# Deep Learning Library Testing via Effective Model Generation

## 问题背景

时下有许多关注于DL（Deep Learning）模型的研究，但是对于深度学习库的测试却少之又少。测试深度学习库面临两大难题：

- 没有办法获得大量的数据模型
  - 现有的模型只关注于流行的任务，只覆盖了深度学习库的一部分模型
- 难以暴露由模型BUG产生的问题
  - 因为随机性等不确定因素也会产生不一致性，即**如何区别一个问题到底是由BUG产生还是由随机性因素产生**

## 类似的工具

### CRADLE

原理：利用已有的模型来进行输入，用差分测试来捕获错误

贡献：

- 定义了两个指标来衡量预测输出的不一致程度
- 设置了一个阈值来区分真正的错误和不确定的影响

缺陷：

- 依据现有的模型来进行测试，无法获得大量的测试模型
- 简单依据阈值，很难区分真正的错误和不确定带来的影响，它们之间的差异往往很小。

简而言之，作者认为CRADLE并**没有彻底解决**深度学习库面临的两大问题

基于此，作者提出了**Lemon**这种方法来测试深度学习库。作者分别用以下两种方法来解决

- **用变异测试来产生模型**(针对第一点，获得大量数据模型)
- **运用启发式算法**，扩大由BUG产生的问题的不一致性（换句话说就是扩大真正bug和不确定因素产生的问题之间的差异）

## 模型架构

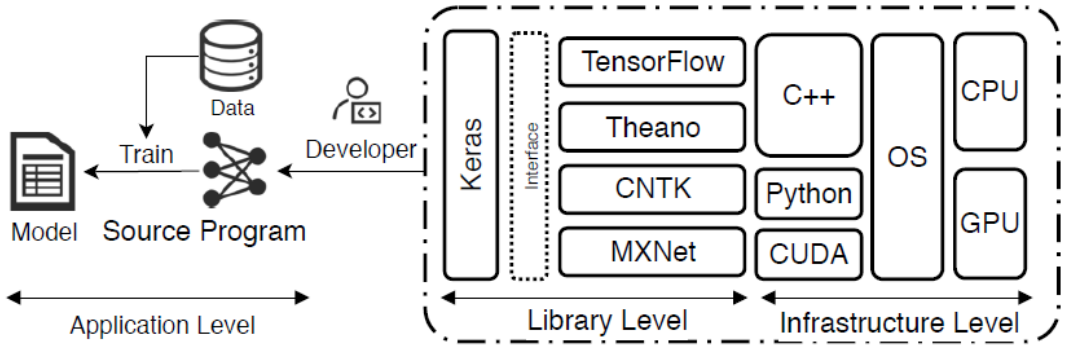



**Figure 1: The structure of a typical DL system**

图1展示了一个DL系统的典型架构，该论文针对Library Level进行研究

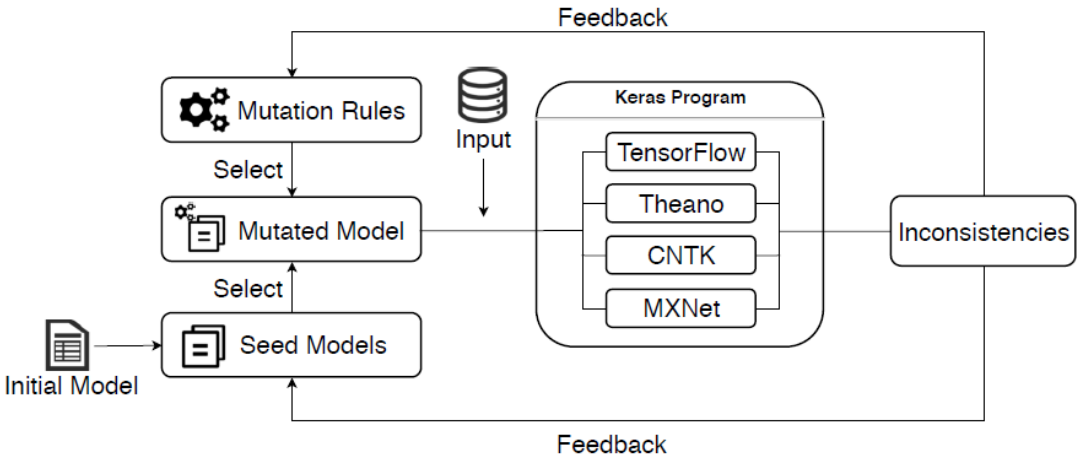



**Figure 2: Overview of LEMON**

图二展示了LEMON的架构，在利用已有模型的基础上，作者设计了一系列变异规则来有效生成DL模型，并设计了启发式算法，制定了变异规则和种子模型的选择方法。

# 核心算法

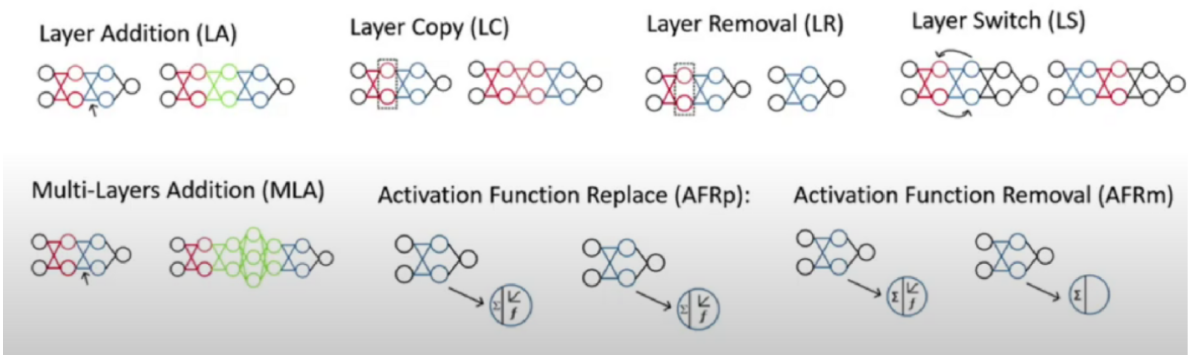
## Model Mutation

该算法分为两部分，Intact—Layer Mutation Rules和Inner—Layer Mutation Rules的设计。分别针对模型的层级和每个层级的神经元

### Intact—Layer Mutation Rules

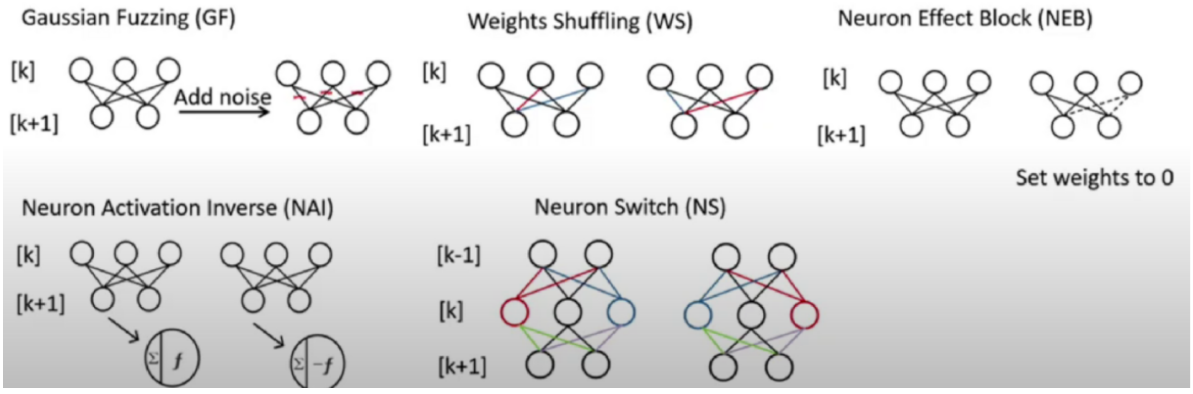
- Layer Removal (LR):
  - 从模型中移除一层
- Layer Switch (LS):
  - 交换模型中的两层
- Layer Copy (LC):
  - 将某一层进行复制
- Layer Addition (LA):
  - 为模型增加一层

- Multi-Layers Addition (MLA):
  - 为模型增加多层
- Activation Function Removal (AFRm):
  - 删除激活层的功能
- Activation Function Replace (AFRp):
  - 替换激活层的功能




### Inner—Layer Mutation Rules

- Weights Shuffling (WS):
  - 交换神经元之间链接的权重
- Neuron Activation Inverse (NAI):
  - 在将神经元传递给激活函数之前，改变一个神经元的输出值的符号
- Neuron Effect Block (NEB):
  - 将该神经元设置到下一层的权重设置为0
- Neuron Switch (NS):
  - 交换两个神经元即其链接的位置
- Gaussian Fuzzing (GF):
  - 按高斯分布，为一个神经元添加噪声




## Heuristic-Based Model Generation

启发式算法分为Seed Model Selection和Mutation Rule Selection。Lemon首先选择出一个种子来变异，然后基于该种子选择一个变异方法来运用。

## Seed Model Selection

一个种子模型，如果之前被选来进行变异的次数较少，那么就提高它被选中的几率。$c_i$表示种子被选中的几率。它与$score_i$成反比，可见如果一个被选择的次数越少，它的得分就越高，因此被选中的几率就越高。

$$score_i = \frac{1}{c_i + 1}$$

接着将得分转为为概率,计算每个种子被选中的概率。（Roulette Wheel Selection方法）在每次迭代中，都按照概率的大小来选择模型（**注意，这里的模型包含了已经变异过的模型，以变异的模型会再次加入种子池当中来**）。

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

## Mutation Rule Selection

采用MH算法来指导测试规则选择。MH是在基于当前的变异规则的基础上，从一个概率分布中获取随机样本。

$$Pa(MU_b|MU_a) = \frac{Ps(MU_b)}{Ps(MU_a)} = (1-p)^{k_b - k_a}$$

$MU_b$和$MU_a$分别表示选择变异规则a和b，$k_a$和$k_b$分别表示规则a和规则b在列表中的排名（该列表排名的顺序是一个规则被选中的次数）。P是作者给定的一个常数。然后比较变异后的模型是否比未变异的模型更能体现出差异性。

那么如何判断它是否成功提高不一致性 （ACC算法）？

$$\delta_{O,G} = \frac{1}{m}\sum_{i=1}^{m}|o_i - g_i|$$

$$D\_MAD_{G,O_j,O_k} = \frac{|\delta_{O_j,G} - \delta_{O_k,G}|}{\delta_{O_j,G} + \delta_{O_k,G}}$$

$$ACC(M) = \sum_{i=1}^{n}\sum_{j,k=1}^{m} D\_MAD_{G,O_{ji},O_{ki}}$$

G表示基准向量，一共有m个，$o_j$和$o_k$表示预测的输出向量,n表示输入的模型数量，m表示深度学习库的数量

$ACC(M), ACC(C)$分别表示模型变异后的不一致性和模型变异前的不一致性如果$ACC(M) > ACC(C)$,则表示成功扩大了不一致性。

下面是作者提供的启发式算法的一段伪代码，我对他进行了一下复现

**该算法的输入：变异规则列表、种子模型列表、需要产生的模型数量**

**算法输出：变异后的模型数量**

```
MU_a ← random(Rules)
while Size(Models) < N do
    foreach i from 1 to Size(Seeds) do
        Pro[i]←calProb(Seeds[i])/* calculate the probability for
           Seeds[i] by Formula 5                                   */
    end
    r ← random(0,1)
    bound ← 0
    foreach i from 1 to Size(Seeds) do
        bound ← bound + Pro[i]
        if r ≤ bound then
            s ← Seeds[i]/* s is the selected seed model            */
            break
        end
    end
    k_a ← position(MU_a)
    do
        MR_b ← random(Rules)
        k_b ← position(MR_b)
        f ← random(0,1)
    while f ≥ (1 − p)^(k_b − k_a);
    m ← Mutate(s, MU_b)
    Models ← Models ∪ {m}
    if ACC(m)≥ ACC(s) then
        /* ACC is defined in Section 3.2.1 to calculate the
           accumulated inconsistent degrees                        */
        Seeds ← Seeds ∪ {m}
    end
    updateScore(s)/* update the score of s defined in Formula 4    */
    updateRatio(MU_b)/* update the priority score of MU_b defined in
       Section 3.2.2                                               */
    Rules ← sort(Rules)
    MU_a ← MU_b
end
return Models
```

```python
rules = []  # 将所以的规则放在一个列表里面
seeds = []  # 将所有的模型放在一个列表里面
models = []  # 将所有的结果模型放在一个列表里面
pro = {}  # 每个模型出现的概率 格式为字典 model:概率
times = {}  # 每个模型出现的次数 格式为字典 model：次数
position = []  # 用来记录每个rule的排名
p=0.3


# 由模型出现的次数计算的它的得分
def cal_score(model):
    return 1 / times[model] + 1


# 根据模型的得分计算出该model被选中的概率
def cal_prob(model):
    cnt = 0
    for i in range(len(seeds)):
        cnt = cnt + cal_score(seeds[i])
    return cal_score(model) / cnt

# 模型+规则进行变异操作,这里需要用到算法1
def mutate(model,rule):
    return model
```

```
def heuristic_alg(rules,seeds,n):
    global k_b, k_a, s
    mu_a = random.choice(rules)

    # 执行Roulette Wheel Selection
    while len(models) < n:  #n 需要的模型数量
        for i in range(len(seeds)):
            pro[seeds[i]] = cal_prob(seeds[i])
        r = random.random()  # 随机（0-1）之间的小数
        bound = 0
        for i in range(len(models)):
            bound = bound + pro[seeds[i]]
            if r < bound:
                s = seeds[i]
                break
        # s是选择出来的seed 下面对rule进行选择
        # selects the next mutation rule MU_b based on the current one MU_a
        for i in range(len(position)):
            if position[i] == mu_a:
                k_a = i                # 找到a的排名


        while True:
            mu_b = random.choice(rules)
            for i in range(len(position)):
                if position[i] == mu_b:
                    k_b = i  # 找到b的排名
            f = random.random()  # 随机（0-1）之间的小数
            # 利用概率分布
            if f > pow(1-p, k_b-k_a):
                continue
            else:
                break
        m = mutate(s, mu_b)
        if m not in models:
            models.append(m)
        if acc(m) >= acc(s):
            if m not in seeds:
                seeds.append(m)
        rules.sort() #重新进行排序，进入下一次迭代
        mu_a=mu_b
    return models
```

**注：图片中画黄线部分应该是作者的笔误，正与他进行联系。$MR_b$应该改成$MU_b$**

# 分析与评价

## LEMON的有效性

**Table 3: The number of new bugs detected by LEMON**

| 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 |

* IC is short for inconsistency and IC Bugs refer to the bugs analyzed from inconsistencies. The last cell refers to the total number of new bugs detected by LEMON on all the five libraries.

LEMON在最新发布的深度学习库中检测出了24个BUG。其中13个是分析不一致性得出的BUG，6个是崩溃的BUG，4个非数据错误和一个表现错误

**Table 4: Detailed comparison results in terms of the number of detected bugs**

| ID | Lib | AlexNet | | | DenseNet | | | Inception | | | $LeNet5_F$ | | | $LeNet5_M$ | | | LSTM-1 | | | LSTM-2 | | | MobileNet | | | ResNet5 | | | VGG16 | | | VGG19 | | | Xception | | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I | L | M | I |
| $E_1$ | TF | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 5 | 2 | 0 |
| | TH | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 2 | 0 |
| | CN | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| | MX | 3 | 2 | 0 | 4 | 3 | 0 | 2 | 1 | 0 | 4 | 3 | 0 | 3 | 3 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 4 | 3 | 0 | 4 | 2 | 0 | 1 | 1 | 0 | 2 | 2 | 0 | 3 | 3 | 0 | 12 | 7 | 0 |
| $E_2$ | TF | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 6 | 2 | 0 |
| | TH | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 6 | 4 | 0 |
| | CN | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 |
| | MX | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 2 | 2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 4 | 2 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 2 | 2 | 0 | 10 | 5 | 0 |
| $E_3$ | TF | 3 | 2 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 4 | 0 | 0 |
| | TH | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 5 | 3 | 0 |
| | CN | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 |
| | MX | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 3 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 8 | 3 | 0 |
| $E_4$ | TF | 3 | 2 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | 0 |
| | TH | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 |
| | CN | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 |
| | MX | 2 | 1 | 0 | 3 | 2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 3 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 8 | 4 | 0 |
| $E_5$ | TF | 2 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | 0 |
| | TH | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 6 | 4 | 0 |
| | CN | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 3 | 2 | 0 |
| | MX | 1 | 0 | 0 | 3 | 2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 9 | 5 | 0 |

* Column "L" presents the number of detected bugs by LEMON, Column "M" presents the number of unique bugs that are detected by the mutated models but are not detected by the initial model, and Column "I" presents the number of unique bugs that are detected by the initial model but are not detected by the mutated models. $LeNet5_F$ refers to the LeNet5 model based on the Fashion-MNIST input set, $LeNet5_M$ refers to the LeNet5 model based on the MNIST input set, DenseNet refers to DenseNet121, MobileNet refers to MobileNetV1, and Inception refers to InceptionV3.

LEMON对每个单个的数据集进行了差分测试。对每个模型和深度学习库也都进行了分别的测试。在5个实验中，LEMON平均检测到了22个错误。

**Table 5: Average inconsistent degree comparison for the detected inconsistencies**

| 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% |

* Columns "$V_M$" and "$V_I$" present the average inconsistent degrees achieved by the mutated models and the initial models across all the used models, respectively. Column "$\Uparrow_{rate}$" presents the average improved rates of inconsistent degrees achieved by the mutated models over the initial models.

Lemon对所有的深度学习库进行了差分测试，实验表明变异后的模型能有显著增大不一致度，从而解决了DL testing的第二个关切问题

更多了解可参考PPT和视频展示。