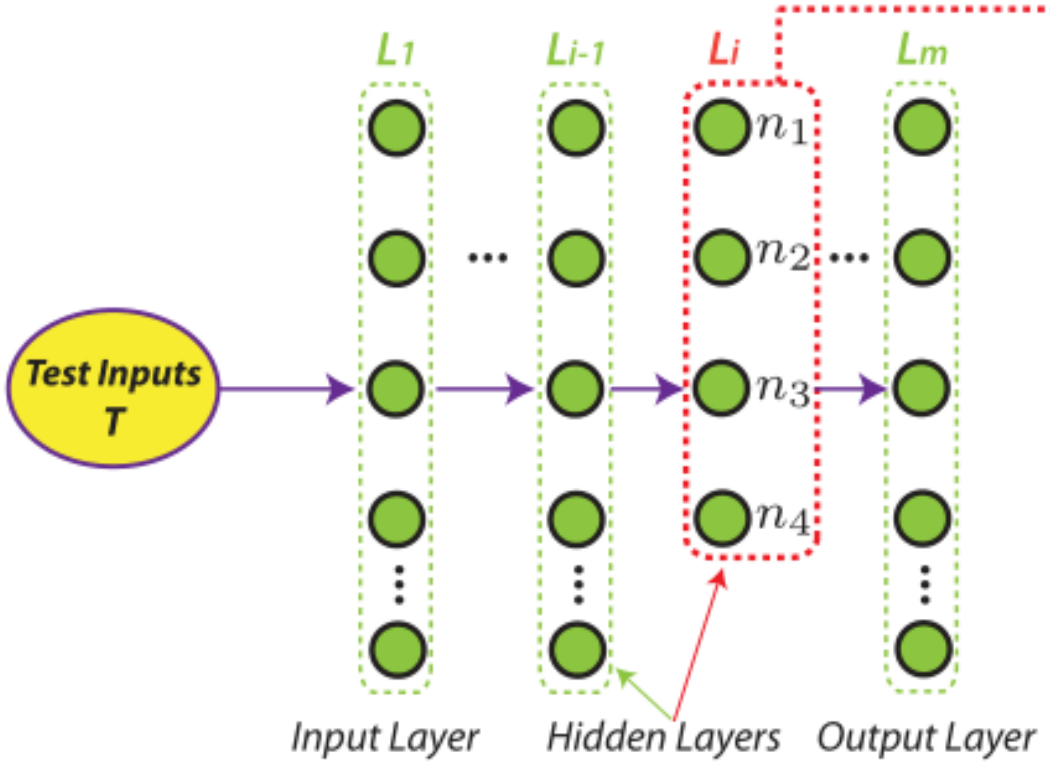
- t-way 组合稠密覆盖

$$\mathrm{DenseCov}(t, L_i, T) = \frac{\sum_{\theta \in \Theta(t, L_i)} |C(t, \theta, T)|}{2^t |\Theta(t, L_i)|}$$

  - 当前神经元激活配置种类数与神经元激活配置所有情况总数之比。

算法原理

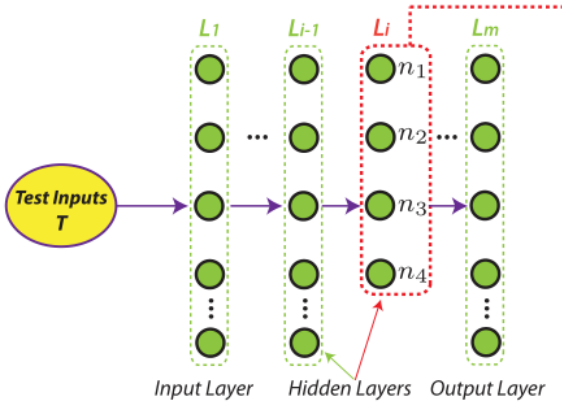
- 示例



n_1	n_2	n_3	n_4
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

算法原理

- 示例
  - L_i 层共有4个神经元$\{n_1, n_2, n_3, n_4\}$
  - 2-way组合共有6种：
    - (n_1, n_2) (n_1, n_3) (n_1, n_4) (n_2, n_3) (n_2, n_4) (n_3, n_4)
  - 每种2-way组合共能产生4种配置：
    - (0,0) (0,1) (1,0) (1,1)
    - 所以这 4 个神经元共能产生 24 种激活配置
  - 其中(n_1, n_3) = (0,1)、(n_1, n_3) = (1,0)、(n_2, n_4) = (0,1)、(n_2, n_4) = (1,0) 没有覆盖到。
    - L_i 层的2-way组合稀疏覆盖为 4 / 6 =66.7%
    - L_i 层的2-way组合稠密覆盖为 20 / 24 =83.3%



n_1	n_2	n_3	n_4
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

算法原理

- 生成数据集

Algorithm 1: DeepCT Test Generation

INPUT: DNN N, t-way, CT Criteria CTC, seeding Tests T_s
OUTPUT: Passed Test Suite T, Adversarial Test Suite T'

1: $T \leftarrow \{\}, T' \leftarrow \{\}$
2: CT_table←initialize_CT_coverage_table(T_s, t-way, CTC)
3: **for** $t \in T_s$ **do**
4: $T_{work} \leftarrow \{\}$
5: **for** each layer neuron set L_i in N **do**
6: CT_table ← update_CT_coverage(CT_table, T_{work}, CTC)
7: CT_targets←calculate_CT_targets(CT_table, T_{work}, L_i, CTC)
8: **while** CT_targets $\neq \emptyset \wedge$ time budget exists **do**
9: ct_j ←random_select(CT_targets)
10: gen_tests←TestGen(t, ct_j)
11: covered_targets ← cal_cov_targets(CT_table, T_{work}, gen_tests, CTC)
12: CT_targets←CT_targets - covered_targs
13: update_TestSuite(gen_tests, T_{work}, T, T')
14: **return** T, T'

Step.2:CT覆盖表初始化。

Step.6-7:对生成的测试进行覆盖分析计算L_i层未覆盖的激活配置。

Step.9-12:在未覆盖的激活配置中随机选择一个ct_j，生成覆盖ct_j的测试，并检测该测试对其他未覆盖目标的覆盖情况。

算法原理

- 实验过程
  - 使用开源数据集MNIST预训练好2个DNN模型。
  - 从MNIST中筛选了1000个测试作为种子测试集，通过随机生成的方式生成了10000条测试数据，并对两个DNN模型进行检测。
    - 共有26种子测试集可导致两个模型出现问题。
  - 对于剩余734个种子，随机抽取50个测试，通过DeepCT算法递增地生成测试，以逐层覆盖神经元激活配置。
  - 统计各层覆盖比例、完备性，观察对抗样本的比例。

算法原理

- 实验结果

	Testing Method	2-way Combinatorial Testing Coverage (%)				Accu. Tests	Adv. Per.(%)
		SparseCov	DenseCov	(0.5,2)	(0.75,2)		
DNN_1	Random	2.28	34.95	33.75	3.75	10,000	0.00
	CT l_1	60.27	81.56	95.01	70.98	4,073	0.29
	CT l_2	76.94	91.98	99.67	91.30	6,768	2.17
	CT l_3	93.62	98.23	100.00	99.32	8,032	9.91
DNN_2	Random	1.18	32.56	26.98	2.10	10,000	0.00
	CT l_1	46.96	75.10	91.95	61.50	8,547	1.87
	CT l_2	68.91	87.52	98.64	82.55	11,573	3.53
	CT l_3	97.15	99.05	100.0	99.03	13,129	8.84
	CT l_4	97.41	99.11	100.0	99.03	13,217	9.35
	CT l_5	97.81	99.21	100.0	99.03	13,351	9.98

- 第4、5列： (n，t)表示 t-way 组合中神经元激活配置超过 n 的比例。

优劣分析

- 优势
  - 将组合测试思想引入到神经元覆盖率准则。
  - 充分考虑神经元之间组合关系。

- 劣势
  - 未考虑DNN内部的层间关系。
  - 不能扩展到其他有不同种类层的DNN中。
  - 论文中未说明DeepCT的测试生成方法。

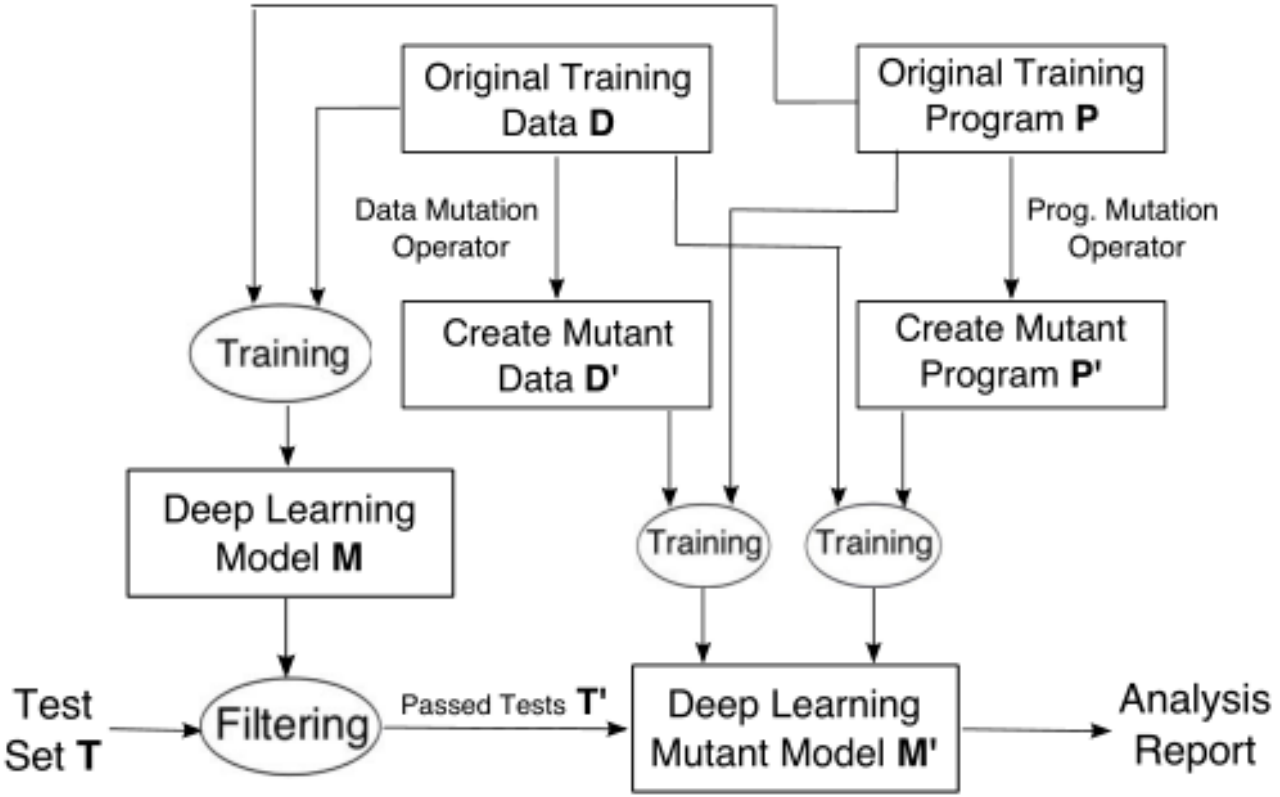
算法原理

T	检测测试数据的充分性
I	突变算子、DNN模型、测试数据
P	1. 使用原DNN模型筛选测试数据 2. 使用突变算子对模型/数据/训练好的模型进行突变 3. 检测突变后突变分数/平均错误率
O	突变分数/平均错误率

P	将突变测试技术引入DNN模型测试
C	可直接将微小的突变作用于DNN模型/训练好的DNN模型上
D	如何为数据驱动型程序设计有效的突变算子
L	CCF B类会议

算法原理

- 源级别突变
  - 与传统软件突变测试相似，直接将突变引入DL系统的编程源。



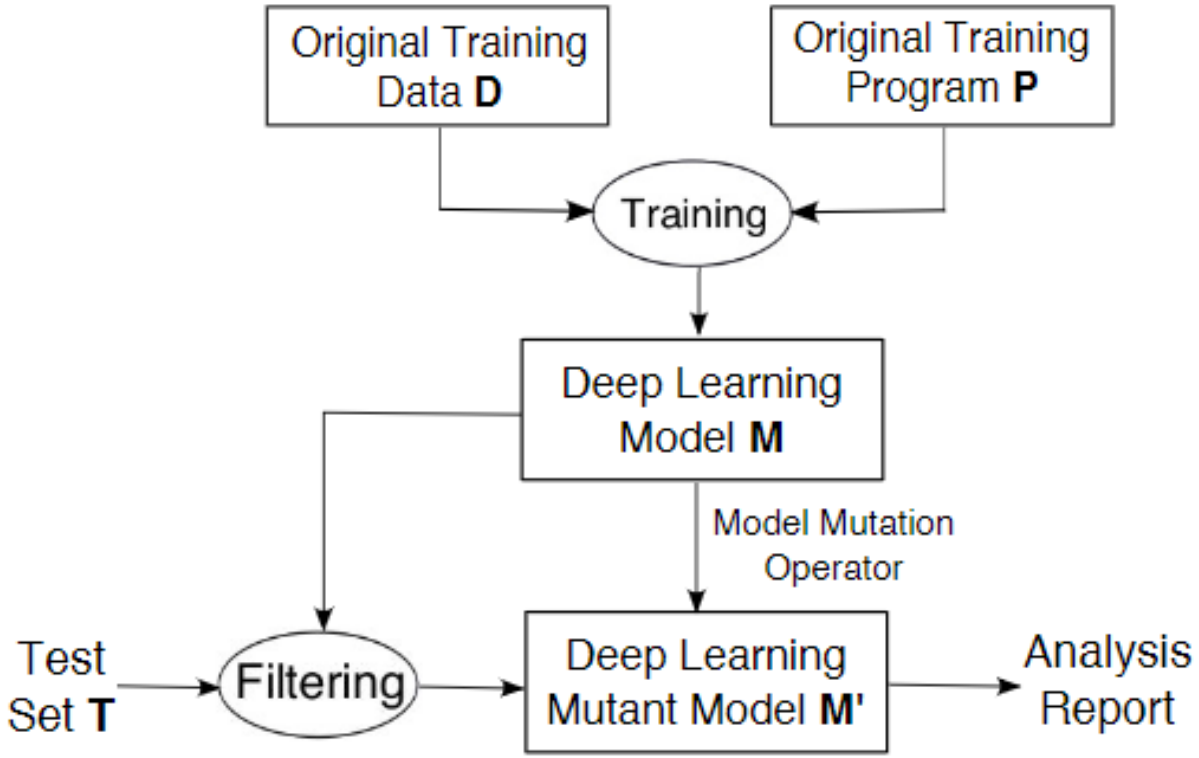
算法原理

- 源级别突变

	故障类型	等级	目标	描述
1	数据重复（DR）	全局、局部	数据	1）重复训练数据；2）重复训练特定类型的数据
2	标签错误（LE）	全局、局部	数据	1）伪造数据结果；2）伪造特定特定数据结果
3	数据缺失（DM）	全局、局部	数据	1）删除所选数据；2）删除特殊类型数据
4	数据混序（DF）	全局、局部	数据	1）随机打乱训练数据；2）随机打乱特殊类型数据
5	噪声干扰（NP）	全局、局部	数据	1）向训练数据添加噪声；2）向特殊类型数据添加噪声
6	层移除（LR）	全局	程序	删除层
7	层添加（LA）	全局	程序	添加层
8	Act.Fun.Remov(AFR)	全局	程序	删除激活函数

算法原理

- 模型级别突变
  - 将突变引入已经训练好的DL模型中。



算法原理

- 模型级别突变

序号	故障类型	目标	描述
1	高斯模糊（GF）	权重	使用高斯模糊权重
2	打乱权重（WS）	神经元	随机打乱选择的权重
3	神经元阻断（NEB）	神经元	阻断神经元对下一层的影响（可通过权重设置为0实现）
4	神经元反转（NAI）	神经元	反转神经元激活状态（改变符号？或其他）
5	神经元切换（NS）	神经元	切换同一层的两个神经元
6	层停用（LR）	层	取消某一层的影响
7	层添加（LA）	层	添加层
8	Act.Fun.Remov(AFR)	层	删除激活函数

算法原理

- 衡量指标
  - 对于测试数据 $t' \in T'$ ，称t' 杀死了$c_i \in C$ 类的突变体$m' \in M'$ 满足以下两个条件
    - t' 被原始DL模型M 正确分类为c_i ；
    - t' 未被突变DL模型m'分类c_i 。
  - 突变分数

$$\text{MutationScore}(T', M') = \frac{\sum_{m' \in M'} |\text{KilledClasses}(T', m')|}{|M'| \times |C|}$$

  - 平均错误率

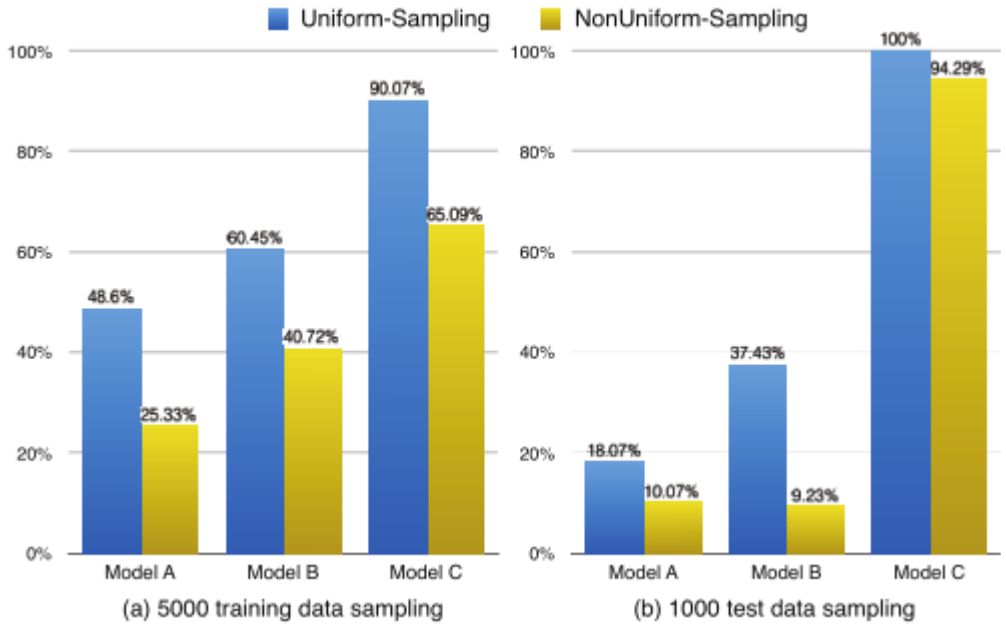
$$\text{AveErrorRate}(T', M') = \frac{\sum_{m' \in M'} \text{ErrorRate}(T', m')}{|M'|}$$

  - 一个好的突变候选不应该引入较大的行为差异。

算法原理

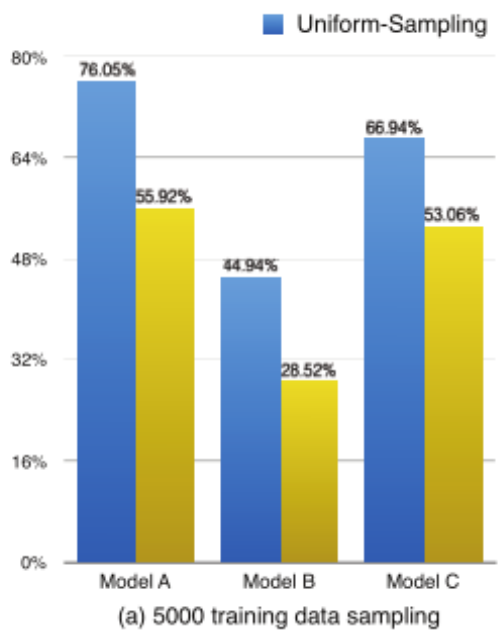
- 实验结果

Model	Source Level (%)				Model Level (%)			
	5000 train.		1000 test.		5000 train.		1000 test.	
Samp.	Uni.	Non.	Uni.	Non.	Uni.	Non.	Uni.	Non.
A	2.43	0.13	0.23	0.17	4.55	4.30	4.38	4.06
B	0.49	0.28	0.66	0.21	1.67	1.56	1.55	1.47
C	3.84	2.99	17.20	13.44	9.11	7.34	11.48	9.00

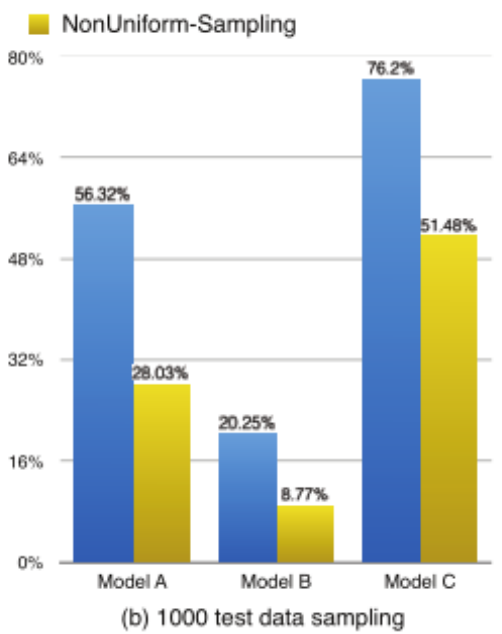


(a) 5000 training data sampling

(b) 1000 test data sampling



(a) 5000 training data sampling



(b) 1000 test data sampling

优劣分析

- 优势
  - 首次将突变测试引入智能化系统测试。
  - 设计了两种级别的突变。

- 劣势
  - 突变分数和平均错误率两种衡量指标过于简单。
  - 16种突变算子局限于DNN模型。

应用总结

- 应用价值
  - 智能化系统的自动化测试。
  - 指导智能化系统测试数据的生成。
- 应用领域
  - 自动驾驶。
  - 恶意软件检测。
  - ……
- 未来发展
  - 向其他深度学习模型进行扩展。

参考文献

- [1] Pei K, Cao Y, Yang J, et al. Deepxplore: Automated whitebox testing of deep learning systems[C]//proceedings of the 26th Symposium on Operating Systems Principles. 2017: 1-18.

- [2] Ma L, Juefei-Xu F, Xue M, et al. Deepct: Tomographic combinatorial testing for deep learning systems[C]//2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2019: 614-618.

- [3] Ma L, Zhang F, Sun J, et al. Deepmutation: Mutation testing of deep learning systems[C]//2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2018: 100-111.

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！