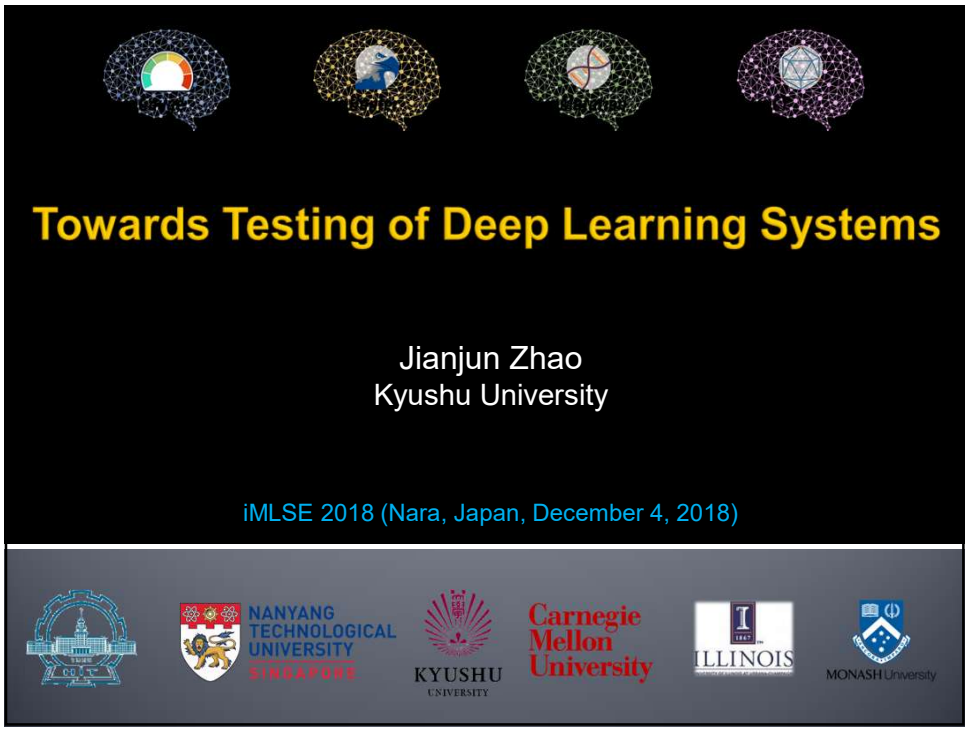


Pangu Research Group (Kyushu Univ.)

（知能ソフトウェア工学研究室）

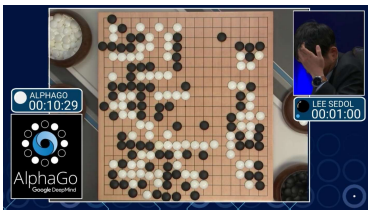
(https://pangukaitian.github.io/pangu/en/index.html)

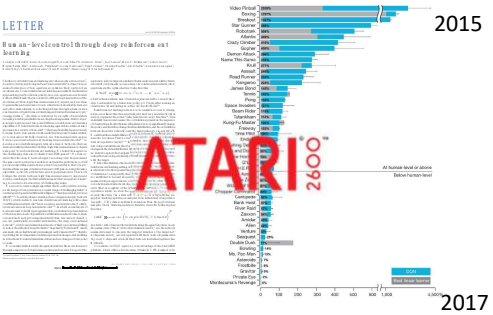
- On-going work
  - researches on the potential symbioses between software engineering SE and artificial intelligence AI
- The overall goal
  - to obtain better software and AI systems making them more robust, reliable, and secure, and easier to specify, build, maintain, or improve
- Group members
  - 2+ faculties and 3 PhD and 7 MS students

2

Deep Learning Matches Human Intelligence

AlphaGo 4:1 Human Champion 2016





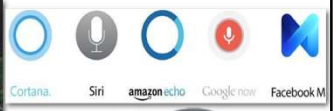
2015

AlphaGo ZERO 100:0 AlphaGo 2017

2017

3

Wide Deployment in Real World



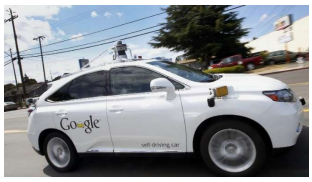
4

Driving Force of Many Novel Technologies

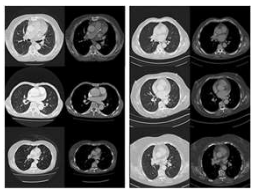
5

Deep learning is increasingly used in safety-critical systems

- **Deep learning reliability and safety is crucial**



Self-driving car



Medical diagnosis

Robotics

Malware detection

6

Problems and Critiques for DL Systems

- Un-safe, e.g., lack of robustness (reliability and safety)
- Hard to explain to human users (interpretability)
  - Deep neural networks are essentially black-boxes and researchers have a hard time to understand how they deduce conclusions
- Fairness, accountability, ethics, trustworthiness, etc.
  - What would human review entail if models were available for direct inspection?

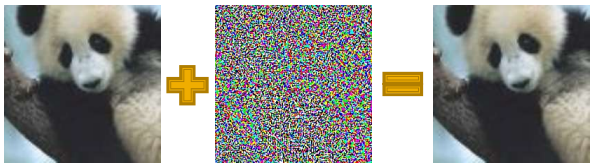
7

Problems and Critiques for DL Systems

- Un-safe, e.g., lack of robustness (reliability and safety)
- Hard to explain to human users (interpretability)
  - Deep neural networks are essentially black-boxes and researchers have a hard time to understand how they deduce conclusions
- Fairness, accountability, ethics, trustworthiness, etc.
  - What would human review entail if models were available for direct inspection?

8

Current Deep Learning is Vulnerable



Classified as panda Small adversarial noise Classified as gibbon

Ian Goodfellow, Jon Shlens, Christian Szegedy, Explaining and Harnessing Adversarial Examples, ICLR, 2014

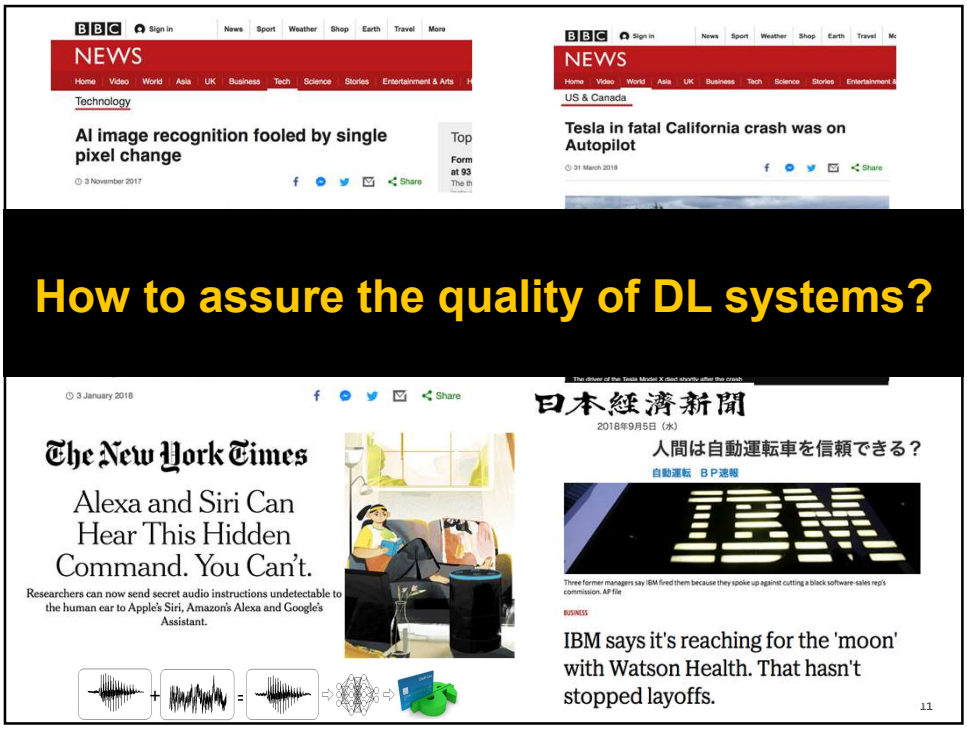
Accident

9

Unreliable Deep Learning

Tesla autopilot failed to recognize a white truck against bright sky leading to fatal crash

10



To design reliable systems, engineers typically engage in both testing and verification

- By testing: we mean evaluating the system in several conditions and observing its behavior, watching for defects.
- By verification: we mean producing a compelling argument that the system will not misbehave under a very broad range of circumstances.

* Ian Goodfellow and Nicolas Papernot. 2017. The Challenge of Verification and Testing of Machine Learning.

12

To design reliable systems, engineers typically engage in both testing and verification

- By testing: we mean evaluating the system in several conditions and observing its behavior, watching for defects. (this talk focuses on testing)
- By verification: we mean producing a compelling argument that the system will not misbehave under a very broad range of circumstances.

* Ian Goodfellow and Nicolas Papernot. 2017. The Challenge of Verification and Testing of Machine Learning.

13

Testing Issues for DL Systems

- Test coverage criterion
  - How to define the test coverage criteria of DL systems?
- Test data generation
  - How to automatically generate a mass of test data for DL systems?
- Test data quality
  - How to evaluate the quality of test data for DL systems?

14

Testing Issues for DL Systems

- Test coverage criterion
  - How to define the test coverage criteria of DL systems?
- Test data generation
  - How to automatically generate a mass of test data for DL systems?
- Test data quality
  - How to evaluate the quality of test data for DL systems?

15

Quality Assurance for Traditional Software
Testing Criteria and Tools

- **Line Coverage**
- **Branch Coverage**
- **Function Coverage**
- **Data Flow Coverage**
- **Combinatorial Coverage**
- **Mutation testing Coverage**

16

Quality Assurance for DL is at Early Stage

SOSP'17

DeepXplore

ICSE'18

DeepTest

ASE'18

Fairness

DeepConcolic

DeepGauge

DeepRoad

ISSTA'18

TensorFlow Program Bugs

FSE'18

MODE: DL Debugging

17

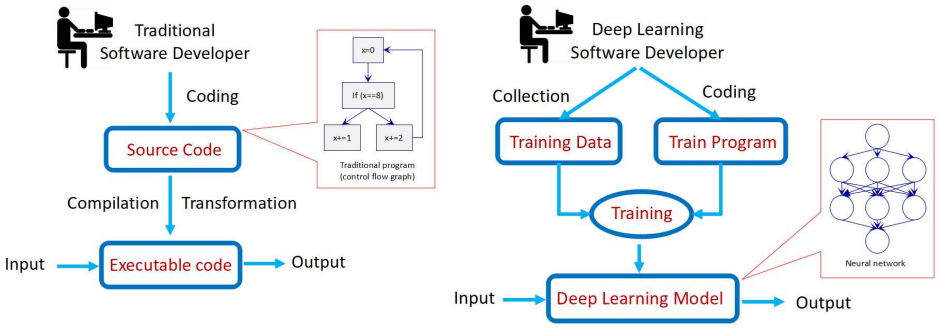
Traditional Software

SQA

Deep Learning Software

Fundamental Different Programming Paradigm

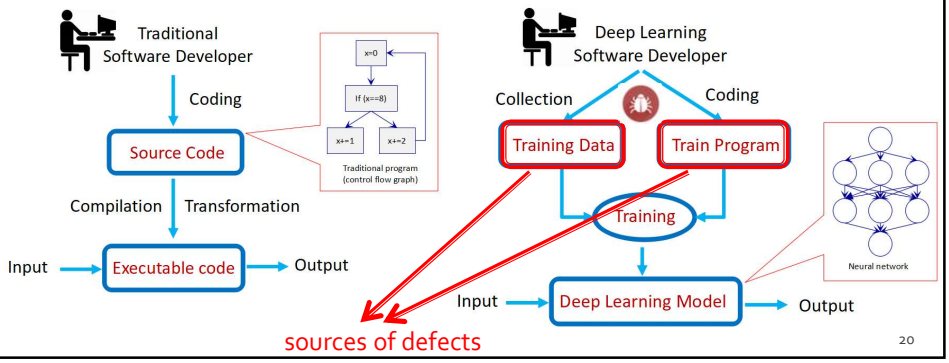
- **The decision logic of a traditional software:**
  - In the form of code
- **The decision logic of a DL system:**
  - The structure of DNN
  - The connection weights



19

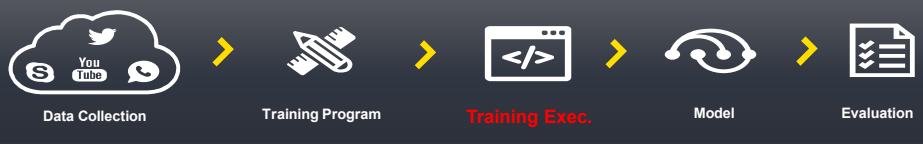
Source of defects: programming paradigm

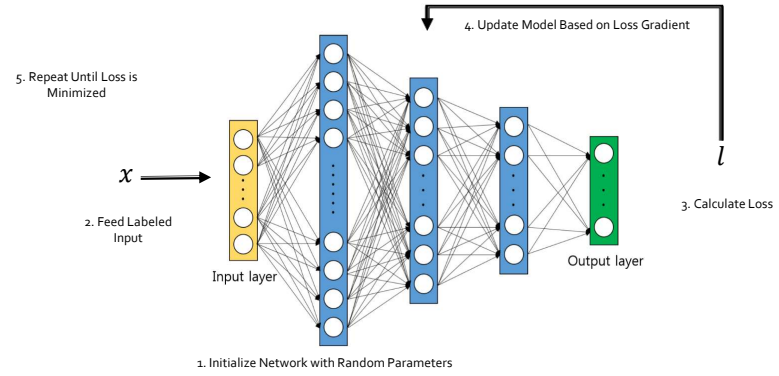
- **The decision logic of a traditional software:**
  - In the form of code
- **The decision logic of a DL system:**
  - The structure of DNN
  - The connection weights



20

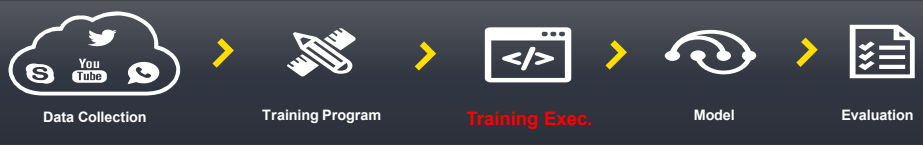
DL Software Decision Logic Learning

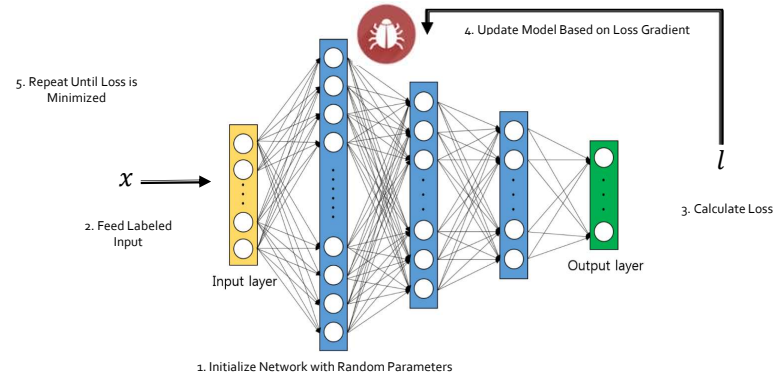




21

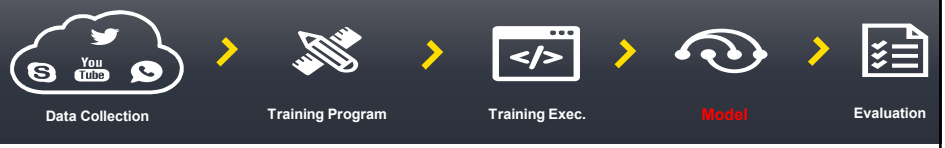
Source of defects: Training Procedure





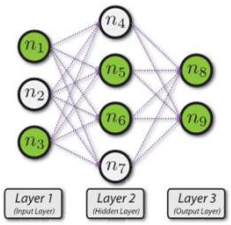
22

Lacking of Interpretability and Understandability



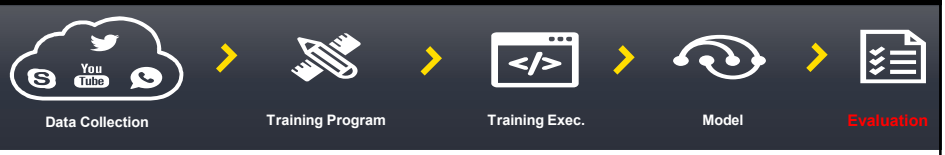
Behavior of the DL system?

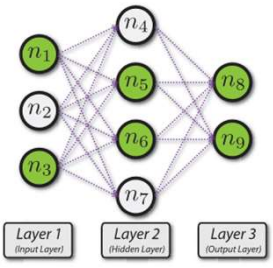
Neurons → Connection Strength



23

Quality Measurement Immature



Test data set →  → Test Accuracy

High accuracy ⇸ High DL quality

24

Towards Quality Assurance for DL Systems

- **Multi-granularity testing criteria for DL systems (ASE 2018)**
  - ACM SIGSOFT Distinguished Paper Award

- **Coverage-guided fuzzing testing framework**

25

Towards Quality Assurance for DL Systems

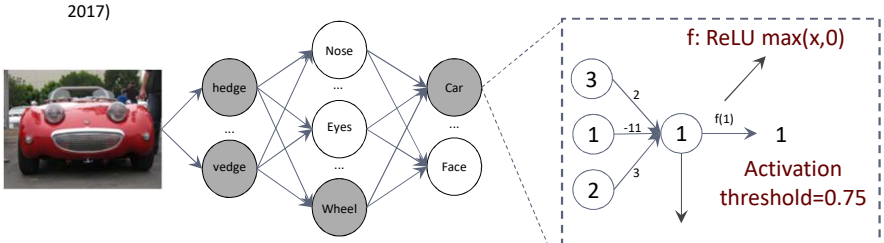
- **Multi-granularity testing criteria for DL systems (ASE 2018)**
  - ACM SIGSOFT Distinguished Paper Award

- **Coverage-guided fuzzing testing framework**

26

Test coverage criteria for DL systems (1)

- Neuron coverage: how much decision logic exercised
  - Neuron coverage = # neurons activated / # total neurons
- Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana. "DeepXplore: Automated Whitebox Testing of Deep Learning Systems", in Proceedings of the 26th ACM Symposium on Operating Systems Principles (SOSP 2017)



Neuron coverage: 4/7=57%

27

Problems for Neuron Coverage

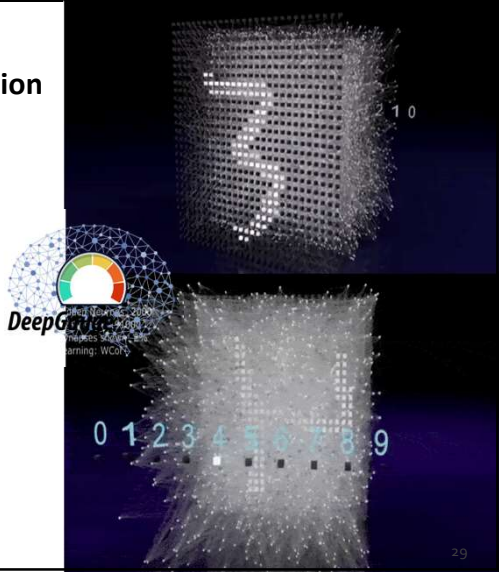
- Neuron coverage uses the same threshold as the activation evaluation for all the neurons
- It is straightforward to obtain a trivial test suite that has high neuron coverage but does not provide any adversarial example

*Testing Deep Neural Networks. Youcheng Sun, Xiaowei Huang, Daniel Kroening. arXiv:1803.04792, 2017

28

Overview of DeepGauge

- **Enable** testing quality evaluation of DL systems from multiple portrayals
- **Provide** systematic guidance of test generation for detecting defects
- **Facilitate** interpretation & understanding



Design of DeepGauge

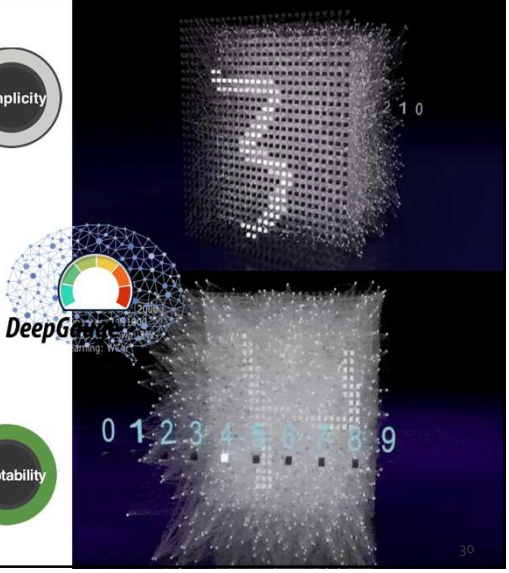
Simple to understand & use

Efficient to compute

General to diverse DNNs

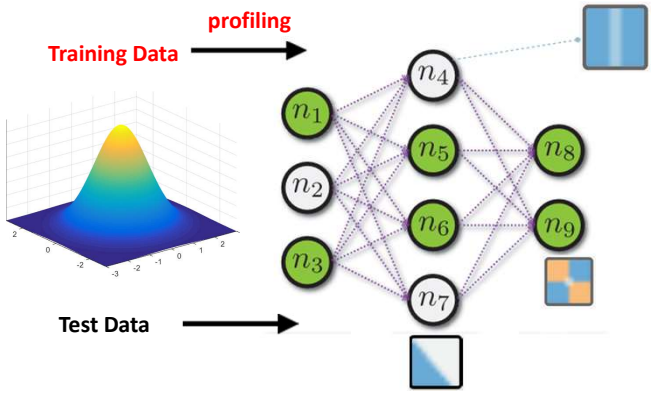
Scale to large DNNs

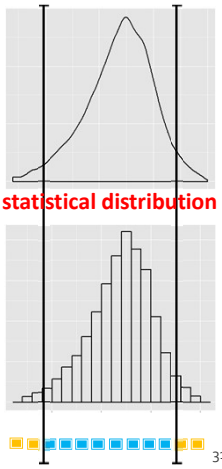
Adaptable by cases



Neuron-level Criteria

We use the output values of neuron *n* determined from the training to characterize its behaviors

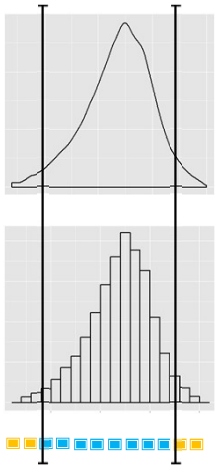




31

Neuron-level Criteria

Large Scale Test Data → Neurons



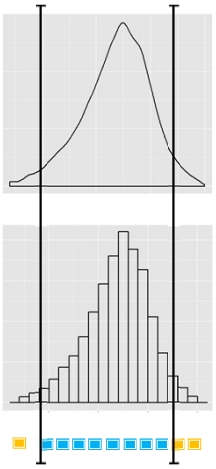
- **K-Multisection Neuron Coverage (KMNC)**
  - measures how thoroughly the given set of test inputs T covers the major function region [low$_n$, high$_n$] of a neuron n.

$$\mathrm{KMNCov}(T,k) = \frac{\sum_{n\in N} |\{S_i^n \mid \exists \mathbf{x} \in T : \ \phi(\mathbf{x},n) \in S_i^n\}|}{k \times |N|}$$

32

Neuron-level Criteria

Large Scale Test Data → Neurons 



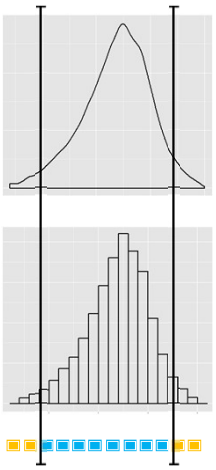
- **Neuron Boundary Coverage (NBC)**
  - measures how many corner-case regions (w.r.t. both of the upper and lower boundary values) have been covered by the given test input set T.

$$\text{NBCov}(T) = \frac{|\text{UpperCornerNeuron}| + |\text{LowerCornerNeuron}|}{2 \times |N|}$$

33

Neuron-level Criteria

Large Scale Test Data → Neurons 



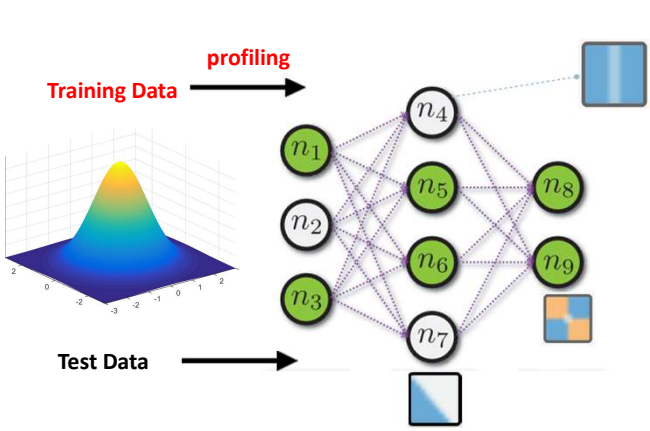
- **Strong Neuron Activation Coverage (SNAC)**
  - measures how many corner cases (w.r.t. the upper boundary value) have been covered by the given test inputs T.

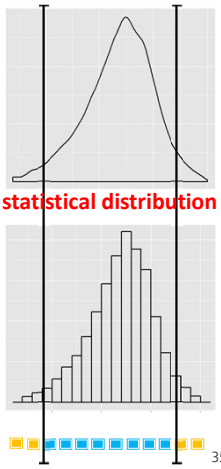
$$\text{SNACov}(T) = \frac{|\text{UpperCornerNeuron}|}{|N|}$$

34

Layer-level Criteria

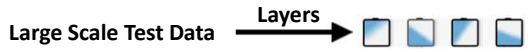
We use the top hyperactive neurons and their combinations (or the sequences) to characterize the behaviors of a DNN



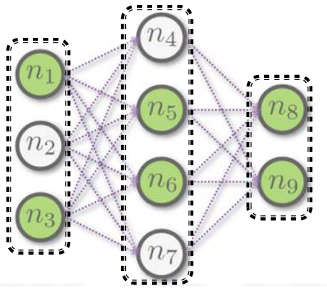


35

Layer-level Criteria



- **Top-k Neuron Coverage (TKNC)**
  - measures how many neurons have once been the most active k neurons on each layer.



$$\mathrm{TKNCov}(T,k) = \frac{|\bigcup_{\mathbf{x}\in T}(\bigcup_{1\le i\le l}\mathrm{top}_k(\mathbf{x},i))|}{|N|}$$

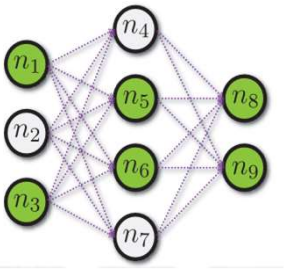
Neuron interactions of same layer

36

Layer-level Criteria

Large Scale Test Data → Layers

- **Top-k Neuron Patterns (TKNP)**
  - Given a test input x, the sequence of the top-k neurons on each layer forms a pattern



$$\mathrm{TKNPat}(T,k) = |\{(\mathrm{top}_k(\mathbf{x},1),\ldots,\mathrm{top}_k(\mathbf{x},l)) \mid \mathbf{x} \in T\}|$$

Neuron interactions across layers

37

Large Scale Empirical Study

MNIST

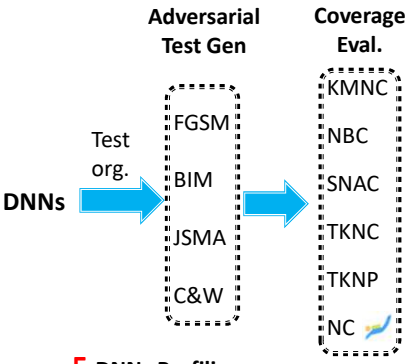
60,000
10,000
(28,28,1)
784 dim.

IMAGENET
(LSVRC-2012)
1,000,000+
50,000
(224,224,3)
150528 dim.

DNNs	#Neurons	#Layers
LeNet-1	**52**	**7**
LeNet-4	**148**	**8**
LeNet-5	**268**	**9**
VGG-19	**16,168**	**25**
ResNet-50	**95,059**	**176**

38

Large Scale Empirical Study



Coverage Parameters

Coverage Criteria	Parameter setting		
KMNC	k=1,000	k=10,000	N.A
NBC	LB=l	LB=l-0.5*σ	LB=l-σ
	UB=u	UB=u+0.5*σ	UB=u+σ
SNAC	UB=u	UB=u+0.5*σ	UB=u+σ
TKNC	k=1	k=2	k=3
TKNP	k=1	k=2	k=3
NC	th.= 0.0, 0.25, 0.5, 0.75		

5 DNNs Profiling

18 Groups of Large Scale Adv Test Gen

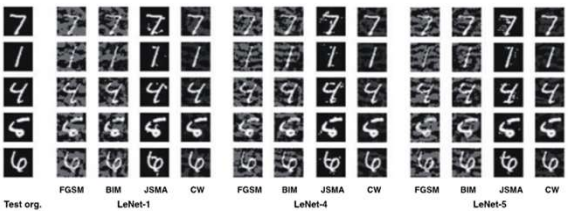
190,000 Test data = 150,000 (MNIST) + 40,000 (ImageNet)

414 Evaluation Configurations= 270 (MNIST) + 144 (ImageNet)

NVIDIA **Tesla M40 GPU 24GB / 18-core 2.3GHz Xeon 64-bit 196 GB**

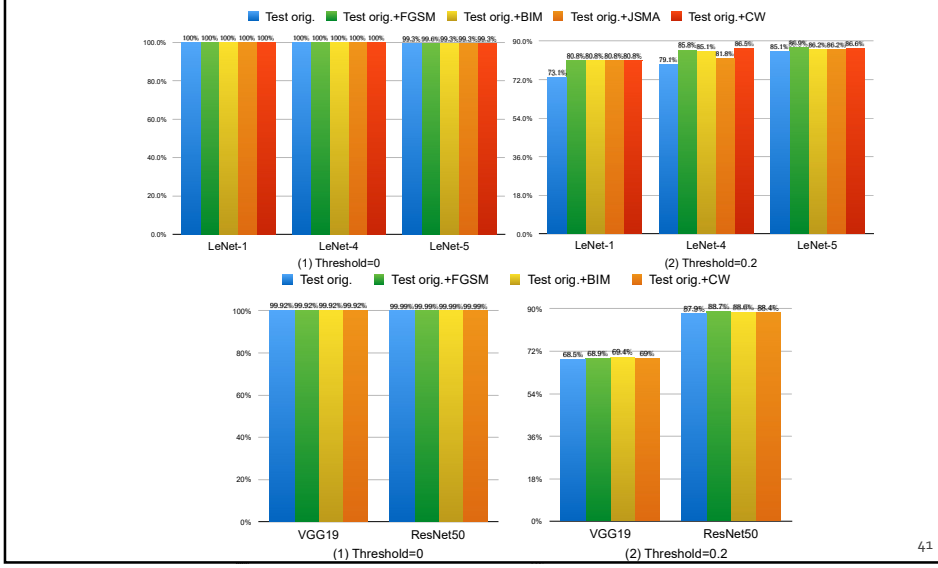
39

Adversarial Examples



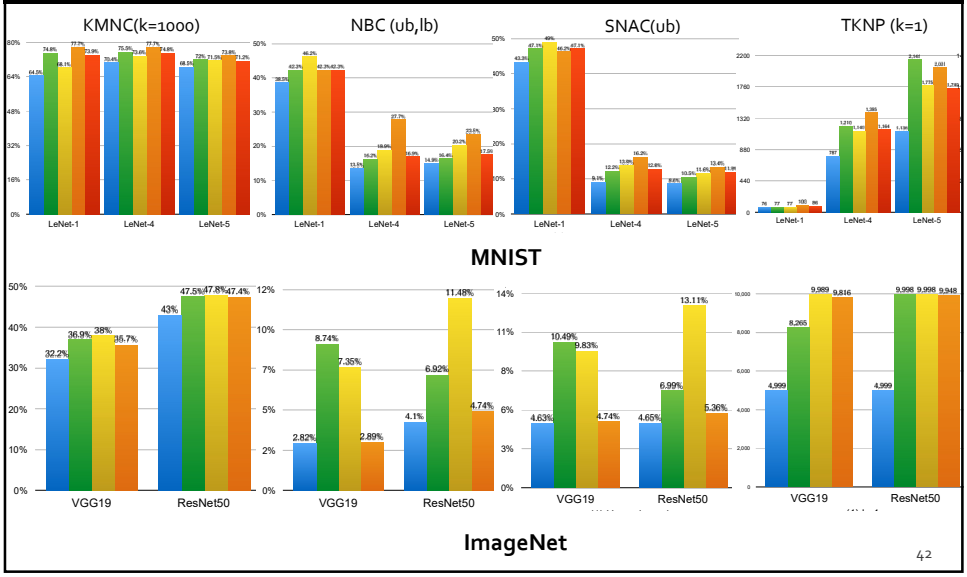
- CW and JSMA make smaller perturbation
- Adv on ImageNet is often imperceptible
- JSMA and CW is computation intensive
- Even more obvious on ImageNet subjects

DeepXplore's Neuron Coverage



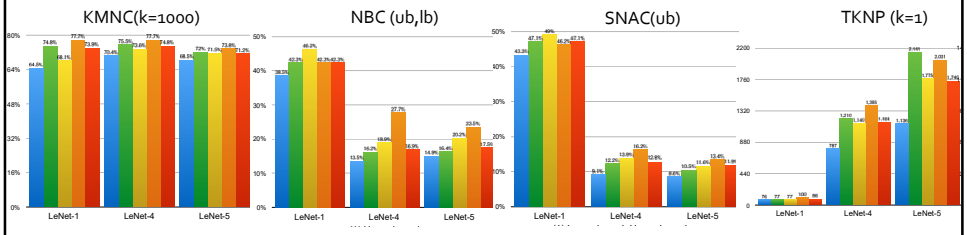
41

DeepGauge Results Glimpse



42

DeepGauge Results Summary (1)

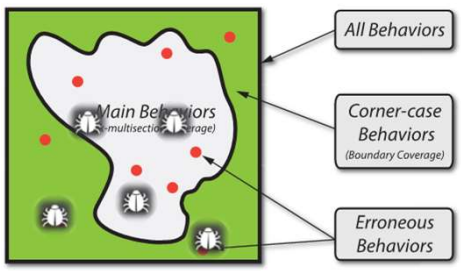


- **Our criteria could distinguish benign and defects with little perturbations**
- **Increase the coverage could potentially increase the chance of defect detection**
- **TestGen guided by DeepGauge generates thousands of unique error trigger tests**

43

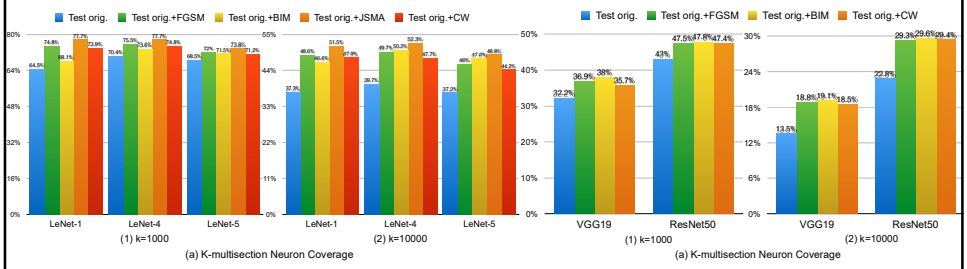
DeepGauge Results Summary (2)

- **Defects could occur in both regions, which need to be tested**
- **KMNC larger than NBC and SNAC, corner cases are difficult to cover**



44

DeepGauge Results Summary



- Criteria show different precision with different parameters
- The accuracy could be adapted on specific DNN models
- Precision could be further enhanced efficiently with bucket

45

Towards Quality Assurance for DL Systems

DeepGauge

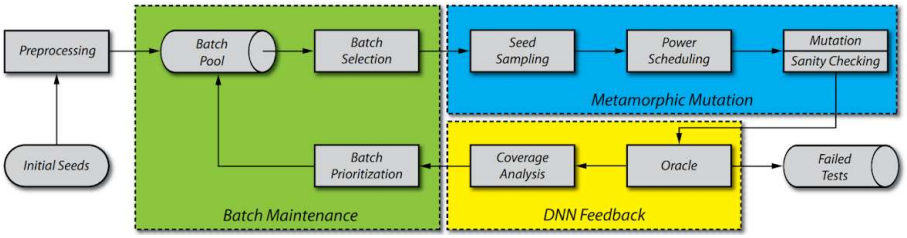
- Multi-granularity testing criteria for DL systems (ASE 2018)
  - ACM SIGSOFT Distinguished Paper Award

DeepHunter

- **Coverage-guided fuzzing testing framework**

46

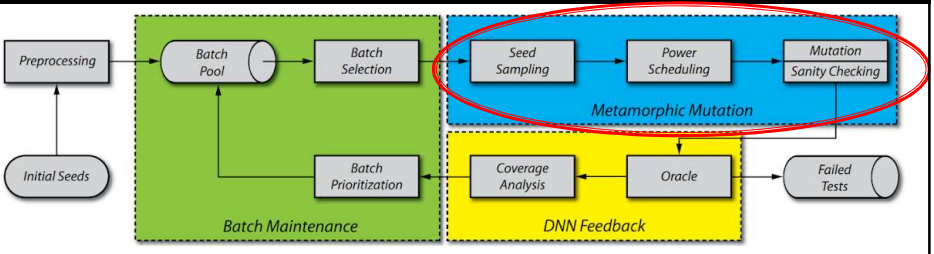
Overview of DeepHunter



6 coverage criteria

47

Metamorphic Transformation



- Brightness
- Contrast
- Pixel Noise
- Blurring

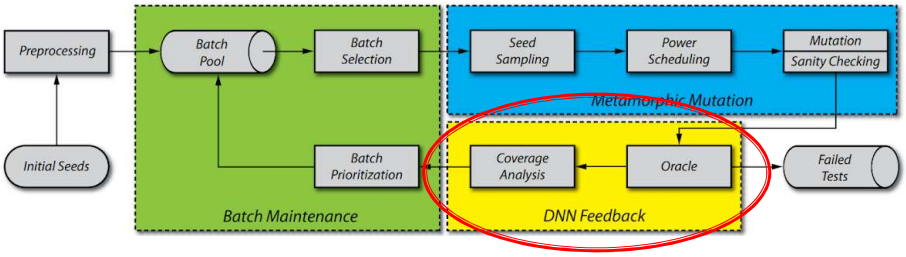
Pixel value transformation

- Translation
- Scaling
- Horizontal Shearing
- Rotation

Affine transformation

48

Coverage Guidance & Oracles



DeepGauge

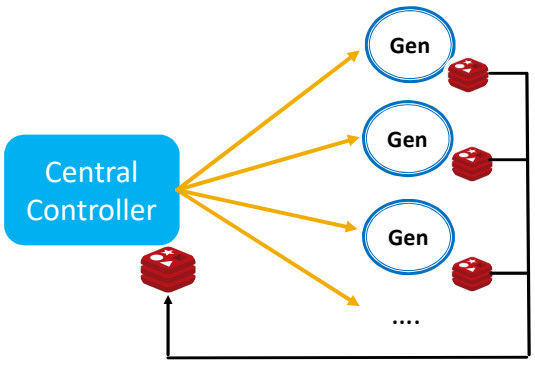
- NC
- KMNC
- NBC
- SNAC
- TKNC

General Oracles

- Robustness
- Fragmentation
- Cross Platform

49

Large Scale Parallelization



iOS BlackBerry TensorFlow Keras Caffe2 Chainer Microsoft CNTK mxnet PYTORCH

50

DeepHunter In Action

- **RQ1:** What coverage can DeepHunter achieve when guided by the six testing criteria?
- **RQ2:** Can DeepHunter enable diverse erroneous behavior detection of DNNs?
- **RQ3:** Can DeepHunter detect potential defects introduced during DNN quantization?

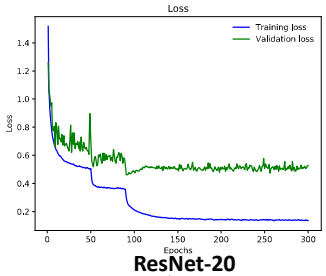
51

Subject Dataset and DNNs

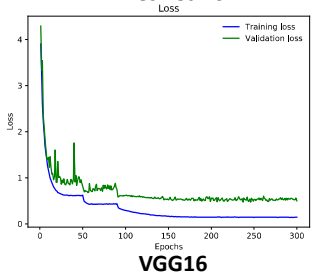
DataSet	Dataset Description	DNN Model	#Neuron	#Layer	Test Acc.
MNIST	Hand written digits recog. from 0 to 9	LeNet-1	52	7	0.976
		LeNet-4	148	8	0.989
		LeNet-5	268	9	0.990
CIFAR-10	General image with 10-class	ResNet-20	2,570	70	0.917
		VGG-16	12,426	17	0.928
ImageNet	1000-class large scale image cla.	MobileNet	38,904	87	0.871*
		ResNet-50	94,059	176	0.929*

* The reported top-5 test accuracy of pretrained DNN model in [45].

DataSet	DNN	Epoch	Syno.	Train Loss	Train Acc.	Test Acc.
MNIST	LeNet-1	10	A	0.131	0.965	0.967
		30	B	0.099	0.975	0.975
		45	C	0.087	0.979	0.976
	LeNet-4	10	A	0.117	0.974	0.978
		25	B	0.077	0.986	0.986
		50	C	0.058	0.990	0.989
	LeNet-5	10	A	0.116	0.977	0.983
		30	B	0.071	0.988	0.989
		45	C	0.056	0.992	0.990
CIFAR-10	ResNet-20	40	A	0.515	0.894	0.859
		55	B	0.385	0.932	0.880
		95	C	0.239	0.977	0.917
	VGG-16	30	A	0.623	0.914	0.850
		55	B	0.443	0.965	0.900
		95	C	0.316	0.995	0.928



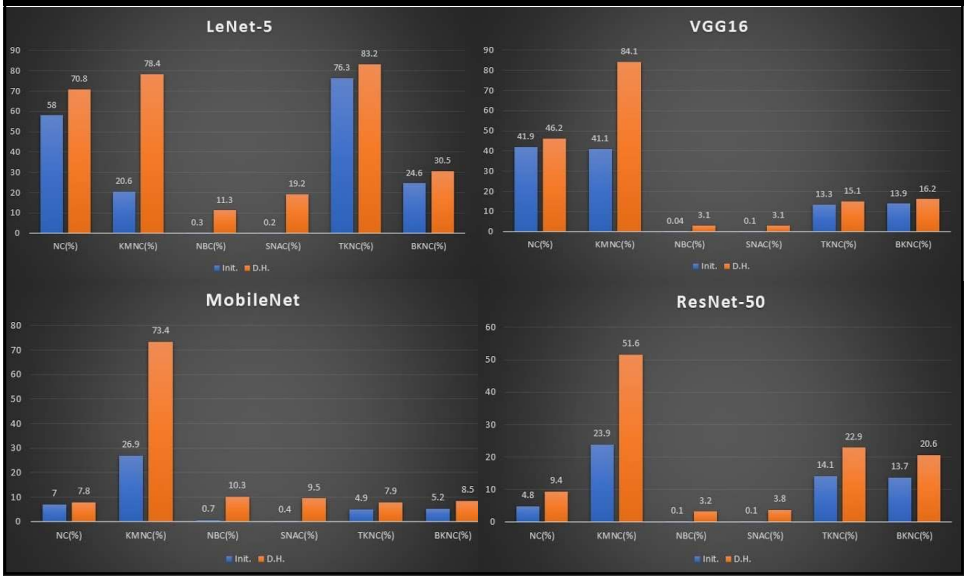
ResNet-20



VGG16

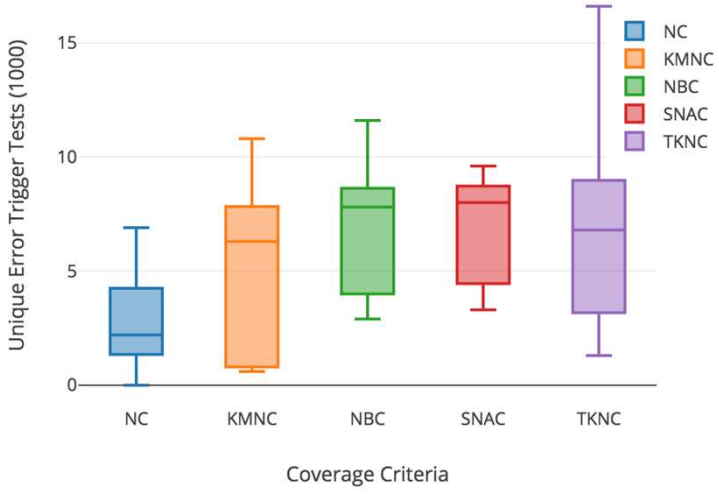
52

Significantly Improve the Coverage



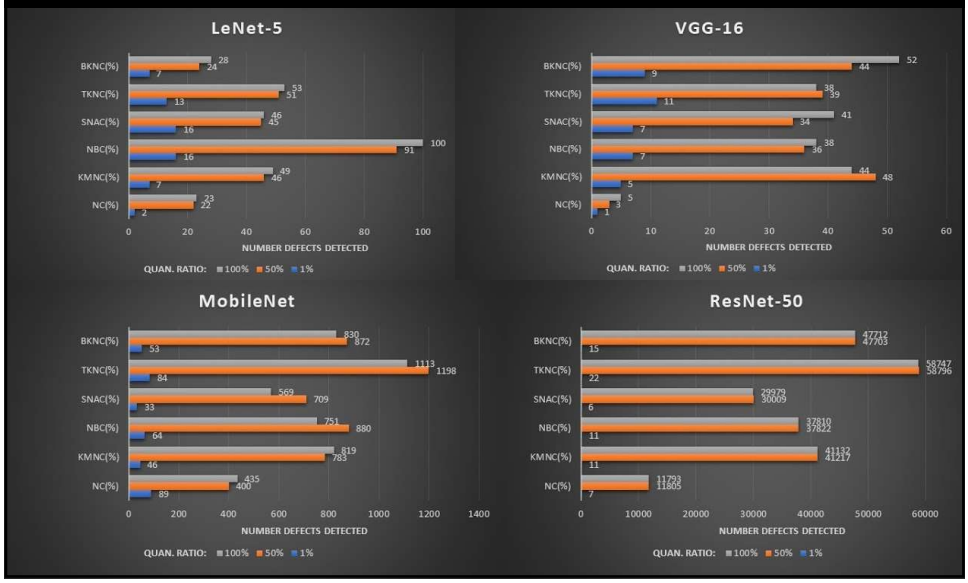
Defect Detection

LeNet-1
LeNet-4
LeNet-5
ResNet-20
VGG-16
MobileNet
ResNet-50



54

DNN Quantization Defects Detection



DeepGuage and DeepHunter

- DeepGuage defines a set of testing criteria to provide a way to gauge testing quality and guide test generation
- DeepHunter leverages coverage feedbacks and performs large scale fuzzing test generation for defect detection of DNN development and deployment

56

DeepMutation: Mutation Testing of DL Systems (ISSRE'18)

DeepMutation: Mutation Testing of Deep Learning Systems

Lei Ma[1,2*], Fuyuan Zhang[2], Jiyuan Sun[2], Minhui Xue[2], Bo Li[4], Felix Juefei-Xu[3], Chao Xie[5], Li Li[2], Yang Liu[2], Jianjun Zhao[1], and Yadong Wang[1]
[1]Harbin Institute of Technology, China [2]Nanyang Technological University, Singapore [3]Kyushu University, Japan
[4]University of Illinois at Urbana–Champaign, USA [5]Carnegie Mellon University, USA [6]Monash University, Australia

Abstract—Deep learning (DL) defines a new data-driven programming paradigm where the internal system logic is largely shaped by the training data. The standard way of evaluating DL models is to examine their performance on a test dataset. The quality of the test dataset is of great importance to gain confidence of the trained models. Using an inadequate test dataset, DL models that have achieved high test accuracy may still lack generality and robustness. In traditional software testing, mutation testing is a well-established technique for quality evaluation of test suites, which analyzes to what extent a test suite detects the injected faults. However, due to the fundamental difference between traditional software and deep learning-based software, traditional mutation testing techniques cannot be directly applied on DL systems. In this paper, we propose a mutation testing framework specialized for DL systems to measure the quality of test data. Specifically, inspired by the role of mutation testing in traditional software, we first define a set of source-level mutation operators to inject faults in the source of DL (i.e., training data and training programs). Then we design a set of model-level mutation operators that directly inject faults into DL models without a training process. Eventually, the quality of test data could be evaluated from the analysis on to what extent the injected faults could be detected. The usefulness of the proposed mutation testing techniques is demonstrated on two public datasets, namely MNIST and CIFAR-10, with three DL models.

***Index Terms*—Deep learning, Software testing, Deep neural networks, Mutation testing**

I. Introduction

Over the past decades, deep learning (DL) has achieved tremendous success in many areas, including safety-critical applications, such as autonomous driving [1], robotics [2], games [3], video surveillance [4]. However, with the existence of recent catastrophic accidents (e.g., Tesla/Uber) relevant to DL, the robustness and safety of DL systems become a big concern. Currently, the performance of DL systems is mainly measured by the accuracy on the prepared test dataset. Without a systematic way to evaluate and understand the quality of the test data, it is difficult to conclude this good performance on the test data indicates the robustness and generality of a DL system. This problem is further exacerbated by many recently proposed adversarial test generation techniques, which performs minor perturbation (e.g., invisible to human eyes [5]) on the input data to trigger the incorrect behaviors of DL systems. Due to the unique characteristics of DL systems, new evaluation criteria on the quality of DL systems are highly desirable, and the quality evaluation of test data is of special importance.

For traditional software, mutation testing (MT) [6] has been established as one of the most important techniques to systematically evaluate the quality and locate the weakness of test data. A key procedure of MT is to design and select mutation operators that introduce artificial faults into the software under test (SUT) to create modified versions (*i.e.*, mutants) of SUT [6], [7]. MT measures the quality of tests by examining to what extent a test set could detect the behavior differences of mutants and the corresponding original SUT.

Unlike traditional software systems, of which the decision logic is often implemented by software developers in the form of code, the behavior of a DL system is mostly determined by the structure of Deep Neural Networks (DNNs) as well as the connection weights in the network. Specifically, the weights are obtained through the execution of training program on training data set, where the DNN structures are often defined by code fragments of training program in high-level languages (e.g., Python [8], [9] and Java [10]).[1] Therefore, the training data set and the training program are *two major sources* of defects of DL systems. For mutation testing of DL systems, a reasonable approach is to design mutation operators to inject potential faults into the training data or the DNN training program. After the faults are injected, the training process is re-executed, using the mutated training data or training program, to generate the corresponding mutated DL models. In this way, a number of mutated DL models $\{M'_1, M'_2, \ldots, M'_n\}$ are generated through injecting various faults. Then, each of the mutant models M'_i is executed and analyzed against the test set T, in correspondence to original DL model M. Given a test input $t \in T$, t detects the behavior difference of M and M'_i if the outputs of M and M' are inconsistent on t. Similar to mutation testing for traditional software [6], the more behavior differences of the original DL model M and the mutant models $\{M'_1, M'_2, \ldots, M'_n\}$ could be detected by T, the higher quality of T is indicated.

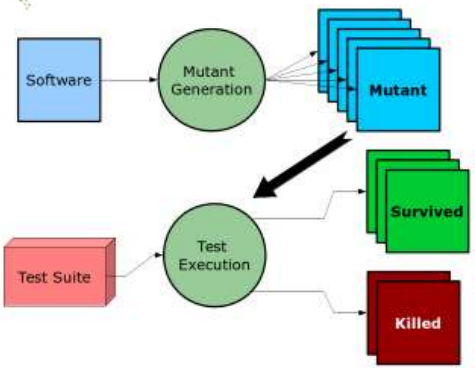
In this paper, we propose a mutation testing framework specialized for DL systems, to enable the test data quality evaluation. We first design eight source-level mutation testing operators that directly manipulate the training data

[1]Although the training program of a DNN is often written in high-level languages, the DNN itself is represented and stored as a hierarchical data structure (e.g., n-d tensor for Keras [8]).

*Lei Ma is the corresponding author. Email: malei@ntu.edu.cn.

Test data Quality Assessment

Mutation Testing Workflow



Combinatorial Testing for DL Systems

Combinatorial Testing for Deep Learning Systems

Lei Ma[1,2], Fuyuan Zhang[2], Minhui Xue[2], Bo Li[4], Yang Liu[2], Jianjun Zhao[1], and Yadong Wang[1]
[1] Harbin Institute of Technology, China
[2] Nanyang Technological University, Singapore
[3] New York University Shanghai, China
[4] University of Illinois at Urbana-Champaign, USA
[5] Kyushu University, Japan
Contact author: malei@ntu.edu.cn

Abstract. Deep learning (DL) has achieved remarkable progress over the past decade and been widely applied to many safety-critical applications. However, the robustness of DL systems recently receives great concerns, such as adversarial examples against computer vision systems, which could potentially result in severe consequences. Adopting testing techniques could help to evaluate the robustness of a DL system and therefore detect vulnerabilities at an early stage. The main challenge of testing such systems is that its runtime state space is too large: if we view each neuron as a runtime state for DL, then a DL system often contains massive states, rendering testing each state almost impossible. For traditional software, combinatorial testing (CT) is an effective testing technique to reduce the testing space while obtaining relatively high defect detection abilities. In this paper, we perform an exploratory study of CT on DL systems. We adapt the concept in CT and propose a set of coverage criteria for DL systems, as well as a CT coverage guided test generation technique. Our evaluation demonstrates that CT provides a promising avenue for testing DL systems. We further pose several open questions and interesting directions for combinatorial testing of DL systems.

Keywords: Combinatorial testing · Deep learning · Adversarial attacks

1 Introduction

Deep learning (DL) systems have been widely applied in various applications due to their high accuracy, such as computer vision [35], natural language processing [39], autonomous driving [5 , and automated medical diagnosis [33]. However, recently DL systems have been shown to be vulnerable against different attacks, such as adversarial examples in computer vision and audio systems. Given that more and more safety-sensitive applications start to adopt DL, deploying DL without thorough testing to safety-critical applications can lead to severe consequences, such as possible accidents in autonomous driving [12]. DL systems are

- Neuron Activation Configuration
- T-way combination sparse coverage
- T-way combination dense coverage
- (p,t)-completeness coverage

Parameters

P1 : A , B , C
P2 : 1 , 2
P3 : X , Y

All Combinations

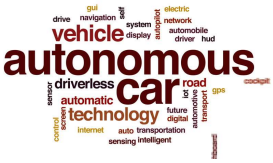
	P1	P2	P3
TC1	A	1	X
TC2	A	1	Y
TC3	A	2	X
TC4	A	2	Y
TC5	B	1	X
TC6	B	1	Y
TC7	B	2	X
TC8	B	2	Y
TC9	C	1	X
TC10	C	1	Y
TC11	C	2	X
TC12	C	2	Y

2-Pair (Pairwise)

	P1	P2	P3
TC1	A	1	X
TC4	A	2	Y
TC6	B	1	Y
TC7	B	2	X
TC9	C	1	X
TC12	C	2	Y

The Fun is Just Starting!

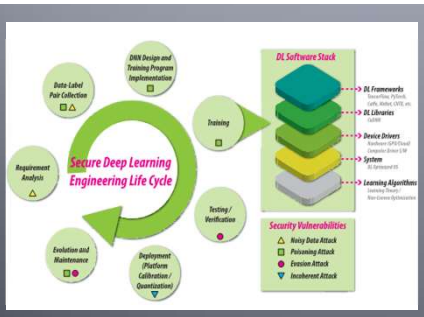
- There are lots of exciting new research problems for DL/ML
  - Build analysis tools for testing and verification of DL/ML systems
  - Build better debugging support for opaque DL/ML systems
  - Make DL/ML systems explainable
  - ….

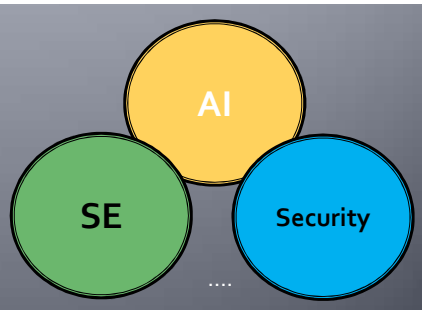


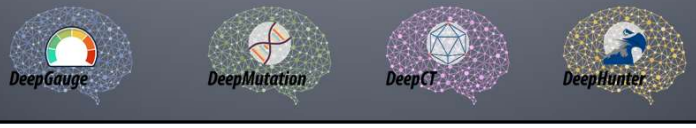
Deep Learning Engineering
Towards the Intelligence of Future

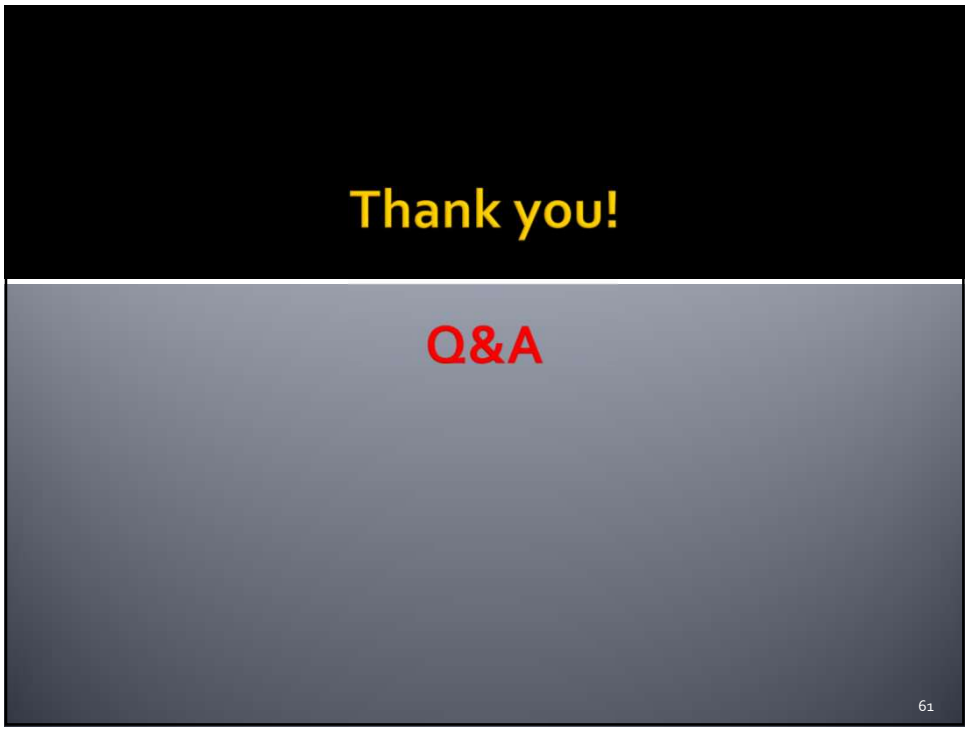
NVIDIA AI Tech Center (NVAITC)

10+ Nvidia Tesla M40, 20+ Tesla V100








Thank you!
Q&A
61