DeepMutation: Mutation Testing of Deep Learning Systems

Master Thesis

Abdulhayyu H. Lawal

Universität Passau
Chair for Software Engineering II
Prof. Dr. Gordon Fraser

Supervisor: Dr. Alessio Gambi

December 2019

Outline

- Introduction
  - The Problem
  - Why Testing?
  - Mutation Testing of Deep Learning Models (DL)
- Related Work
  - Replication Study ⇒ Lei Ma et al. 2018
  - Thesis Aims
  - Why Repeat?
- Implementation
- Evaluation
  - Quantitative Analysis
  - Qualitative Analysis
- Schedule
- Conclusion
- Reference

Introduction(1)

- The Problem (Deep Learning Models)

- Important concept in many areas such as self-driving vehicles
- Tremendous success
- However, robustness and quality are major concern
- Defective DL may lead to catastrophic accidents, consequences
  - Uber autonomous car accident in March 2018
    - ⟶ Led to a pedestrian death

Source: https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe#img-2

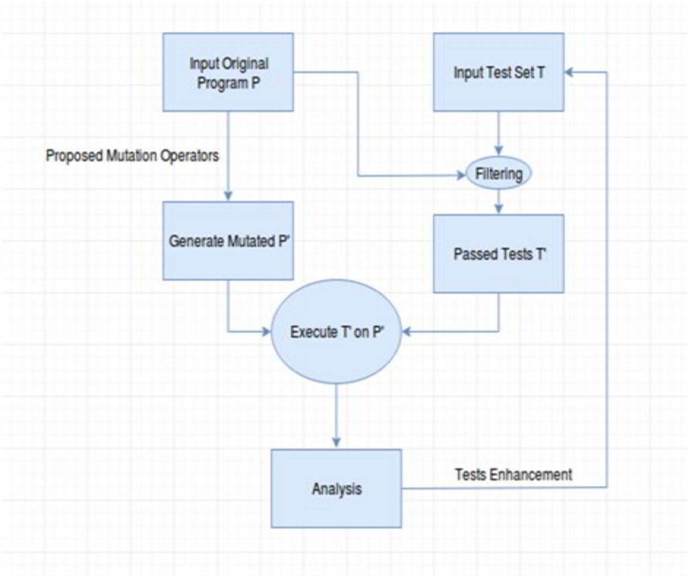
Introduction(2)

- Why do we need to test?:

- Point out defects, errors (Reliable functioning)
- Evaluate quality performance (Robustness)
- Improve safety and accuracy to avoid failure (Accident)

- Mutation Testing(MT) ➡ The Solution:

- Fault-based technique for quality evaluation of test suites
- Mutants to create mutated copy
- Execute test suites
- Mutation score = $\frac{M'_{Killed}}{m' \epsilon M'_{All}}$
  - Ratio of #killed_mutants against total #all_genereated_mutants
- Evaluate test suite quality
- Examine model robustness



Related Work(1)

- Replication Study ➡ Lei Ma et at. DeepMutation Paper

- Many research to turn MT into practice
- But, very few research specifically apply MT to test DL systems

- Thesis aims at:

- Replicating Lei Ma et al. approach to determine a CNN robustness
- Understanding findings & contribution
- Analyzing mutation operators
  - Mutant contribution i.e better or least impact?
  - Improve test data quality and examine model robustness
  - Reduce mutants behavioral and equivalent problem
- Providing quantitative and qualitative analysis
  - Test evaluation, Why replication, why mutants analysis, report

Related Work(2)

- Why re-applying Lei Ma et al. approach?

- framework specialize for testing DL systems
- Analyze findings and contribution
- Further improvement and demonstration of result
- Pathway ➡ Analyze mutation operators relations
  - Investigate mutants impact and relations?
  - Analyze costs
  - Reduce too many behavioral difference
  - Reduce equivalent problem
    - Different mutants producing same result

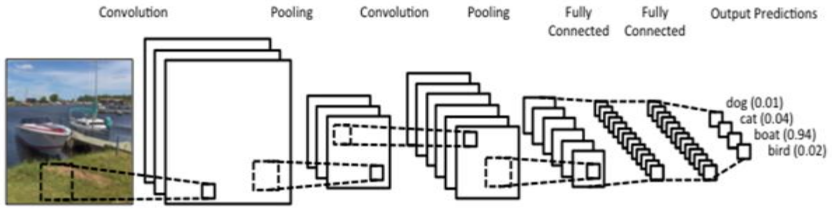
Implementation(1)

- Dataset

- Either MNIST, CIFAR-10 or both – 10% as test data
- Experimental control to ensure dataset quality

- DL Model

- Deep Neural Network – Convolutional Neural Network (CNN)
  - Recognize visual patterns with minimal preprocessing.
  - LeNet 5 suitable for handwritten and machine-printed character recognition



Source: http://deeplearning.net/tutorial/lenet.html

Implementation(2)

- *Proposed Mutants Operators*

- Deliberately introduced to slight modify program
- Traditional MT not possible due to fundamental architecture for DL
- Different experiment with different proposed mutation operators
  - Generate mutants using DeepMutation++ framework tool

- Source-Level Mutation Operators

Fault/Mutant Type
Data Duplication (DD)
Data Property Manipulation (DPM)
Data Outliers Introduction (DOI)
Data Shuffle (DF)
Noise Pertubation (NP)

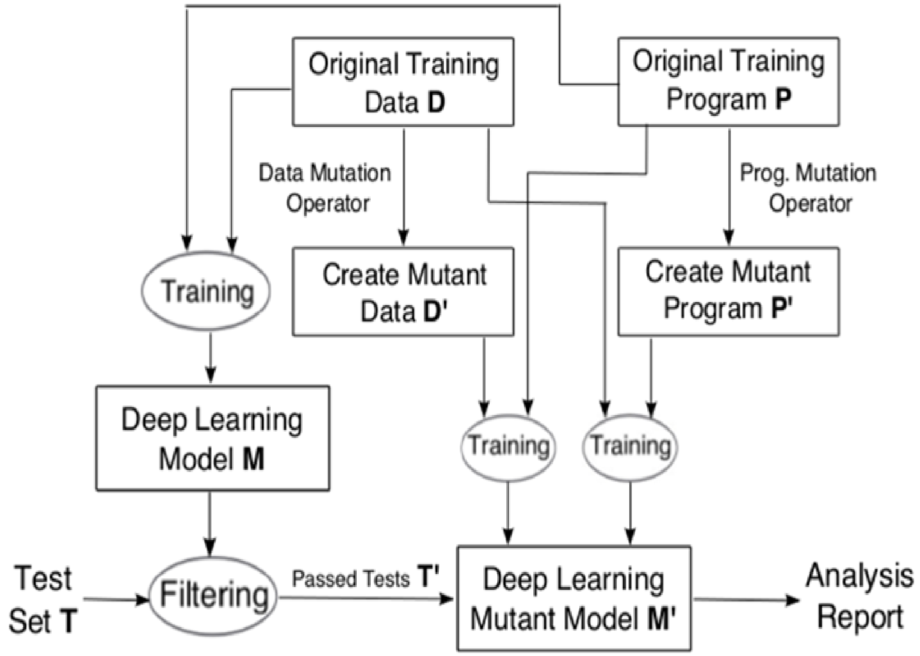
- Model-level Mutation Operators

Mutation Operator
Activation Function Change (AFC)
Weight Shuffling (WS)
Neuron Effect Block. (NEB)
Neuron Activation Inverse (NAI)
Neuron Switch (NS)
Layer Deactivation (LD)

Implementation(3)

- Setup

- Based on Keras 2.3.2 with Tensorflow 2.0.0 backend
- Python 3 implementation environment
- Jupyter notebooks
- Manual and automated mutants generation
  - DeepMutation++ framework tool, Mutpy



Source: Lei Ma et.al 2018

Mutation testing Example:

```
#Program
def EditNum(x, y):
    return x + y

#Test Case
from unittest import TestCase
from example import EditNum

class exampleTest(TestCase):

    def test_num(self):
        self.assertEqual(EditNum(4, 2), 6)
```

```
#Mutating (changing + to -)
def EditNum(x, y):
    return x - y
```

- Running the same test case, the T` will fail

Implementation(4)

- Testing Process (Model)

- Mutating dataset involves manipulating data
  - Editing pixels, properties, dithering …

- Mutating model program ⇒ data *D* & program *P* to *D*` & *P*` mutated copy

  Example,
  - Changing activation function
  - Each neuron has weight (W), bias (B), activation function (Af), network structure (Ns)
  - Modification changes output and affects next layer
  - f(x)' = f(y)', *T* ' is executed to catch $OP_{actfuncw}$
  - If f(y) != f(y)', $OP_{actfuncw}$ is feasible and is killed

Evaluation

- Quantitative Analysis

- Interpret general MT evaluation with statistical inference
- Address precision problem ➡ *T'* size is large enough
  - Enhance metric to focus on the classification problem

$$MutationScore(T', M') = \frac{\sum m' \epsilon M' |KilledClasses(T', m')|}{|M'| \times |C|}$$

- Solve large behavioral difference between original and mutated data/program
  - Average error rate

$$AER(T', M') = \frac{\sum m' \epsilon M' |ErrorRate(T', m')|}{|M'|}$$

- Other evaluation metric (Test accuracy)
  - F1 Score $F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}$

Precision: Correct positive(killed) / total predicted positive observations(all introduced mutant)
Recall: Correct positive (killed) / total actual positive(total introduced mutant)

Evaluation

- Qualitative Analysis

- Elaborate MT understanding and usefulness
- Report analysis indicating test data quality and model's robustness

- Analyzing mutant impact and relations

- Generate based on pre-defined rules
- Specific part, only that part is analyzed
- Classify each mutant
  - Mutant_killed, not killed(not feasible) and stubborn(more test)
- Obtain ratio of impact
  - Evaluate result

Schedule

- Time Schedule

Phase	Weeks
Initiation, proposal and presentation	-
Prepare data and DL models	2
Design and introduce mutation operators	4
Execute test (Train)	4
Evaluation (Using Metrics)	4
Investigate mutants impact	4
Analyse, report and documentation	6

Conclusion

- Close implementation similar to Lei Ma et.al paper
  - But, different experiments
    - Different mutation operators (Faults)
    - DeepMutation++ framework tool
- Evaluate
  - Test cases and examine model robustness on test dataset
- Mutant impact
  - Generate mutants using DeepMutation++ tool
  - Reduce behavioral diff and equivalent problem
- Demonstrate MT usefulness for DL systems
- Other contribution of the thesis
  - Comprehensive study on mutation operators relations and costs

Thank you for your intention!

References

- Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu,Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. Deepmutation: Mutation testing of deep learning systems. In 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), pages 100{111. IEEE,2018.

- Lorena Gutierrez-Madronal, Jose J Domnguez-Jimenez, and Inmaculada Medina-Bulo. Mutation testing: Guideline and mutation operator classication. In The Ninth International Multi-Conference on Computing in the Global Information Technology, pages 171{179, 2014.

- Weijun Shen, Jun Wan, and Zhenyu Chen. Munn: Mutation analysis of neural networks. In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pages 108{115. IEEE, 2018.