DeepMutation: Improving Mutation Testing of Deep Neural Networks

Master Thesis

Abdulhayyu H. Lawal

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

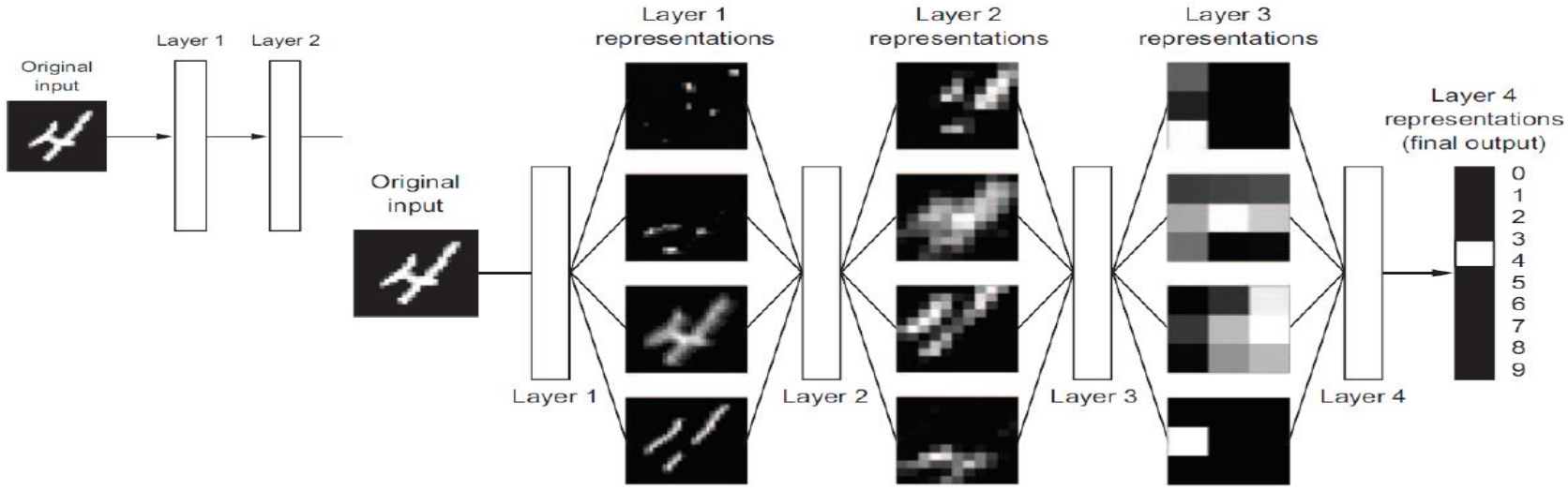
Supervisor: Dr. Alessio Gambi

04 December 2019

Introduction

Deep Learning

- Important concept in many application domains such as self-driving vehicles, image recognition, speech recognition…
- Network learns more when there is more training examples. Hence, improving accuracy



Prof. Dr. Ralf Krestal, Uni-Passau

The Problem

- Tremendous success, but
- Robustness and quality remain a major concern
- Present a big challenge more especially in safety-critical systems
- Defective DL may lead to catastrophic accidents, consequences
- Can easily be fooled with adversarial samples

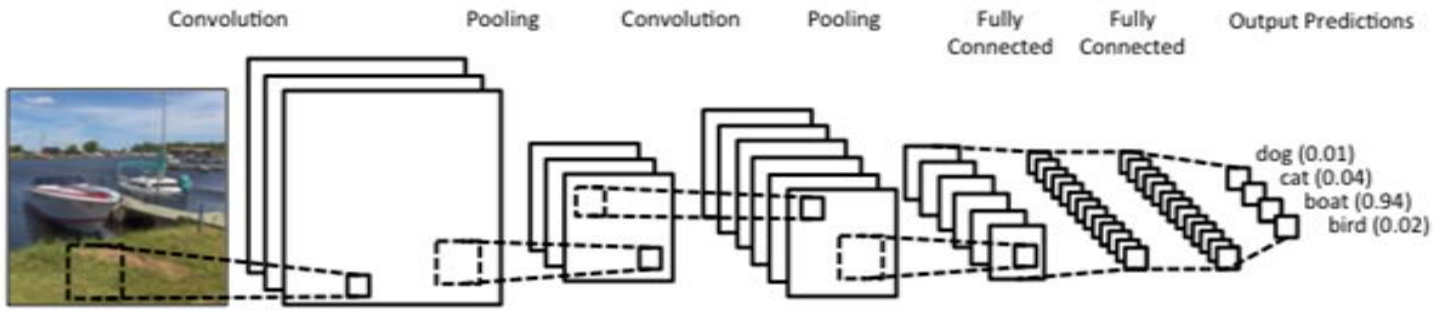
  - Uber autonomous car accident in March 2018
    - Led to a pedestrian death

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

Deep Neural Networks (DNNs)

Convolutional Neural Networks (CNNs)

- Multi-hidden layer neural network architecture, well-adapted to classify images
  - Uses examples to automatically infer rules for recognizing handwritten digits
- Fully connected-layers, each trains on a distinct set of features based on previous layer's output
- The deeper the layer, the more accuracy
  - LeNet 5 architecture is considered to be used,
    - Suitable for handwritten and machine-printed character recognition

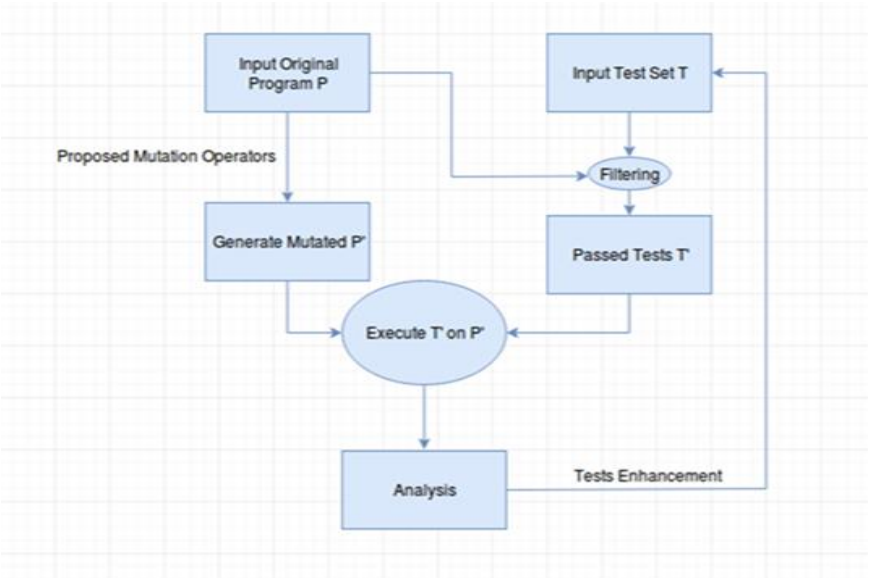


http://deeplearning.net/tutorial/lenet.html

Why Mutation Testing of DL?

- Why Testing in general?
  - Point out defects, errors (Reliable functioning), robustness & accuracy
- DNNs are known to be vulnerable to adversarial sample
  - Can be improved through mutation testing by examining their performance on test dataset
    - a fault-based technique for quality evaluation of test suites
    - mutation operators to create mutated copy of original
    - execute test suites to catch and kill introduced mutants
    - mutation score = $\frac{M'_{Killed}}{m' \epsilon M'_{All}}$

      ratio of #killed_mutants against total #all_generated_mutants
    - evaluate test suites and determine test dataset quality
    - examine test dataset to gain confidence of the trained model




```
#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

Start-of-the for testing NN

- Multiple approaches to improve robustness of the DNN models
- Many research to turn MT into practice
- But, very few research specifically and directly apply MT to DL systems
  - Lei Ma et.al 2018, first well-known approach
- Training cost and equivalent mutants problem are major limitations
- Though, MT improves the DNN robustness to some extent by examining performance on test data quality
- More adequate MT on the dataset is required to increase robustness to max
  - Test to cover more aspects and their impact analyzed (Thesis focus)

Motivation

- Deep high-performing DNNs can be fooled by intentionally constructed samples
  - Adversarial samples or mutants, exposing DNNs lack of robustness
- Improvement of DL models need to be increased
- A test like MT can be very useful
- This thesis will apply MT on CNN adapting Lei et.al approach to improve the testing of DNNs
  - Proposing versatile mutation operators to cover more diverse aspects
  - Investigating the relations of the operators and their impacts
  - Further demonstrating the usefulness of the technique

⟶ More diverse mutant and test coverage will enhance the evaluation

Why Lei et.al 2018 Approach?

- Due to DL architectural designs, traditional testing technique cannot be applied directly
- Lei Ma et. al 2018 proposed a framework suitable for DL
  - One of very few approach to inject mutants to NN
- Designed source and model-level mutation operators directly
- Repeating the approach will provide;
  - an efficient way to apply MT on DL systems
  - a path to investigate further more on mutation operators coverage and their impact
- Double check findings of the authors

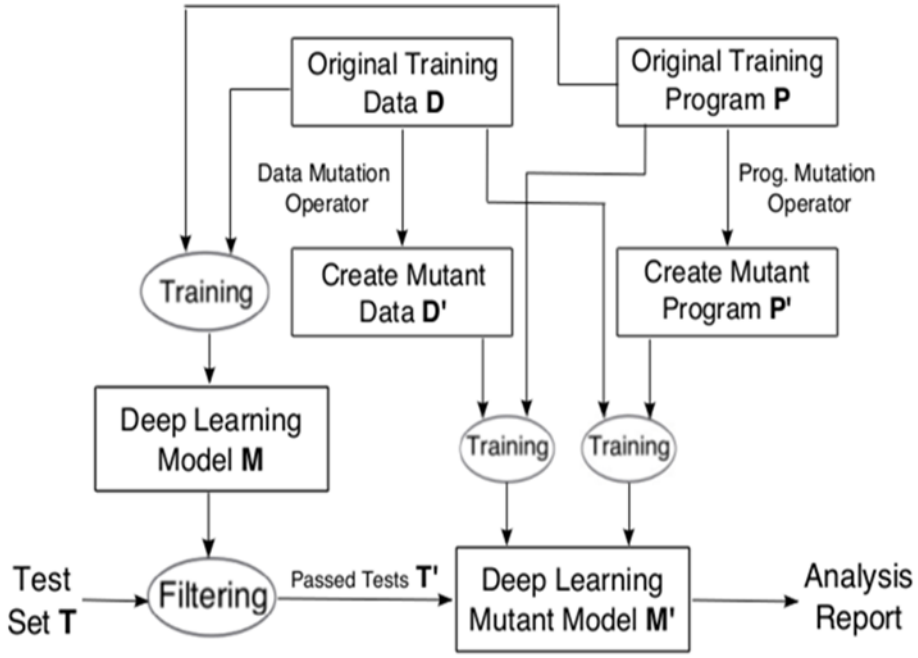
Implementation

- *Dataset:*
  - Either MNIST, – 10% as test data
- *Proposed mutation operators:*

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

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

Manuel and automatic generation using mutants using DeepMutation++ framework tool. Mutpy



Lei et.al 2018

- *Setup:*
  - Based on Keras 2.3.2 with Tensorflow 2.0.0 backend
  - Python 3

- If f(x) = f(y)', *T'* is executed to catch OPactfuncw
- If f(y) != f(y)', OPactfuncw is feasible and is killed

How good are the tests?

- *Improving the tests:*
  - Extend coverage in the program
    - covering diverse aspects of the target parameters, area and relations
    - mutants impact by analyzing their coverage and class
      - killed, stubborn or alive
  - Strong mutation operators for both levels
- Enhance test coverage to cover more parts in the program and evaluate test
  - Test coverage = test_covered_program / all_mutated_program *100%
- Covering methods may include;
  - Sign-Sign , Value-Sign, Sign-Value, and Value-Value coverage

Evaluation Analysis

Qualitative:

- Interpret general MT evaluation with statistical inference
- Address precision problem when *T'* size is large enough
  - Enhance metric to focus on the classification pro $MutationScore(T', M') = \frac{\sum m' \epsilon M' |KilledClasses(T', m')|}{|M'| \times |C|}$
- Solve large behavioral difference between original and mutated data/program
  - Average error ra $AER(T', M') = \frac{\sum m' \epsilon M' |ErrorRate(T', m')|}{|M'|}$
- Other evaluation metric (Test accuracy)
  - F I S $F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}}$

Qualitative:

- Elaborate MT understanding and usefulness
- Report analysis indicating improvement of the test data quality and model's robustness after examining

Conclusion/Contribution

- Close implementation similar to Lei Ma et.al 2018 approach
  - But, different test subject
    - Different mutation operators (Faults)
- Mutation operators to cover more diverse aspects
  - More mutants and test coverage
- Evaluate
  - Test cases and examine model performance on test dataset to gain confidence
- Demonstrate MT usefulness for DL systems
- Other contributions of the thesis
  - A study on cost of mutation testing and equivalent mutant problem (May not be part)

Thank you for your Attention!

Feedback,

Recommendations,