CSE467: Computer Security

25. AI Security

Seongil Wi

Department of Computer Science and Engineering

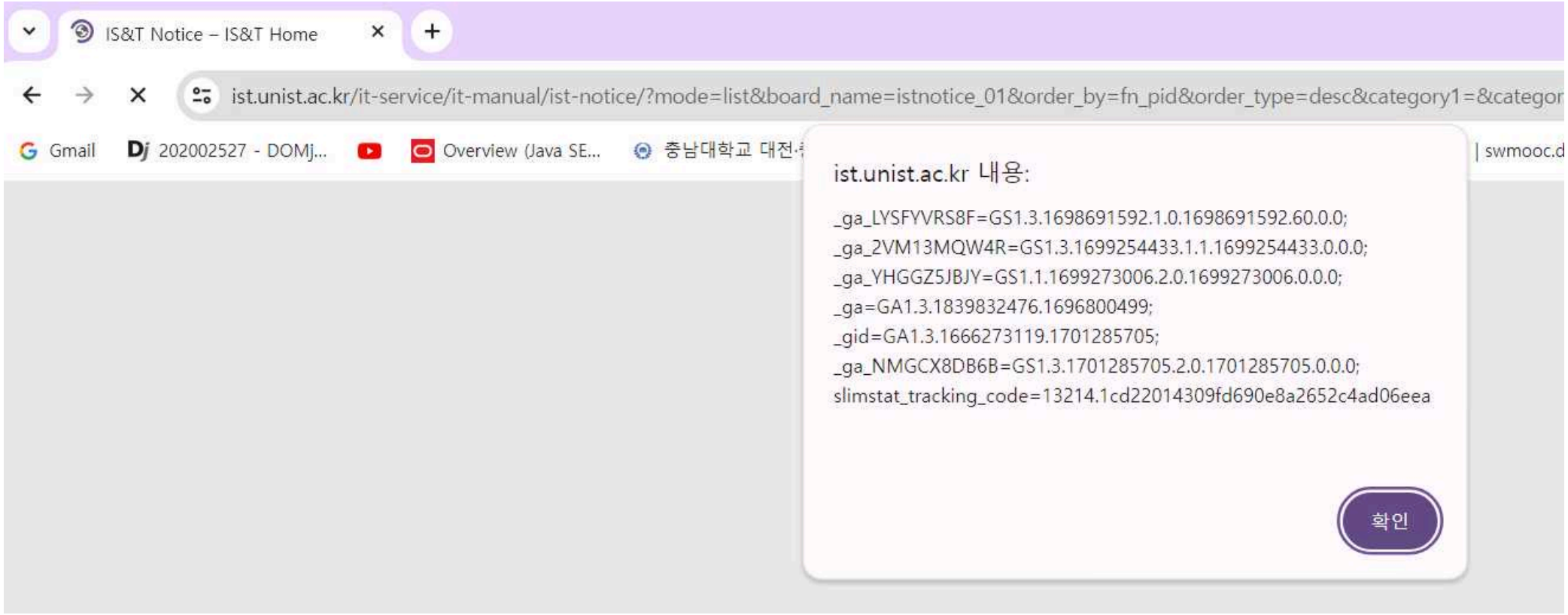
Q&A Session for Your Final Exam

- A Q&A session will be held during the class time on 12/7
  - Thursday!

- No lecture provided
- But, I will be in the classroom during class time (17:30~18:45)
- If you have any questions about what you have learned so far, please come and ask
- You are not required to attend this session!

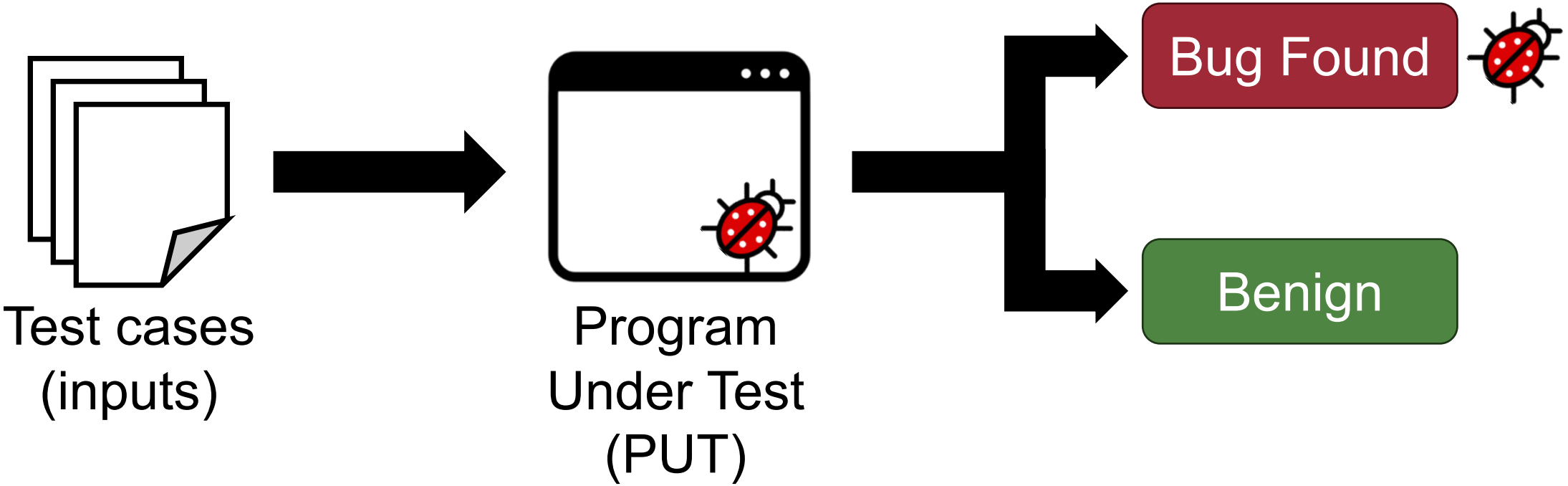
SaveUNIST

- Reflected XSS vulnerability
  - Amazing report from Seungmin Lee ☺



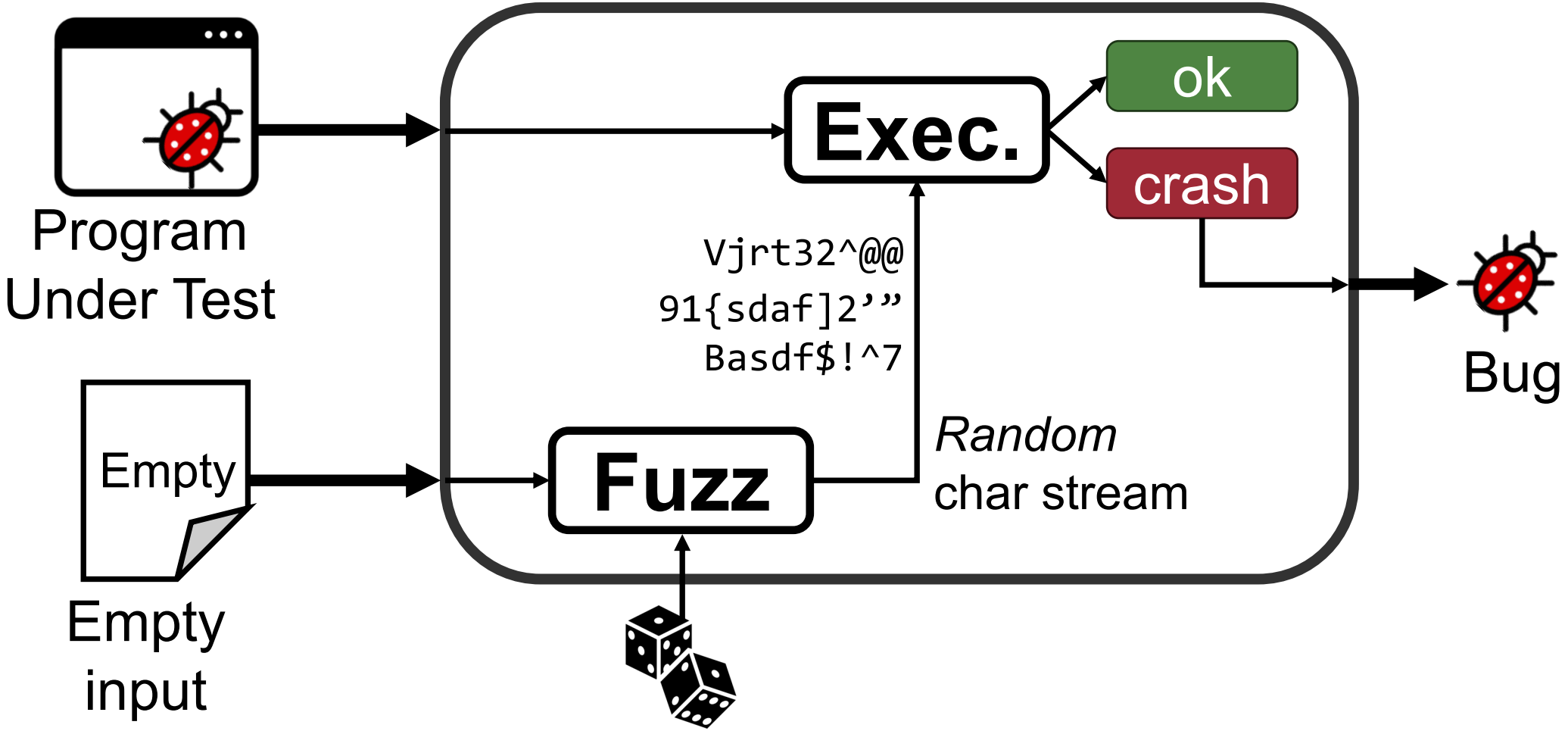
Recap: Dynamic Analysis

- Analyze the program **during an execution** with the concrete input
  - Focuses on a single concrete run
- Keywords: **fuzzing**, penetration testing, scanner, concolic execution, dynamic taint analysis



Recap: Fuzzing in 1990s

- An Empirical Study of the Reliability of UNIX Utilities, ***CACM 1990***



Recap: Definitions

- Based on ***the granularity of what we observe*** in each run:
  - Black-box fuzzing
  - White-box fuzzing
  - Grey-box fuzzing

- Based on ***input production techniques***:
  - Mutation-based fuzzing
  - Grammar-based fuzzing

Recap: Black-box vs. White-box Fuzzing

- **Blackbox**: generate inputs *regardless* of program's logic and structure
  - Pros: easy to implement and low cost
  - Cons: hard to explore deeper parts

```
x = input();
if (x == 482716115)
    bug();
```

- **Whitebox**: generate inputs by *observing* program's logic and structure (a.k.a., dynamic symbolic execution)
  - Pros: can explore deeper parts
  - Cons: Require constraint solving (high overhead)

```
a = input();
b = input();
c = input();
n = input();
if (n > 2)
    if(a^n +b^n == c^n)
        bug();
```

Recap: Grey-box Fuzzing

- White-box fuzzing (strictly speaking)
- Obtain "*some*" partial information about the program execution
  - E.g., code coverage information

- Pros: easy to implement and low cost (easier than white-box fuzzing)
- Pros: can explore deeper parts (deeper than black-box fuzzing)

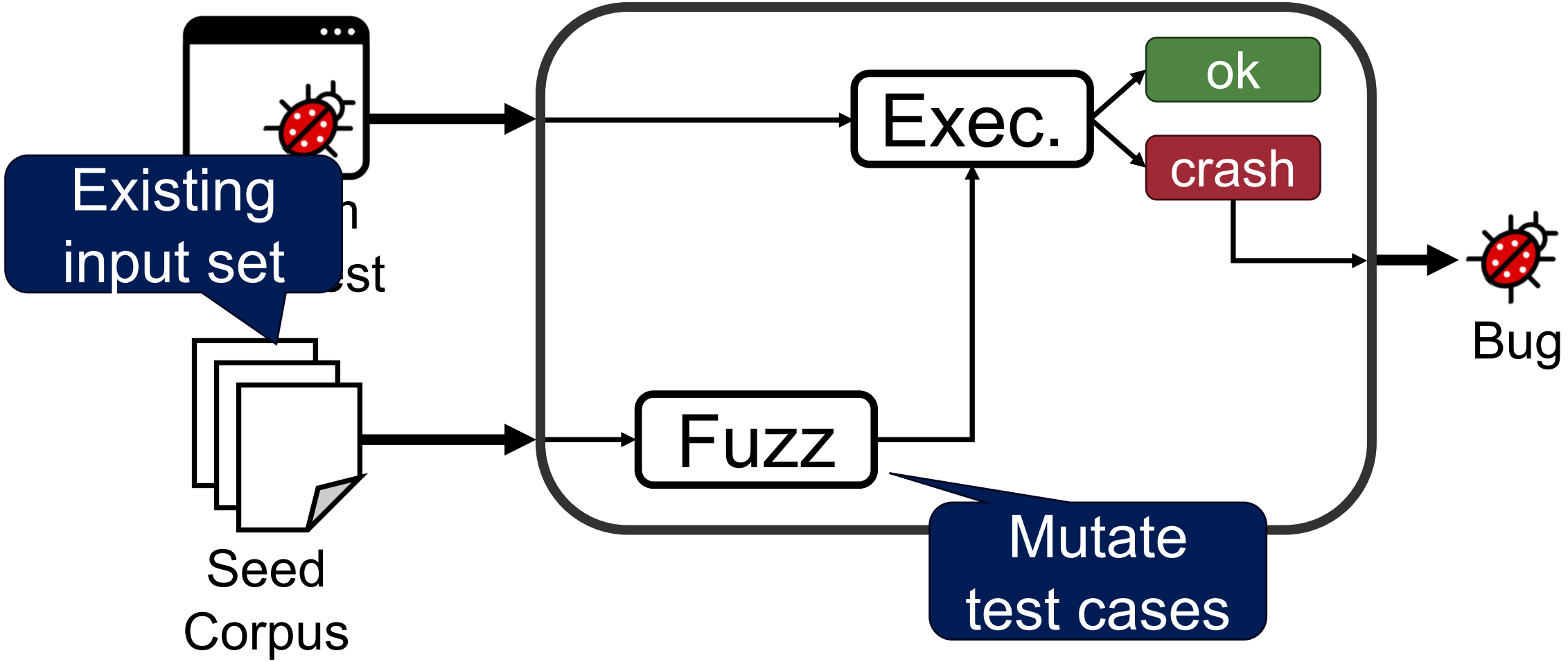
Recap: Mutation- vs. Generation-based Fuzzing

- ***Mutation-based***: mutate a given **seed* to generate test cases
- ***Generation-based***: generate test cases from a model

* ***Seed***: an input to a program

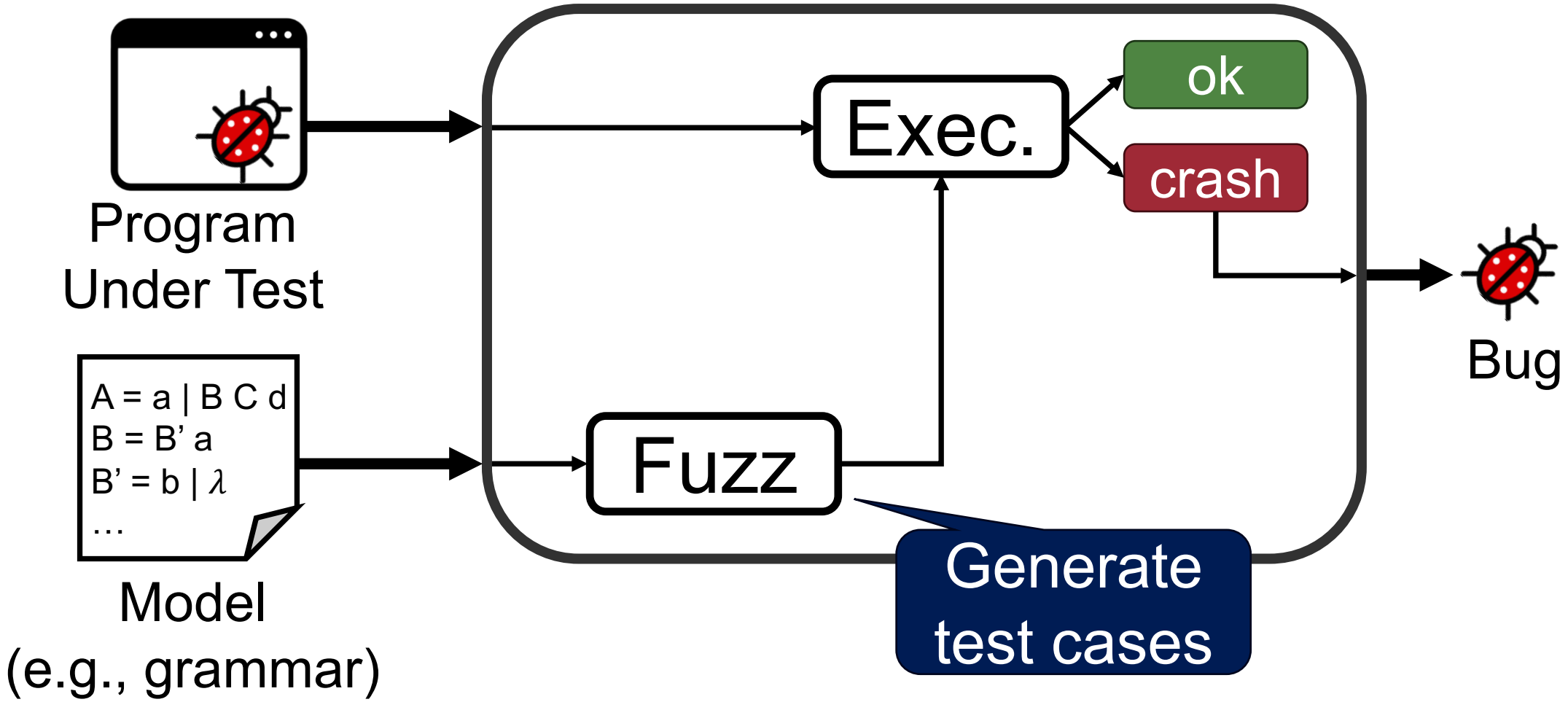
Recap: Mutation-based Fuzzing

- Produce new inputs by mutating existing valid inputs (seeds)

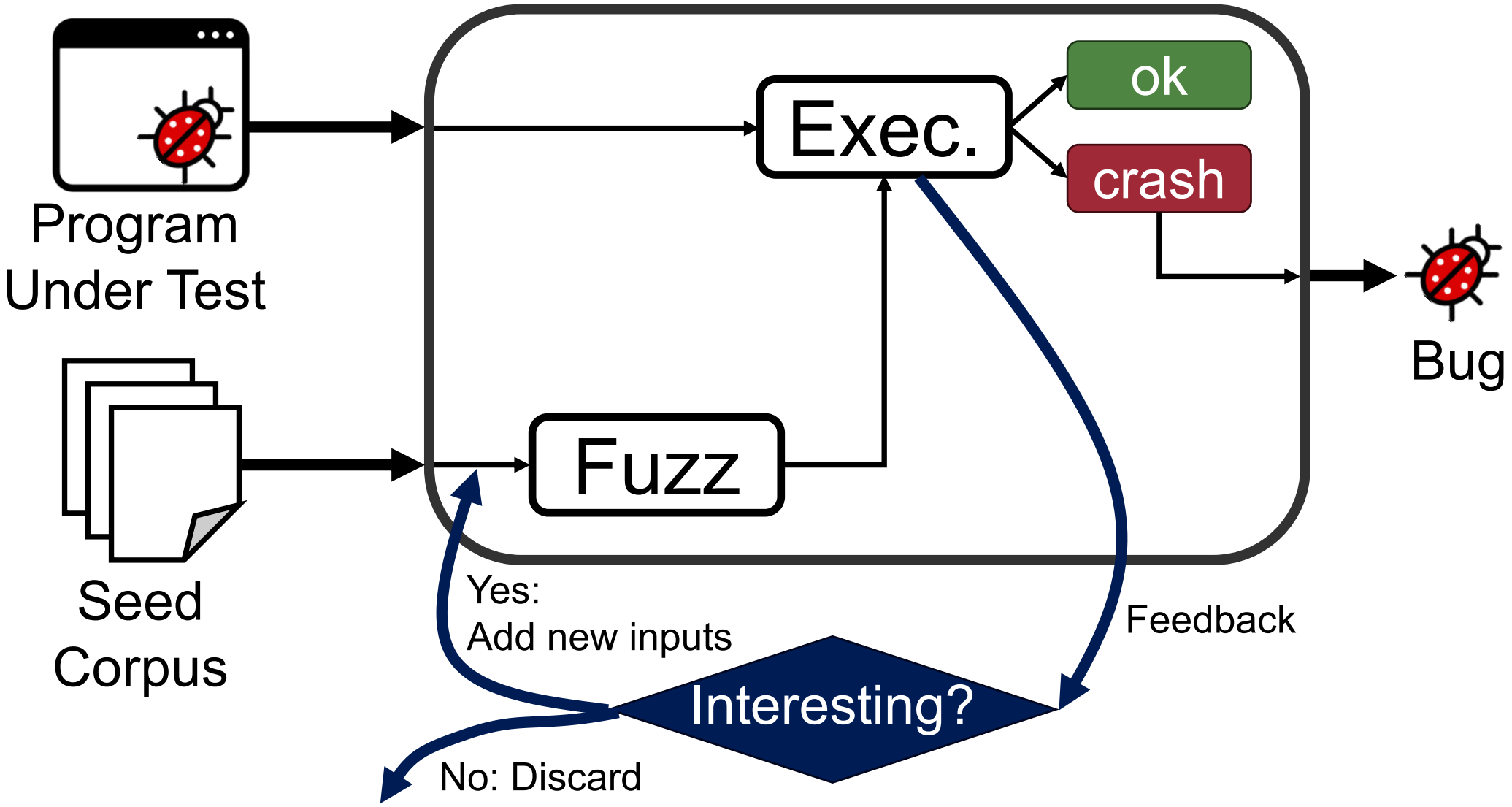


Recap: Generation-based Fuzzing

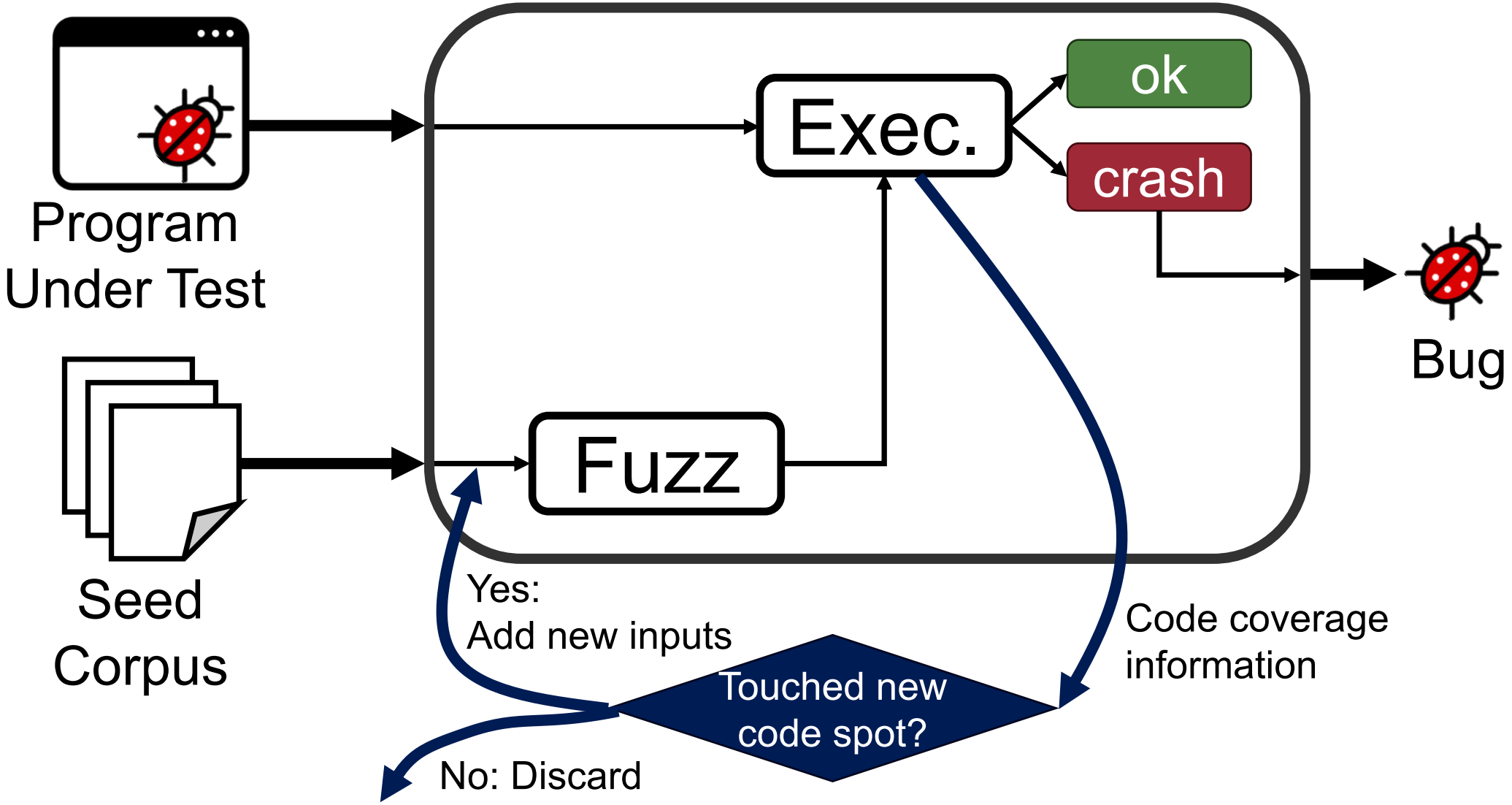
- Generate inputs based on a **given model**
  - Manually defined or automatically inferred



Recap: Feedback-driven Fuzzing



Recap: Coverage-guided fuzzing:
an example of the Feedback-driven Fuzzing



AI Security

Artificial Intelligence (AI)

- The ability of a computer program or a machine to think and learn

- Use cases
  - Image recognition
  - Speech recognition
  - Chatbots
  - Translation
  - Self-driving car
  - …

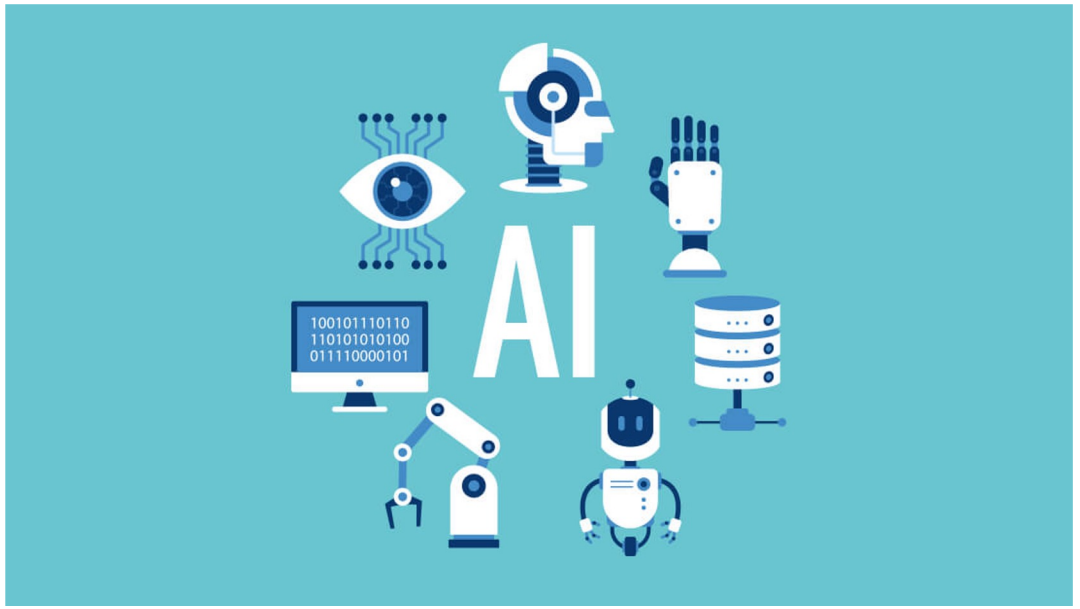


Image from https://tech-blogs.com/

Motivation

- AI (including machine learning or deep learning) has made tremendous progress

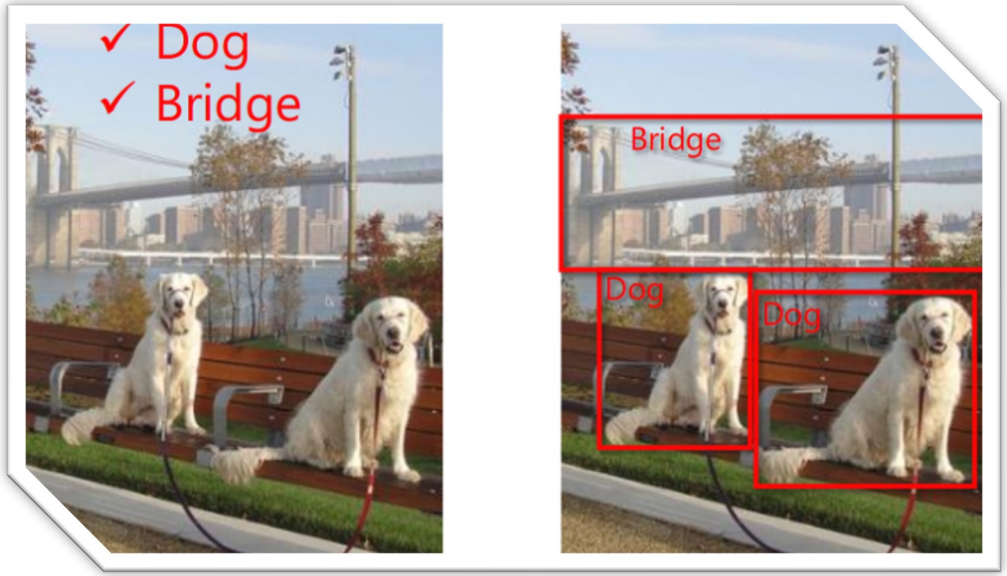


Image from https://www.itnews.com.au/news/driverless-cars-get-victorian-go-ahead-485747
Image from https://devblogs.nvidia.com/malware-detection-neural-networks/

AI Security: Two Aspects

AI for security vs. Security for AI

AI Security: Two Aspects

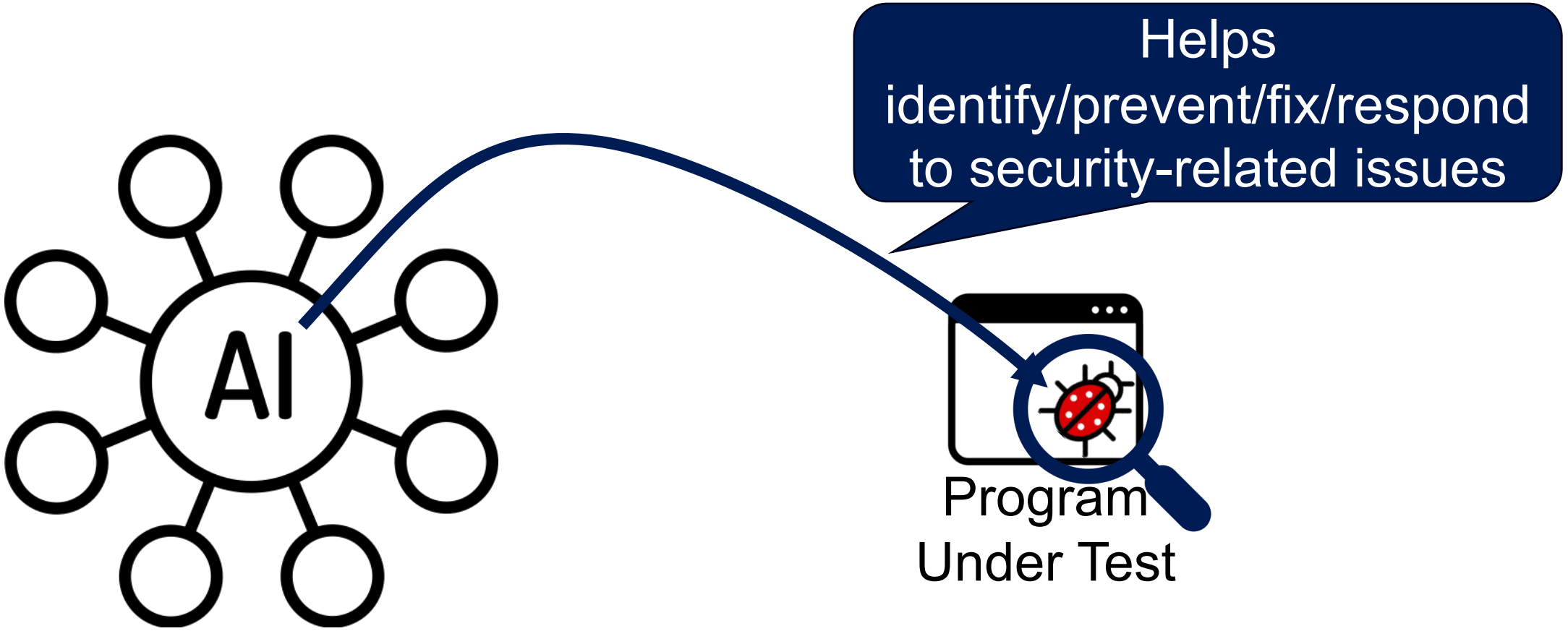
AI for security

vs.

Security for AI

AI for Security

- The tools and strategies that **leverage AI** to identify, prevent, and respond to emerging cyber threats



AI for Security Applications

- Function identification
- Predicting information that has been lost in binaries
- Binary similarity
- Fuzzing
- Symbolic execution
- Password similarity
- Program repair (patching)
- Threat prediction

An Example of AI for Security: RiskTeller

- RiskTeller: Predicting the Risk of Cyber Incidents, ***CCS'17***

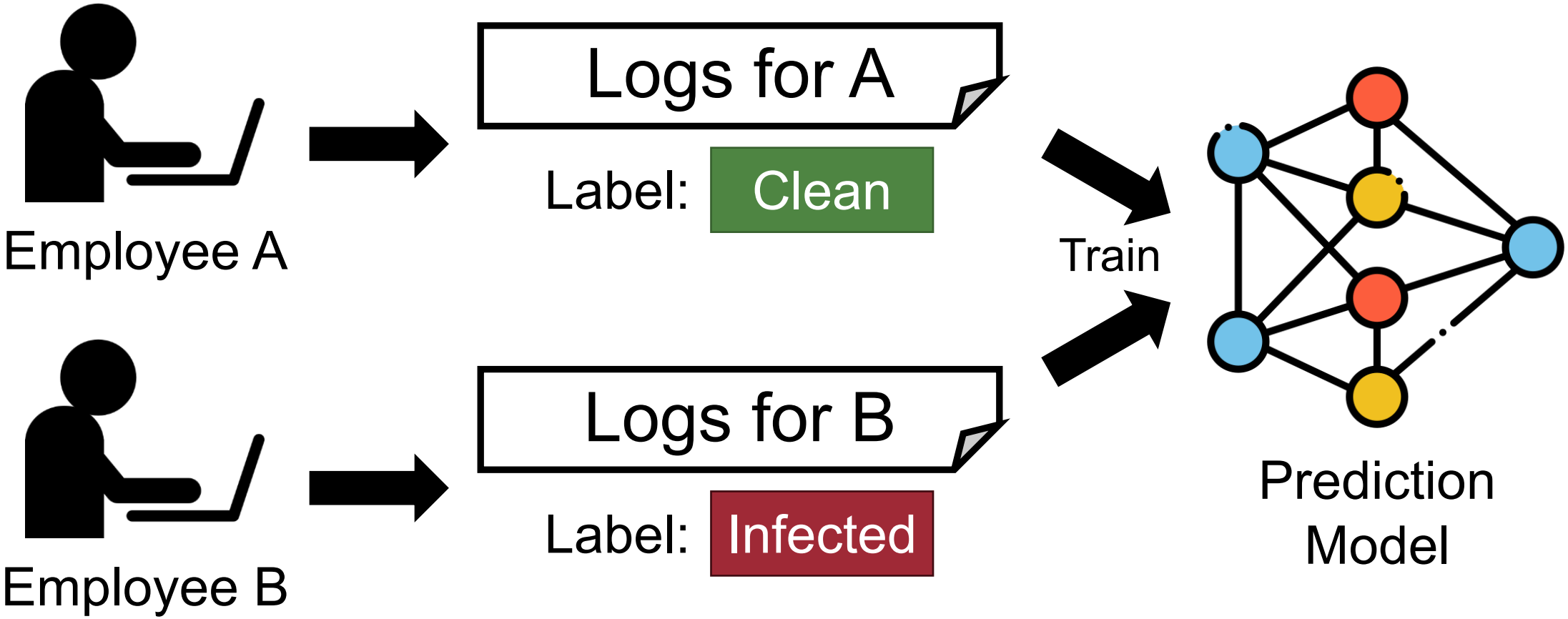
- Problem: no system can be considered invulnerable
- Goal: let's **predict** which machines are at risk of infection!
  - An enterprise would want to know the machines that are more likely to get attacked

Background – Detection vs. Prediction

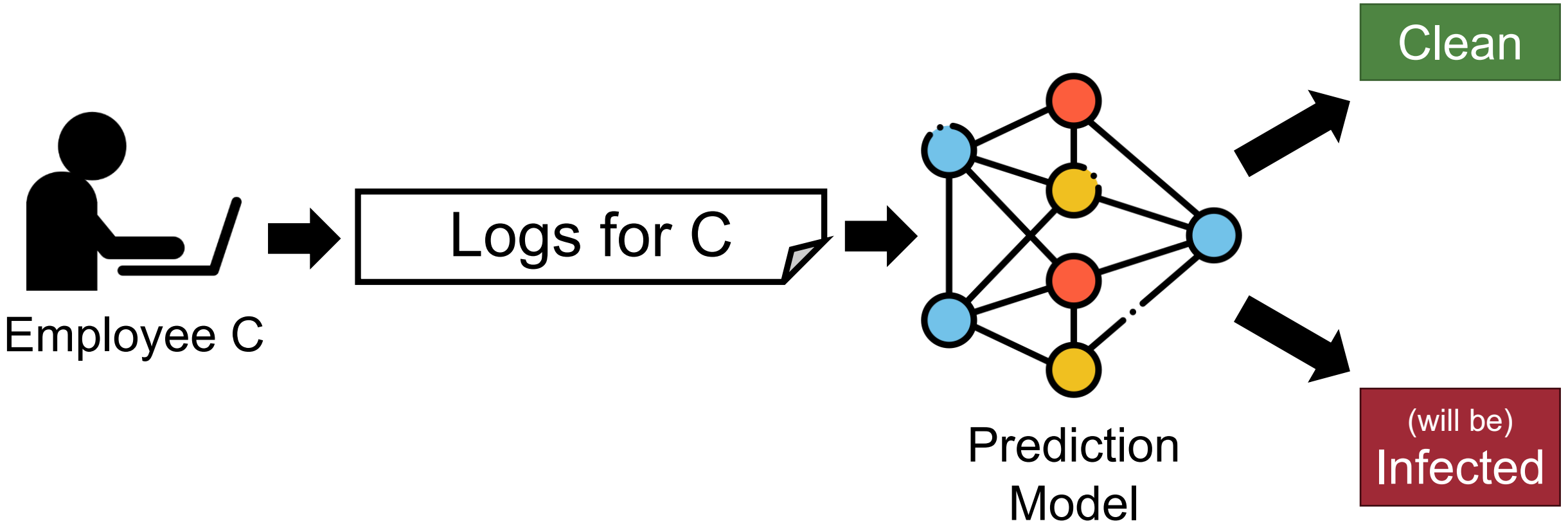
- Predicting future events is a different and more difficult problem than detecting current malicious events!

Overall Approach (Training Phase)

- Key idea: analyze ***per-machine logs*** to discover interesting behavior differences between clean and risky machines



Overall Approach (Testing Phase)



RiskTeller Features for Predictive Modeling

Logs for C

1) ***Volume-based category***: statistics calculated from new binaries appeared during analysis window
   - # of events (appearance of a new binary)
   - Average # of events per active day
   - # of distinct file hashes/filenames
   - # of distinct applications
   - …

RiskTeller Features for Predictive Modeling

Logs for C

2) ***Temporal behavior***: to understand whether longer working hours or working outside the official working hours is correlated with facing higher risk to encounter malware infections
   - Fraction of events during daytime (06:00-18:59)
   - Fraction of events during evening (19:00-00:59)
   - Fraction of events during night (01:00-05:59)
   - Fraction of events during weekdays
   - Fraction of events during weekends
   - …

RiskTeller Features for Predictive Modeling

Logs for C

3) ***Vulnerabilities/patching behaviors***: the patching behavior and the severity of existing vulnerabilities can be highly correlated with the prediction
   - # of patched/unpatched vulnerabilities
   - # of patched/unpatched applications
   - The vulnerability window length for patched applications
   - …

RiskTeller Features for Predictive Modeling

Logs for C

4) ***Application categories***: the used application categories can be correlated with the different risk levels
   - Top-5 application categories with most events
   - Fraction of event per top-5 category
   - …

Table 3: Application categories.

Category	# of Apps	Category	# of Apps
Architecture	59	Government	142
Asset Management	574	Health	1 243
Automobile	172	HR	796
Bank	166	Insurance	246
Business	1 266	IT	353
Chat	87	Legal	547
Chemical	29	Logistics	146
Construction	371	Oil	145
Sales	1 050	Point of Sale	251
Data / DB	254	SDK	490
Education	101	Secretary	100
Engineering	73	Security	294
Finance	1 206	Statistics	71

RiskTeller Features for Predictive Modeling

Logs for C

5) ***Infection history***: it is reasonable to imagine that past infection history is correlated with future events
   - Fraction of events for malicious files
   - Fraction of events for benign files
   - …

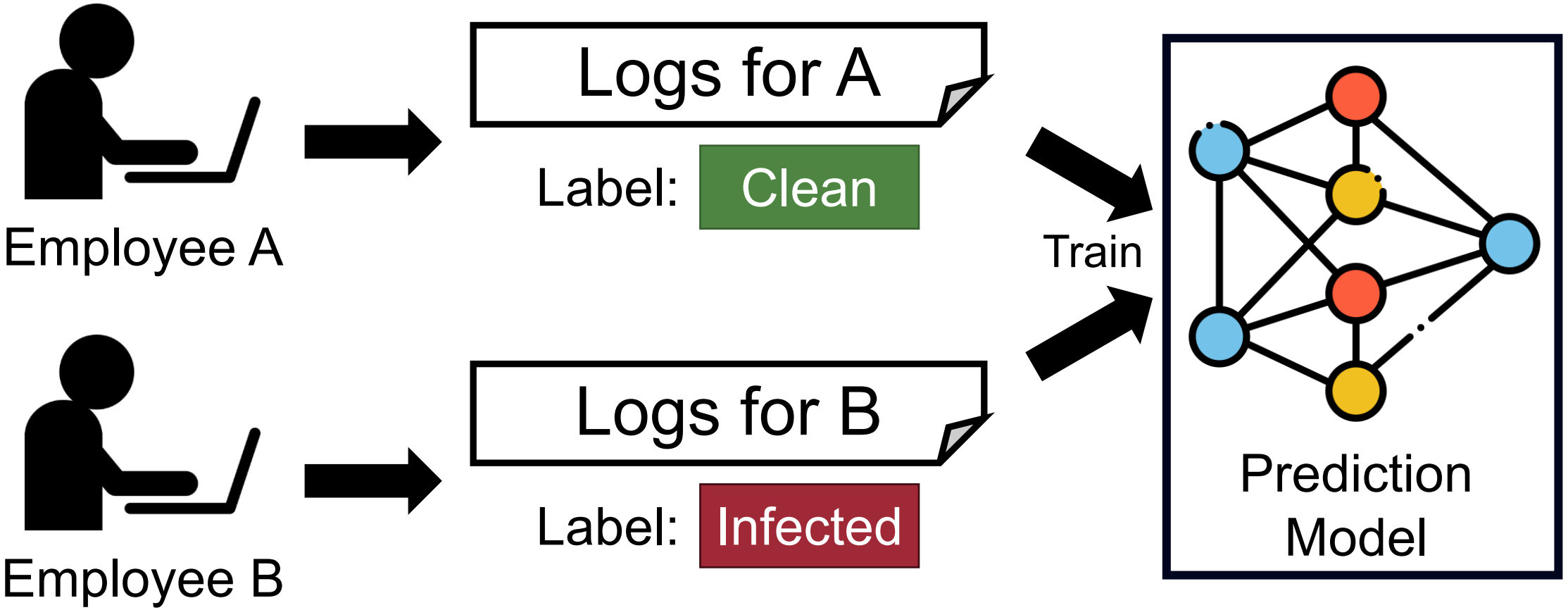
RiskTeller Features for Predictive Modeling

Logs for C

6) ***Prevalence-based features***: malware tends to have lower prevalence than benign software ⇒ the fact that a machine has a large number of low-prevalence files gives us reasons to be suspicious about that machine
   - Fraction of files seen only in one machine
   - Fraction of files seen on [1,10]/[11,100]/[101,1000]/[1001, ∞) machines
   - Fraction of files seen only in one enterprise
   - Fraction of files seen on [1,10]/[11,100]/[101,1000]/[1001, ∞) enterprises

Overall Approach (Training Phase)

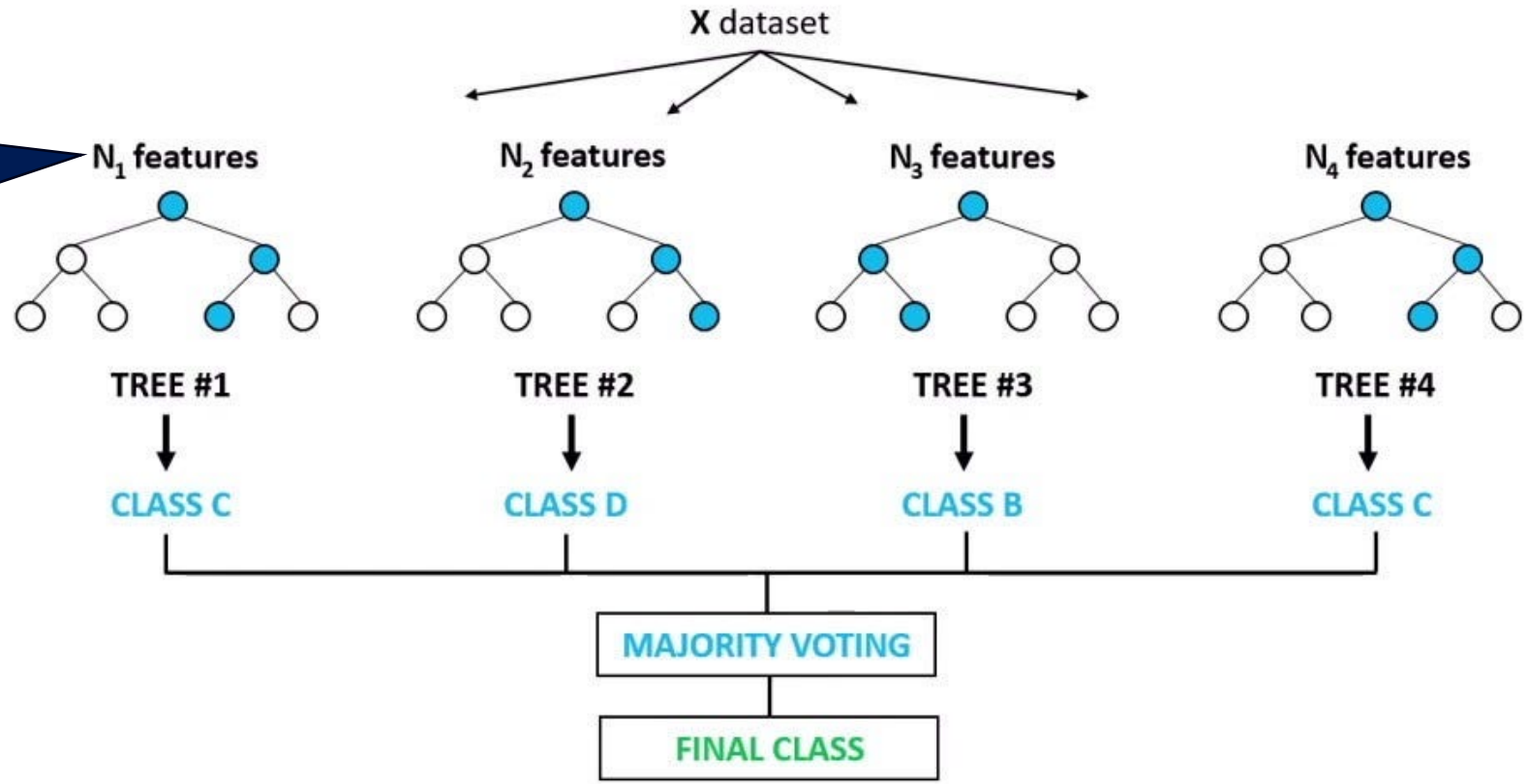
- Key idea: analyze ***per-machine logs*** to discover interesting behavior differences between clean and risky machines



Training Algorithm

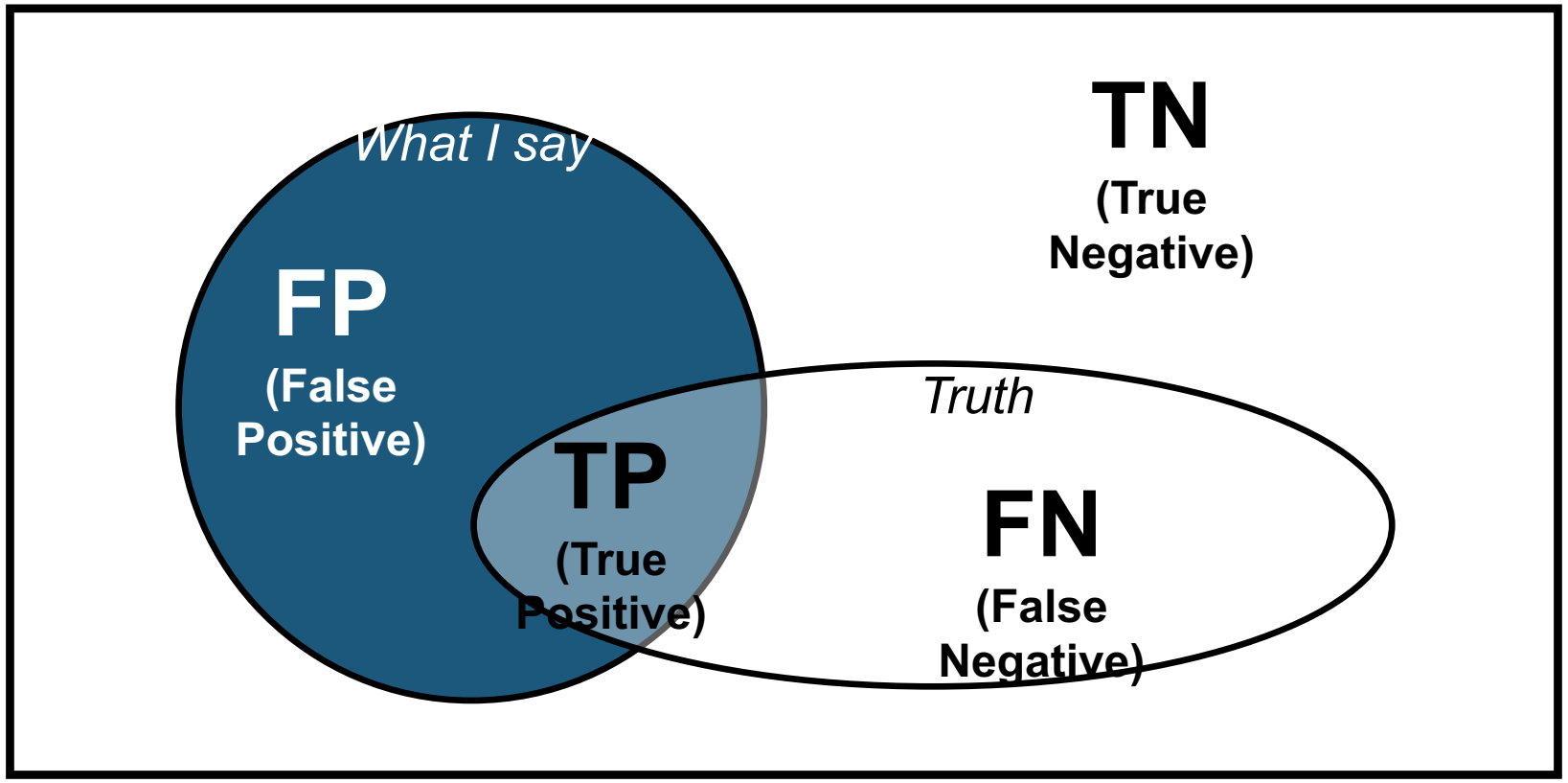
- Used model: random forest classifier
  - Aims at reducing the variance of the learning model through bias-variance trade-off
- Run the random forest classifier with 800 trees as the threshold

Randomly selected subset of features



Evaluation

- Dataset: 16M benign, 214M infected (or unknow)
- Achieve a 96% ***true positive rates*** with only 5% ***false positive rates*** (After a 10-fold cross validation)



- ***true positive rates*** = TP / (TP + FN)
- ***false positive rates*** = FP / (TP + FP)

AI Security: Two Aspects

AI for security

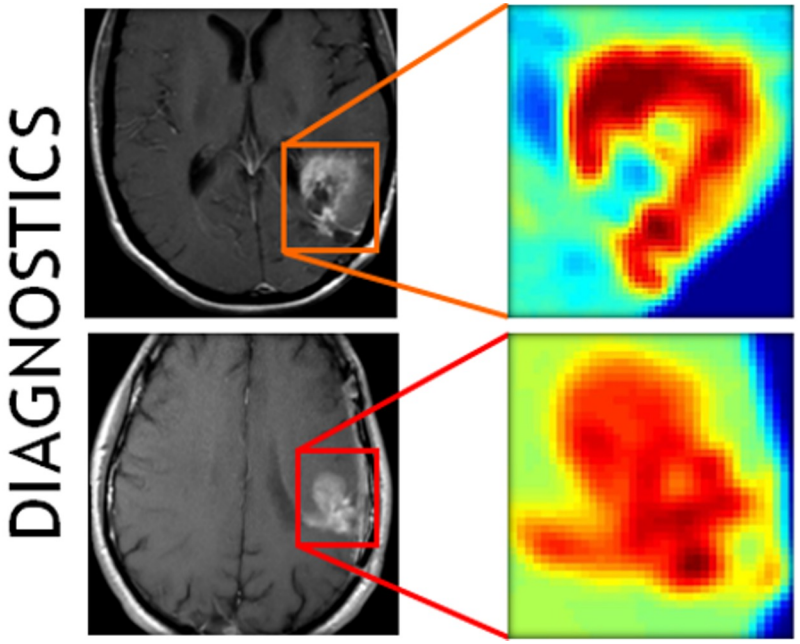
vs.

Security for AI

Motivation

- Deep learning is increasingly used in safety-critical systems

Self-driving car



Medical diagnosis

Malware detection

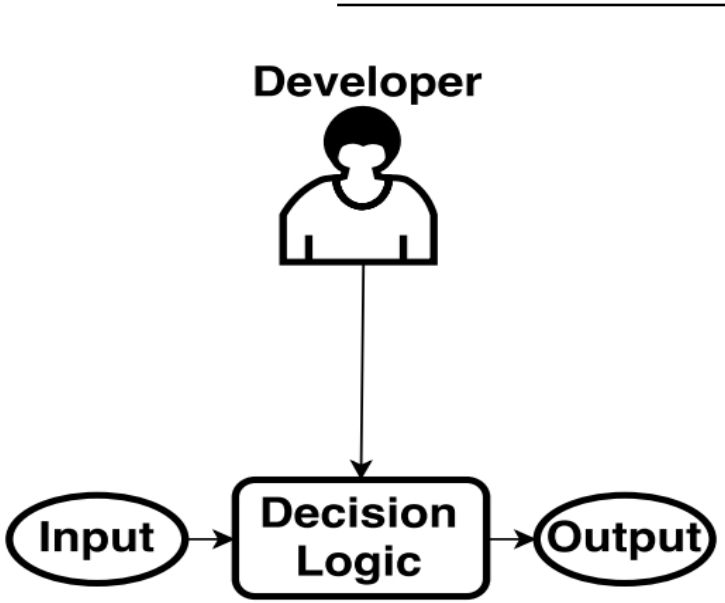
Can We Believe the AI Systems?

Security for AI

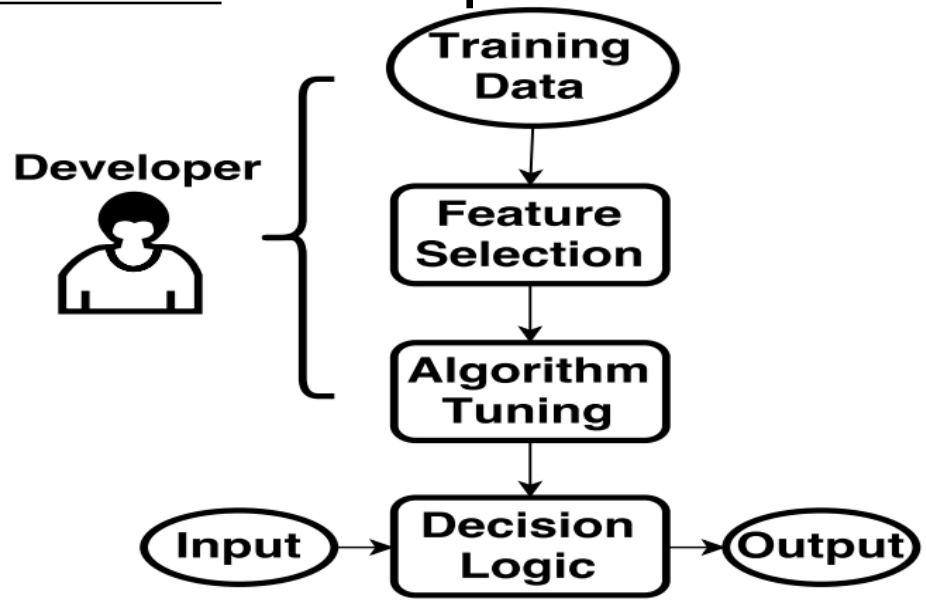
- The tools and strategies to identify, prevent, and respond to new cyber threats due to the emergence of AI

Machine Learning (ML) System Development

- The development process of the machine learning component is *different* from traditional software development



Traditional software development



ML system Development

Finding and fixing of erroneous behaviors of machine learning systems are crucial in safety-critical settings!

Example: HackGPT

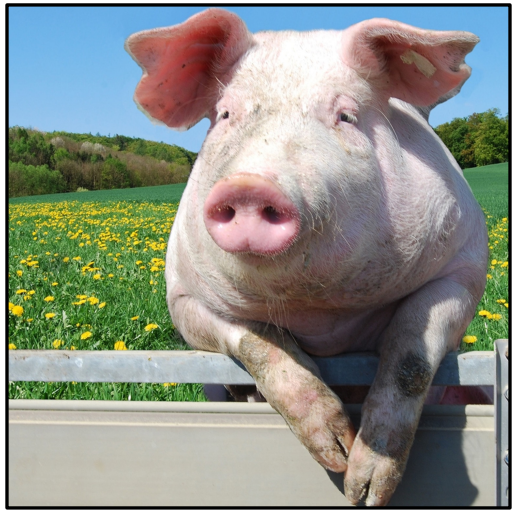
- Does ChatGPT become a friend or villain?
- **Report NEW security threats that can be caused by ChatGPT!**
  - Read “Large Language Models like ChatGPT say The Darnedest Things”, ***CACM’23***
  - DO NOT report known issues (e.g., generating buggy code): a lot of studies already done
  - Describe a concrete and detailed scenario

Image from https://www.the420.in/hackers-abusing-openais-chatgpt-to-build-nefarious-online-tools-israeli-report-claims-amid-microsofts-10-billion-investment-talks/

Adversarial Attacks

- A cyber-attack that aims to misguide a model with malicious input

Normal Cases in AI



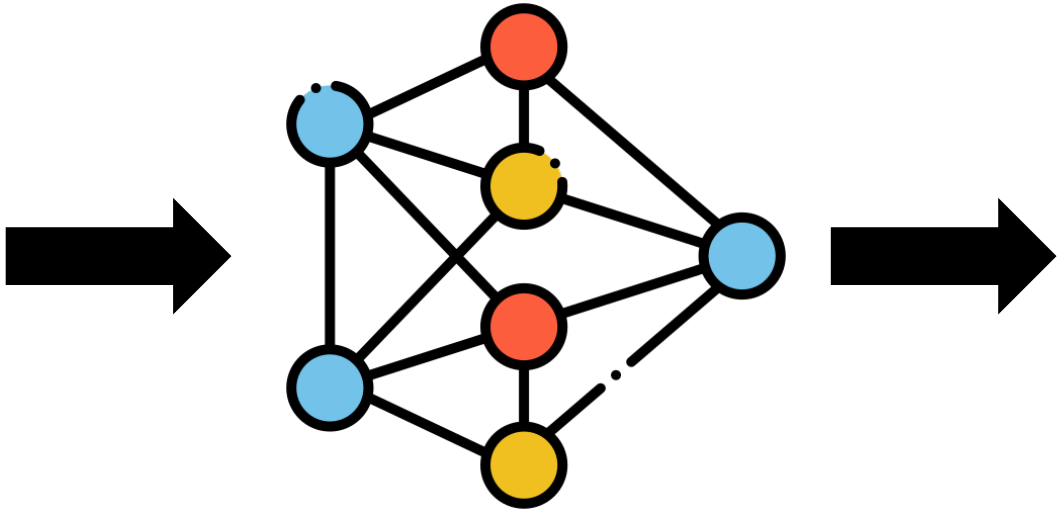


Image
classification

“pig”
58% confidence

Normal Cases in AI

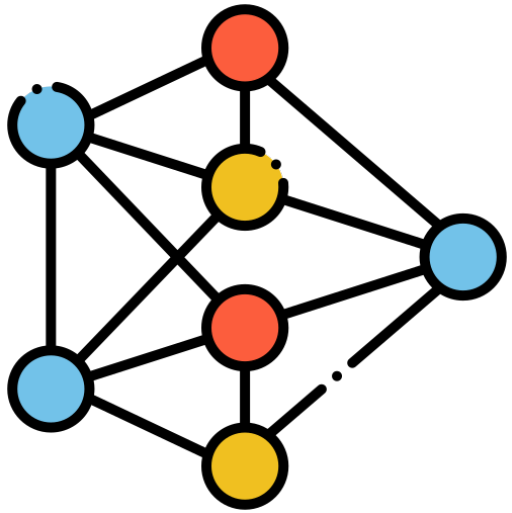
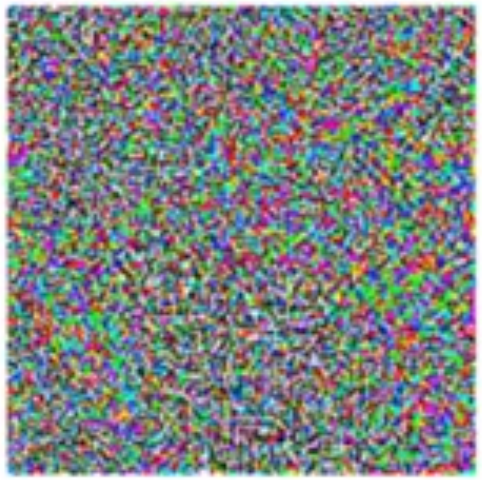


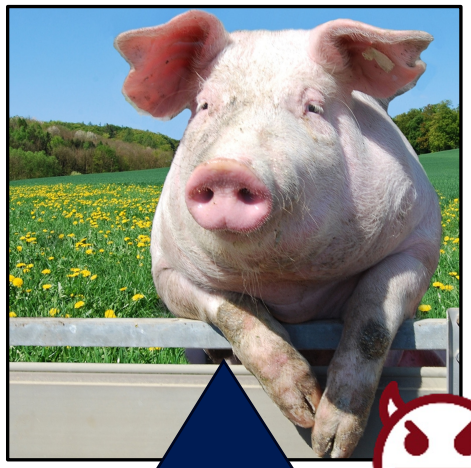
Image
classification

“airplane”
97% confidence

Adversarial Examples in AI



Noise



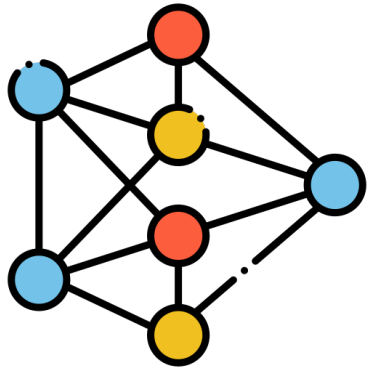


Image classification

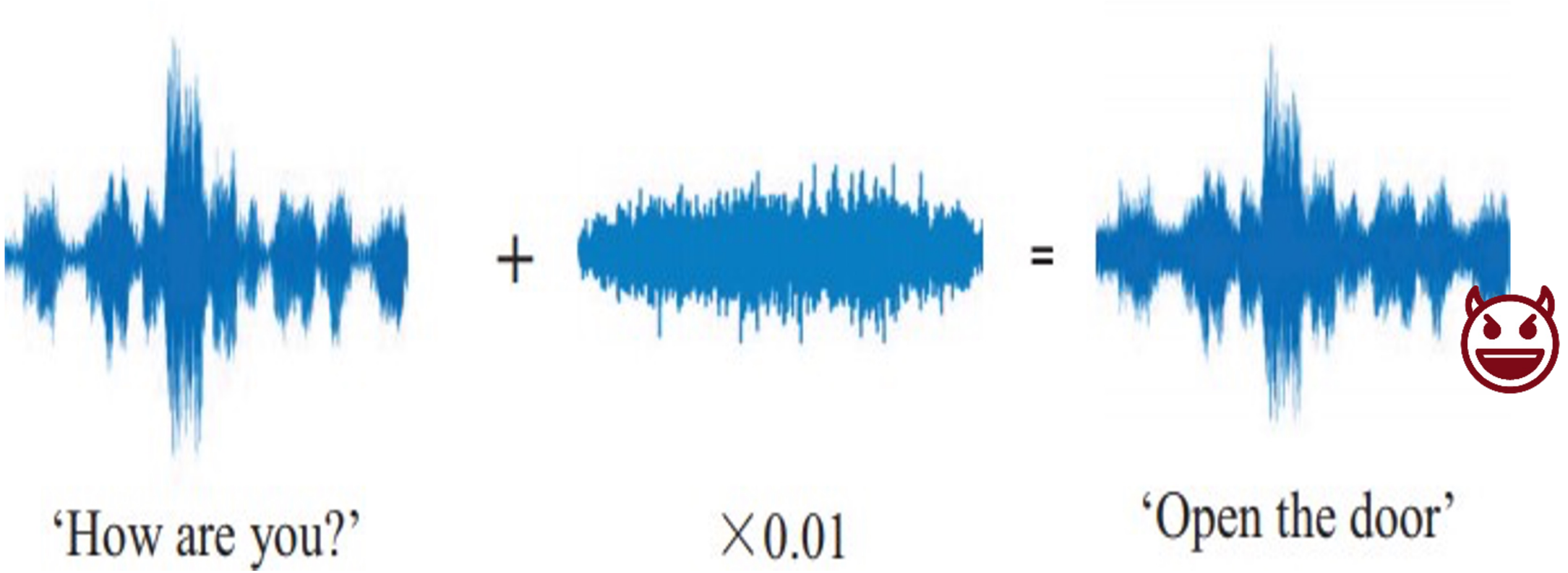
Lead to incorrect classification

"airplane"
99.3% confidence

Introduce a small perturbation (indistinguishable to the human's eye)

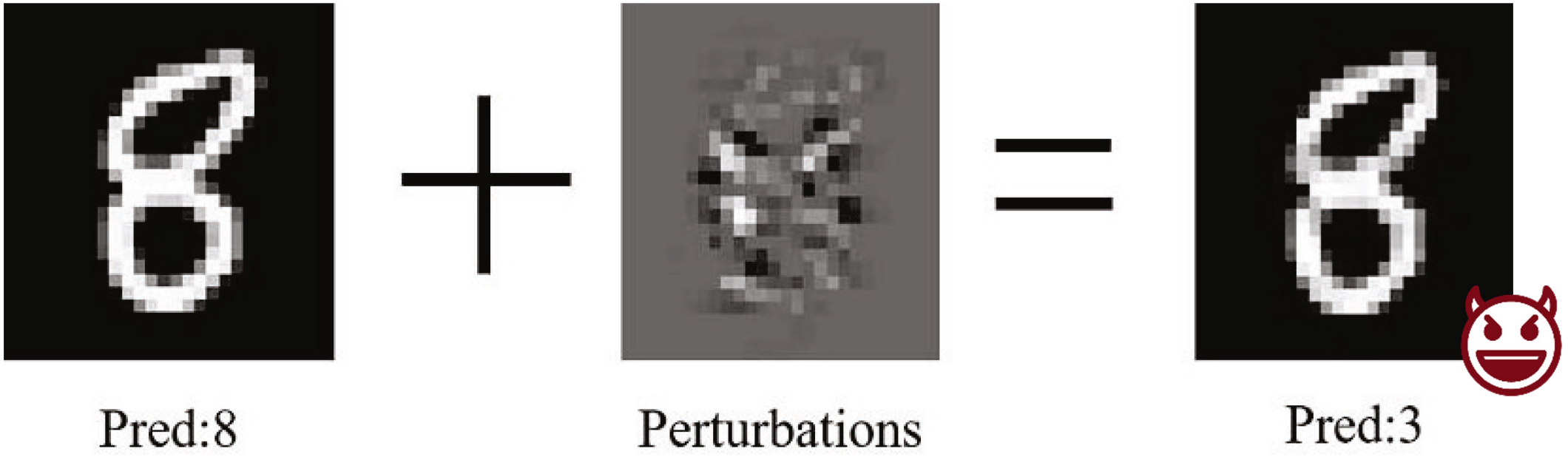
Adversarial Examples in AI

- Voice recognition



Adversarial Examples in AI

- Number recognition



Adversarial Examples in AI

- Self-driving car

(a) Input 1

(b) Input 2 (darker version of 1)

Real-world Examples

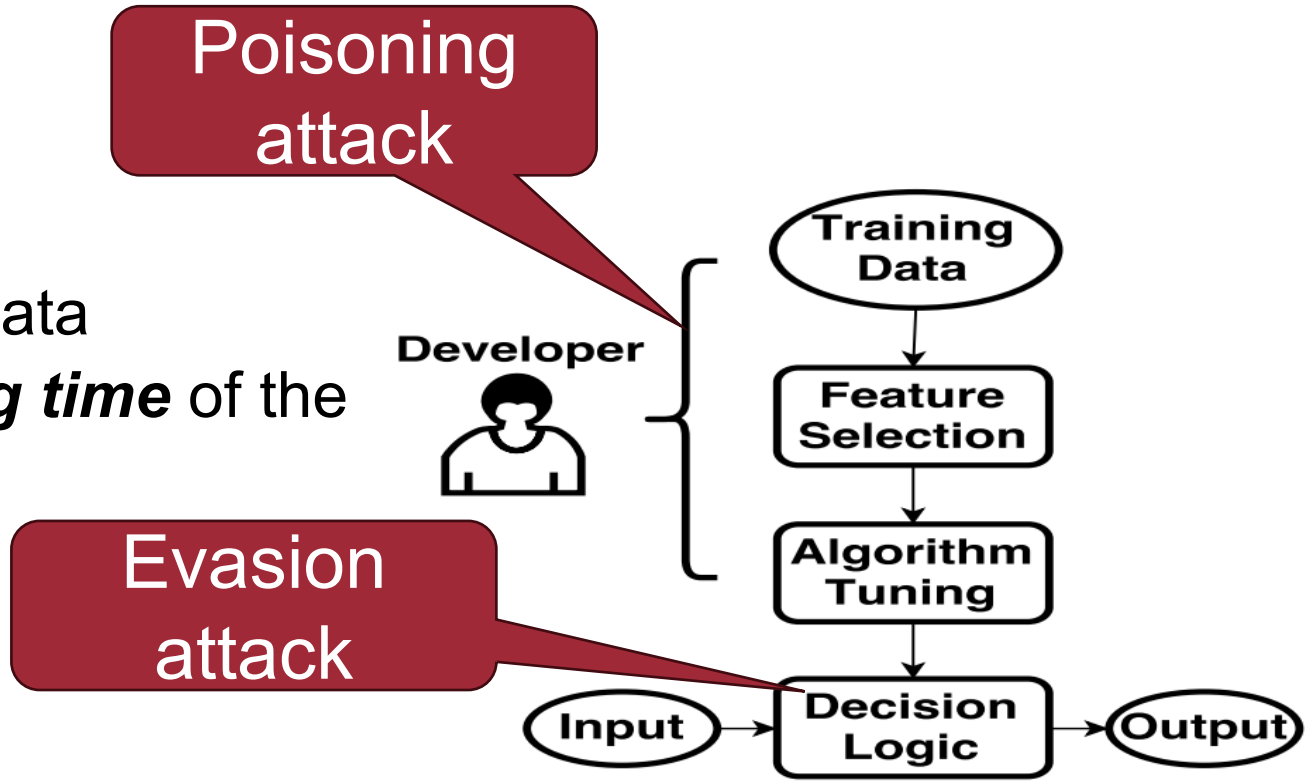
- A Tesla car crashed into a trailer (2015)
  - Autopilot system failed to recognize the trailer as an obstacle

Due to its “white color against sky” and the “high ride height”

Adversarial Attacks: Main Attack Scenarios

- Evasion attack
  - The adversary tries to evade the system by adjusting malicious samples
  - Takes place during the ***testing time*** of the machine learning model

- Poisoning attack
  - Contamination of the training data
  - Takes place during the ***training time*** of the machine learning model



Evasion Attacks

- The most common type of attack in the adversarial setting
- The adversary tries to evade the system by adjusting malicious sample during ***testing phase***

- How to make adversarial examples?:
  - Fast Gradient Sign Method (FGSM)
  - Projected Gradient Descent (PGD)
  - Carlini&Wagner's attack (CW)
  - Etc.

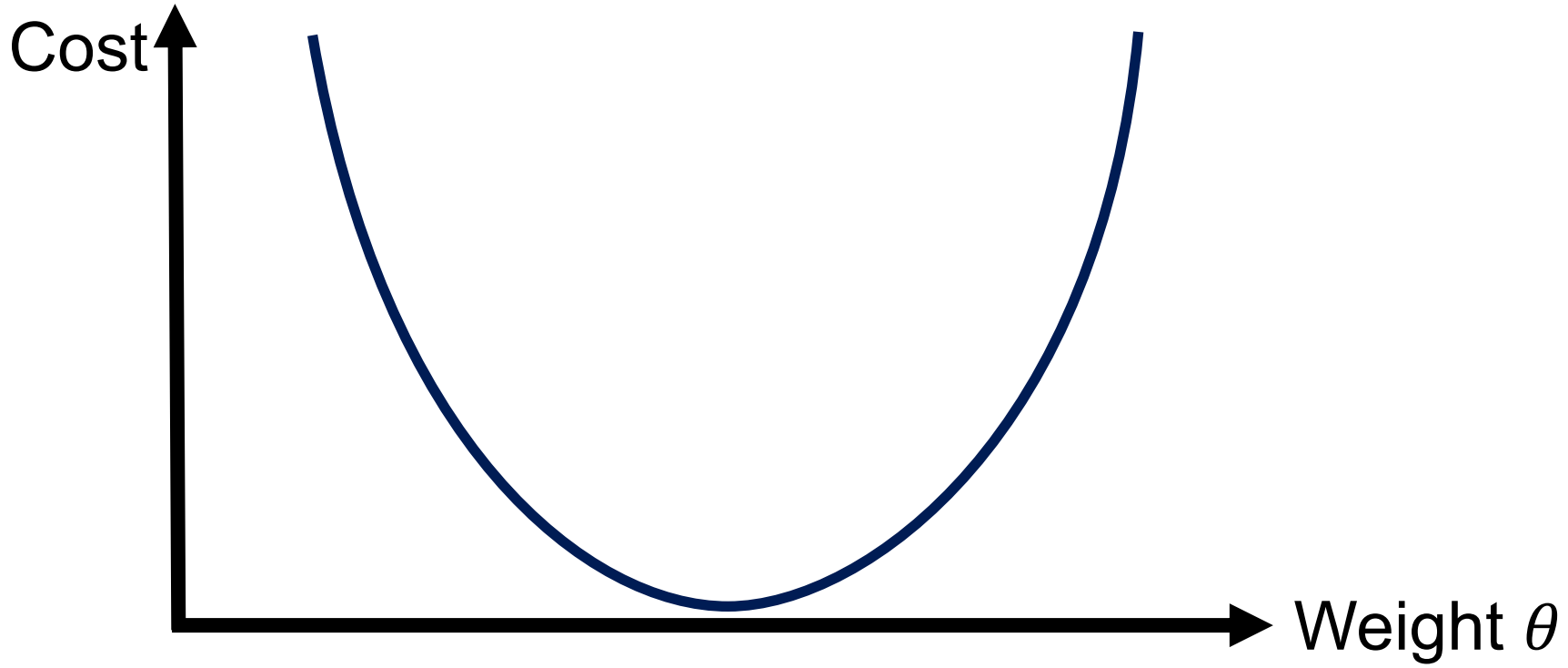
Fast Gradient Sign Method (FGSM)

- Explaining and Harnessing Adversarial Examples, ***ICLR '15***

- One of the most popular method ***to make adversarial examples***
- **Key idea**: move the data x to the sign direction of the gradient to make adversarial example!

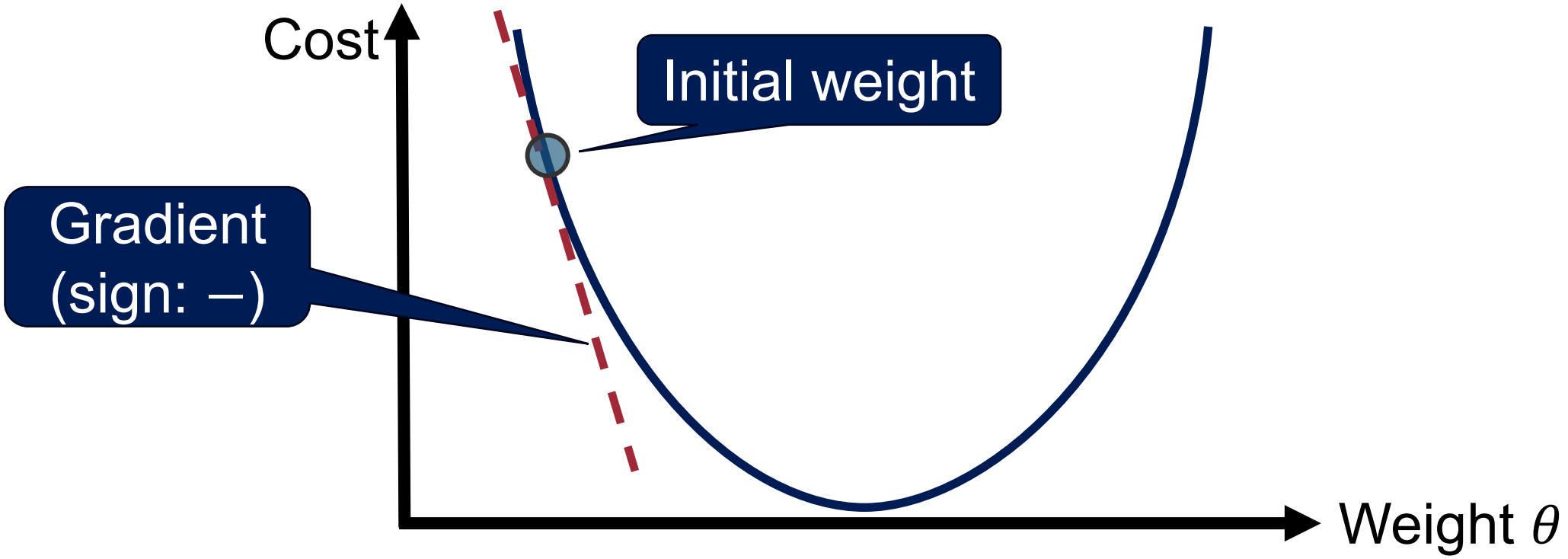
Background: Cost function

- Cost function $f(\theta, x, y)$: the cost of x for class y
  - θ: parameters of a deep neural network
  - x can be represented as an image data



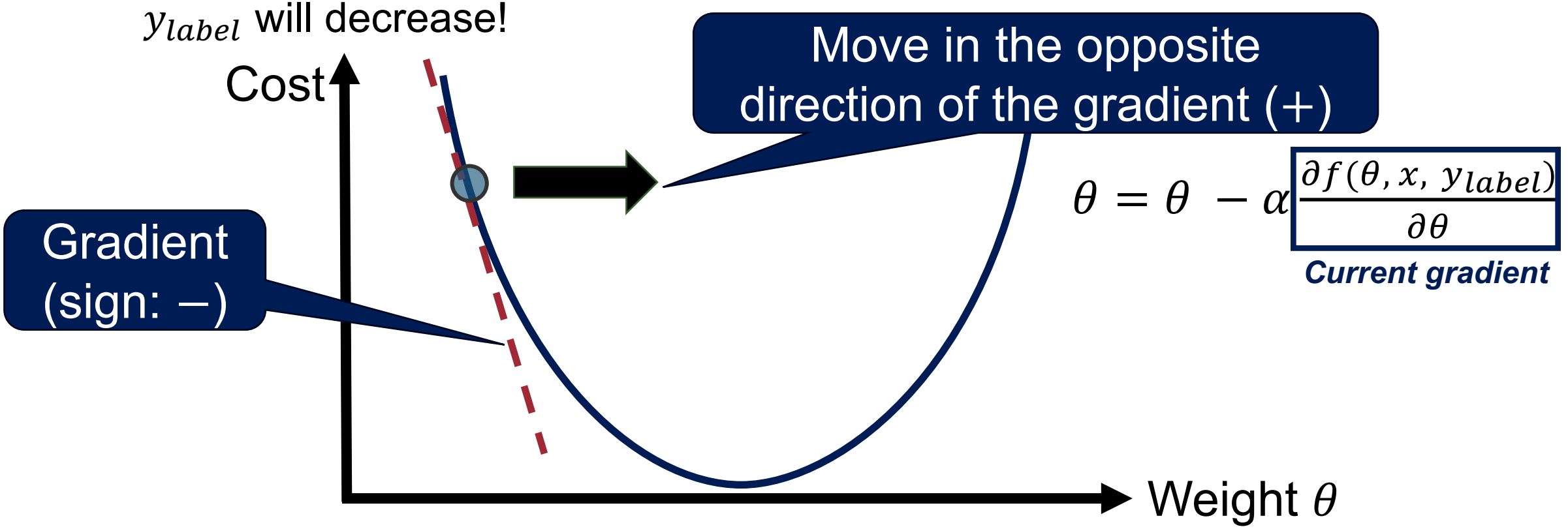
Background: Gradient Descent

- Training process: decrease the cost $f(\theta, x, y_{label})$!
  - Goal: find optimal parameters θ
- How to decrease the cost?
  - If x move to the opposite direction of the gradient, the cost of x for y_{label} will decrease!



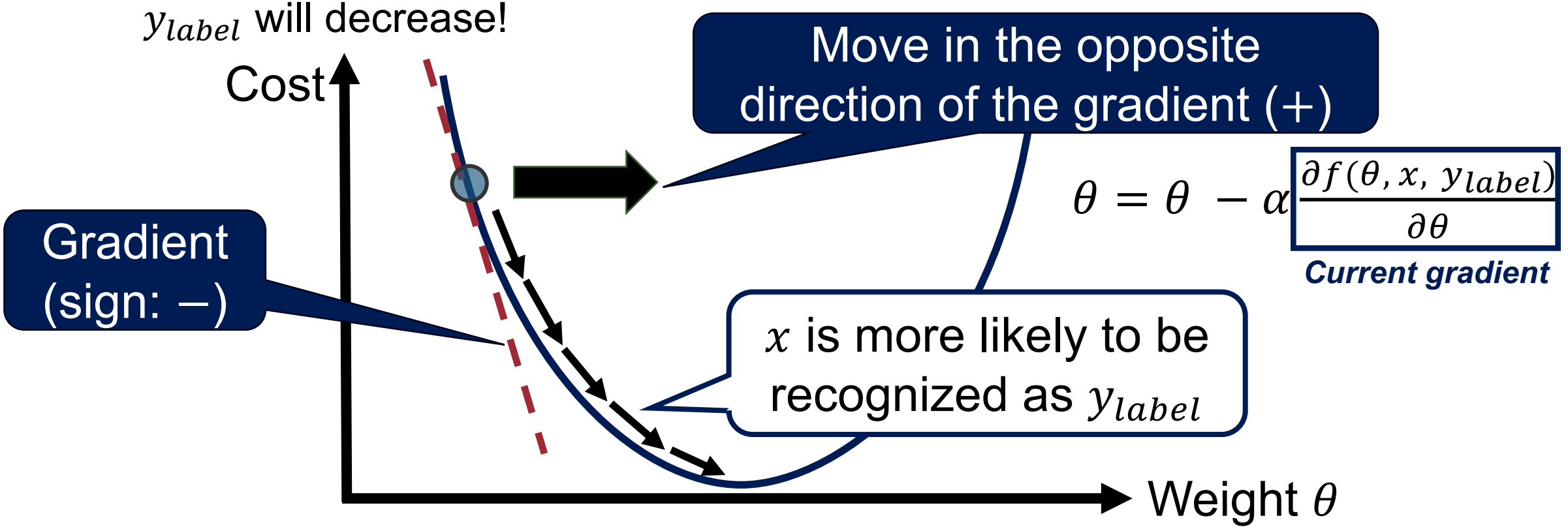
Background: Gradient Descent

- Training process: decrease the cost $f(\theta, x, y_{label})$!
  - Goal: find optimal parameters θ
- How to decrease the cost?
  - If x move to the opposite direction of the gradient, the cost of x for y_{label} will decrease!



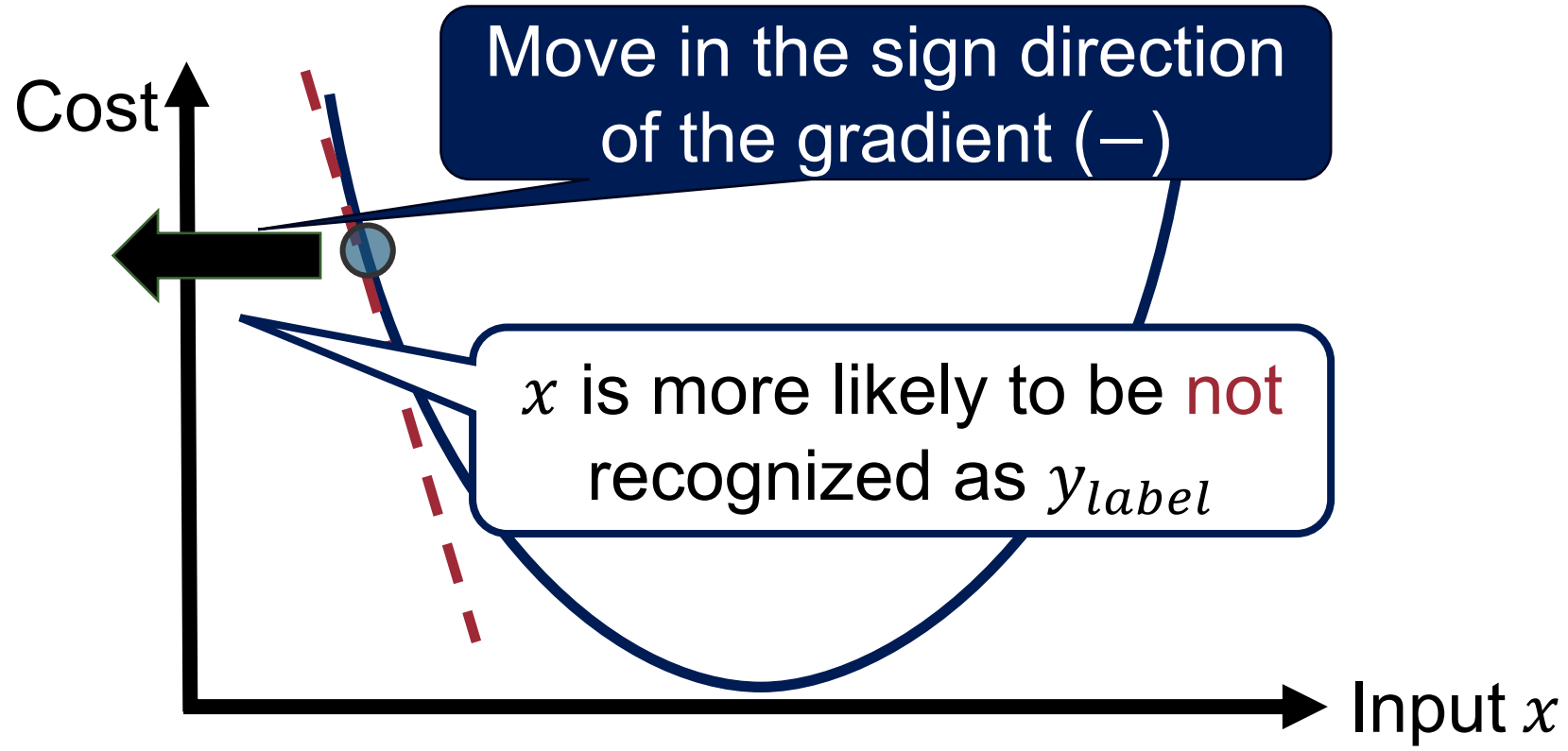
Background: Gradient Descent

- Training process: decrease the cost $f(\theta, x, y_{label})$!
  - Goal: find optimal parameters θ
- How to decrease the cost?
  - If x move to the opposite direction of the gradient, the cost of x for y_{label} will decrease!



FGSM Core Intuition

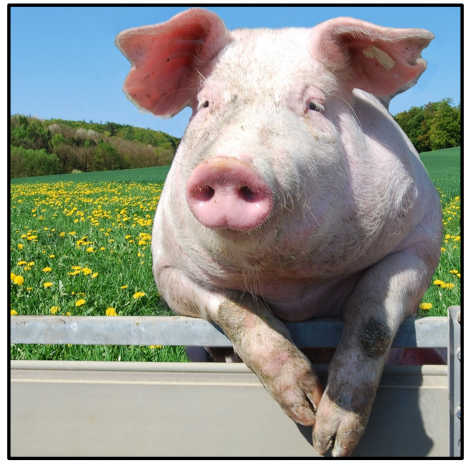
- Idea: increase the cost $f(\theta, x, y_{label})$!
  - Goal: find the input x_{adv} (θ is fixed, i.e., trained)
- If x move to the direction of the gradient, the cost of x for y_{label} will **increase**!



FGSM

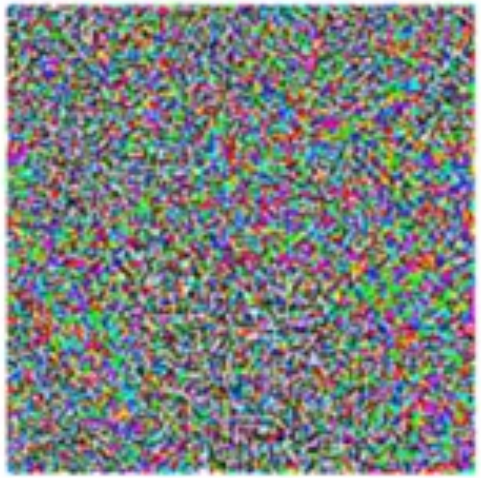
Limit the input pixel changes
E.g., $\epsilon = 0.25$

To make x_{adv} indistinguishable to the human's eye

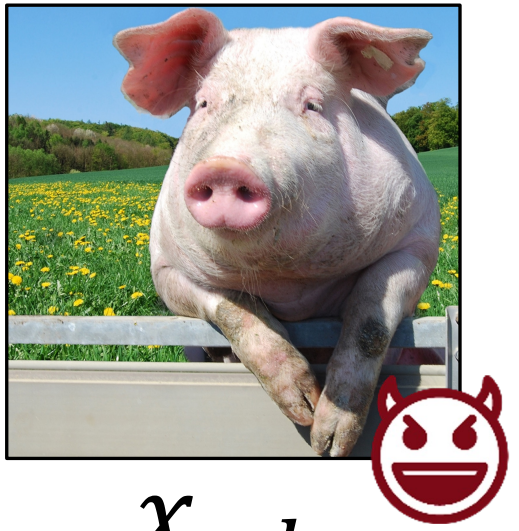


x

$+ \epsilon *$



$= $

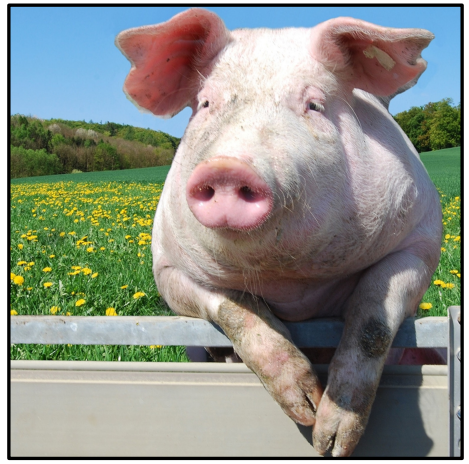


x_{adv}

$$sign(\frac{\partial f(\theta,\ x,\ y_{label})}{\partial x})$$

Gradient for each pixel

FGSM

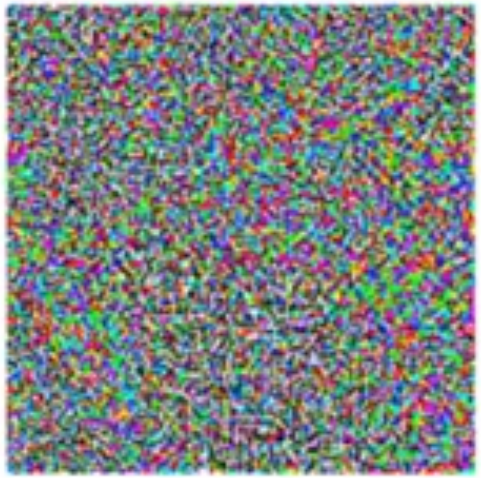


x

“pig”

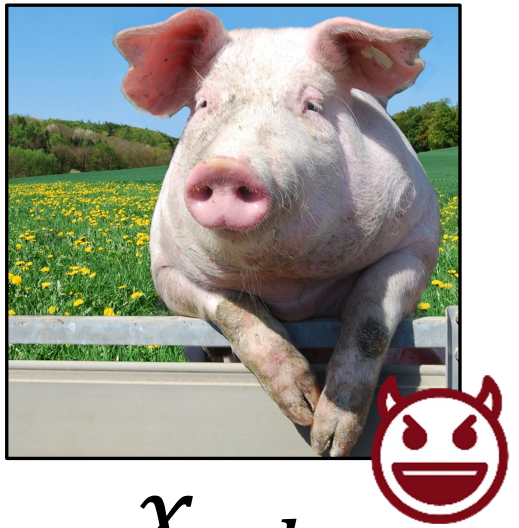
58% confidence

$+ \epsilon *$



$sign(\frac{\partial f(\theta,\ x,\ y_{label})}{\partial x})$

$=$



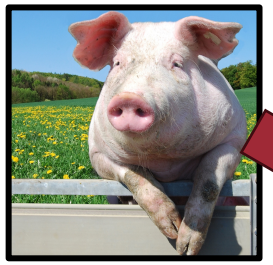
x_{adv}

“airplane”

99.3% confidence

Implications of Eva

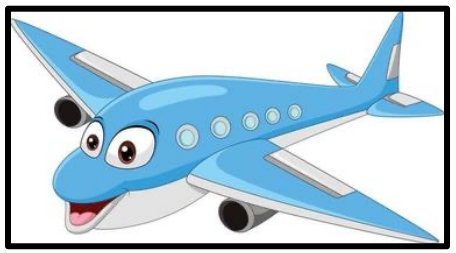
$$\epsilon * sign(\frac{\partial f(\theta,\ x,\ y_{label})}{\partial x})$$



Decision boundary

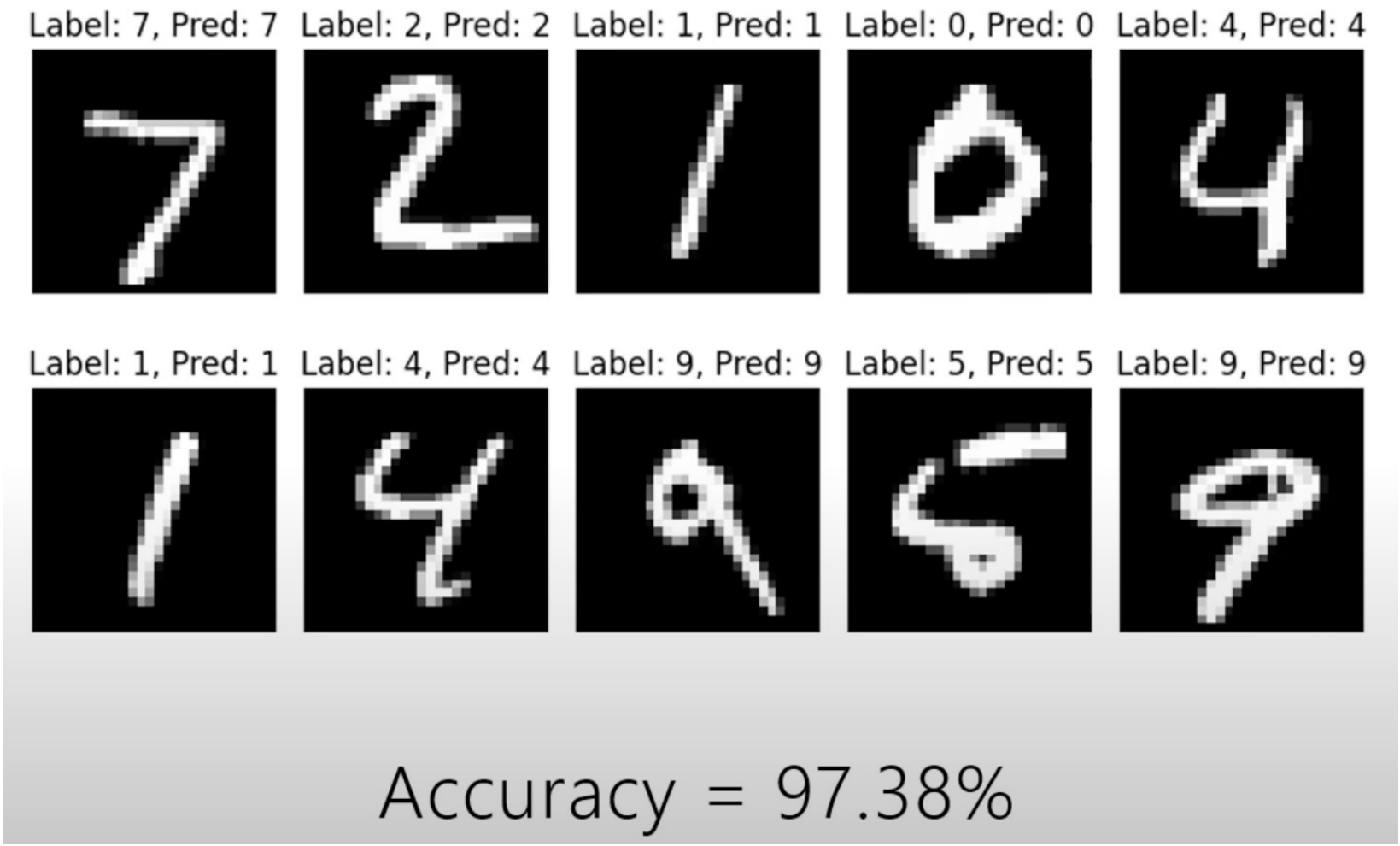
pig

airplane



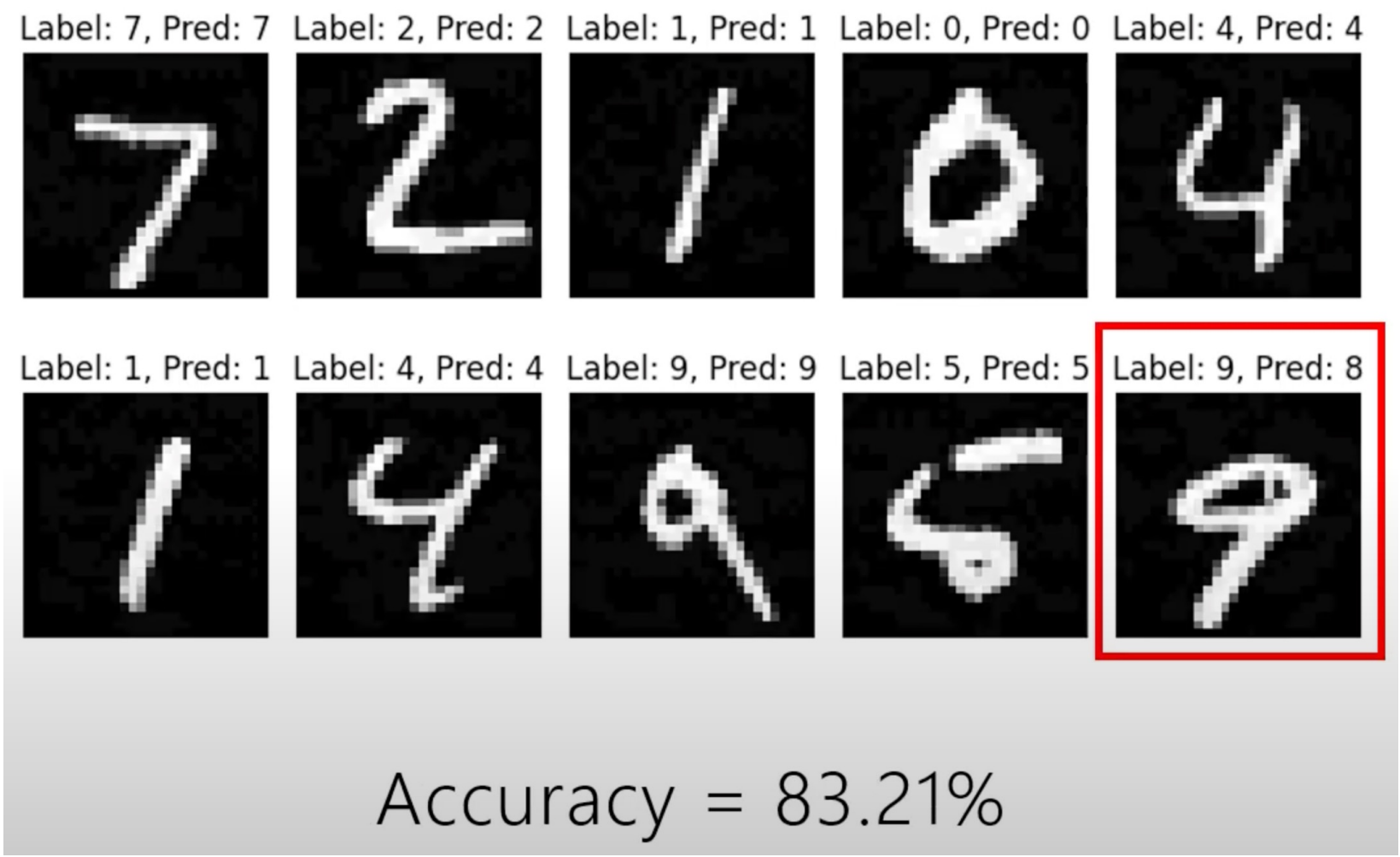
Evaluation of FGSM

- MNIST - Base



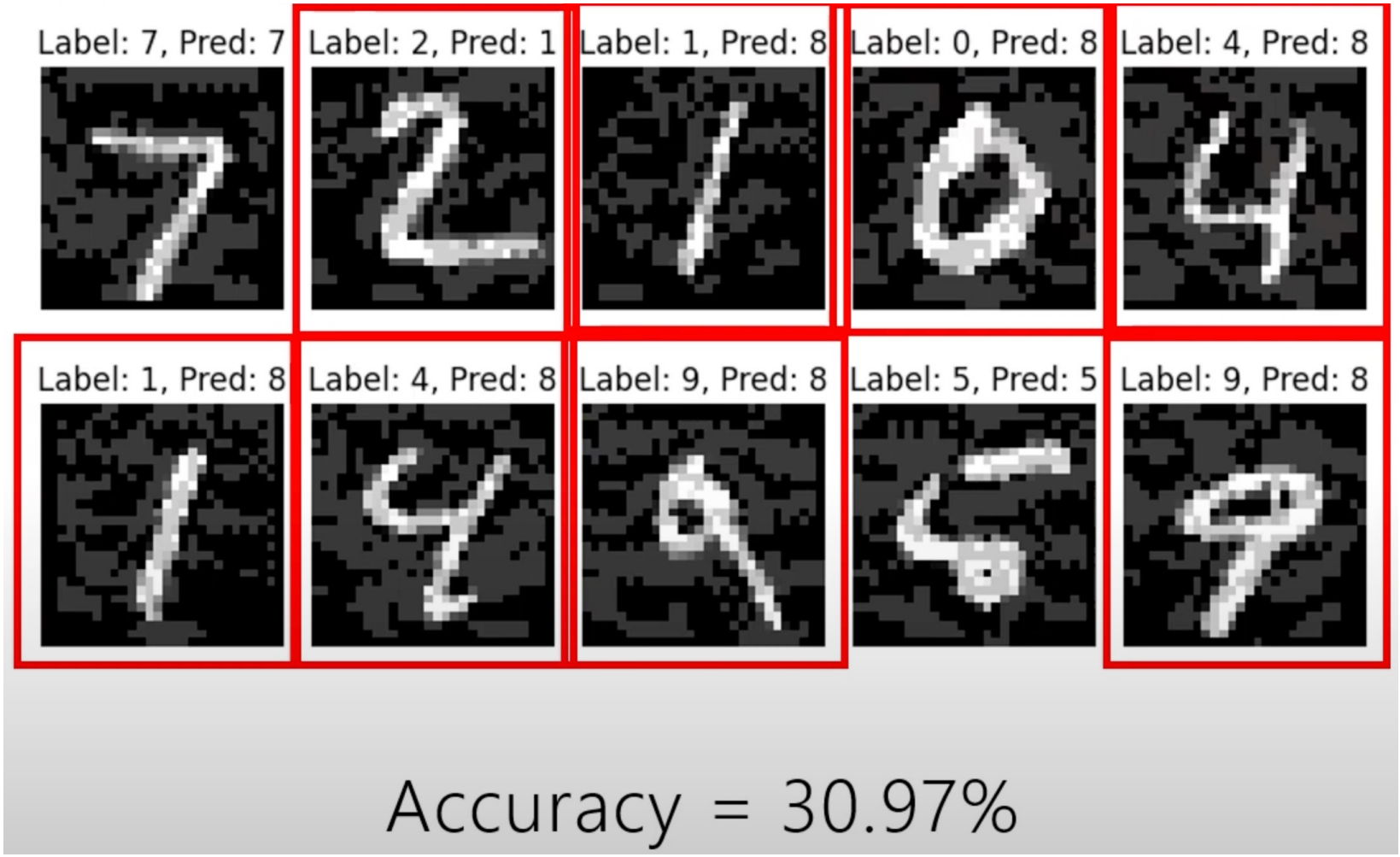
Evaluation of FGSM

- MNIST - $\epsilon = 0.05$



Evaluation of FGSM

- MNIST - $\epsilon = 0.2$



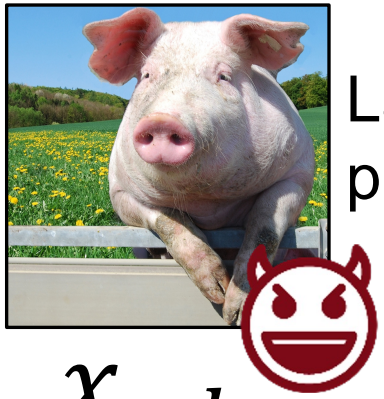
Defense: Adversarial Training

- A method training on a mixture of original and adversarial and examples

Label: pig

x

$$\tilde{f}(\theta, x, y_{label}) = \alpha f(\theta, x, y_{label}) + (1 - \alpha) f(\theta, x_{adv}, y_{label})$$



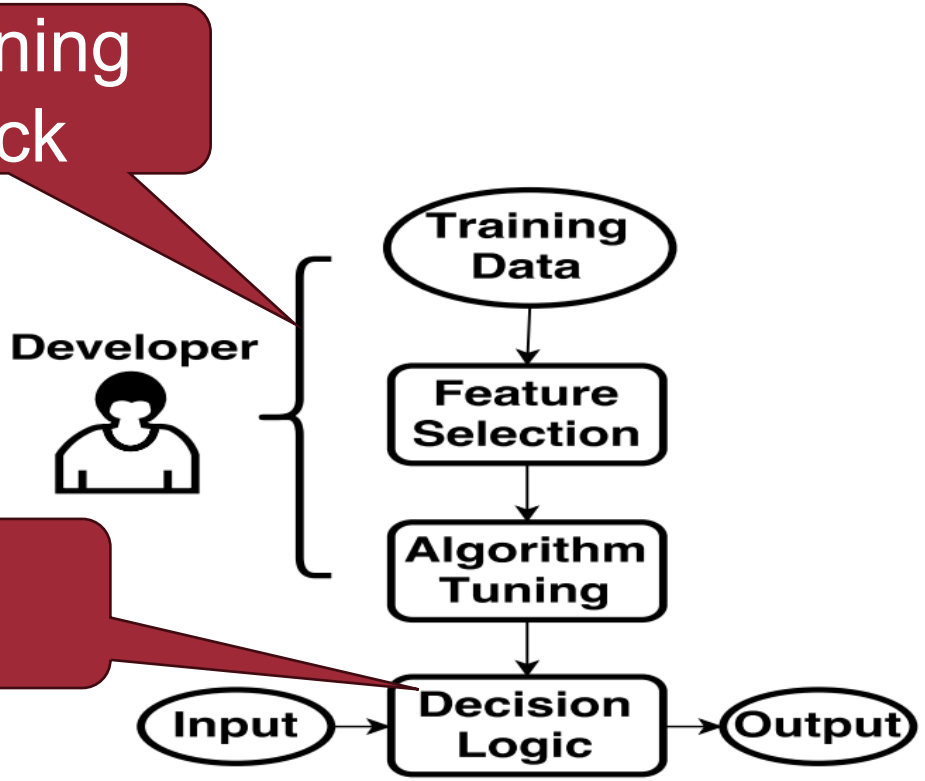
Label: pig

x_{adv}

Adversarial Attacks: Main Attack Scenarios

- Evasion attack
  - The adversary ties to evade the system by adjusting malicious samples
  - Takes place during the ***testing time*** of the machine learning model

- Poisoning attack
  - Contamination of the training data
  - Takes place during the ***training time*** of the machine learning model



Poisoning Attacks

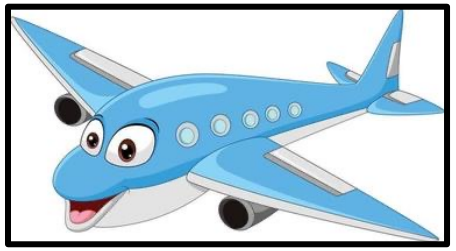
- Take place whenever the machine learning model is under training or during deployment (re-trained during deployment)
- Attackers **influence the data or its labels** when a model is in the training phase
  - Causes system skewed
  - Generates inaccurate decisions in the future

Implications of Poisoning Attacks

Decision boundary

pig

airplane



Example: Tay (chatbot) in Microsoft

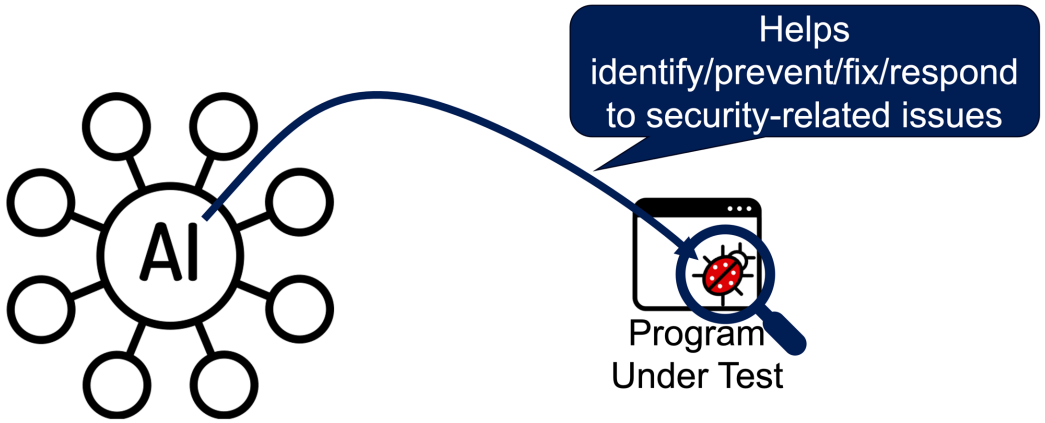
- Microsoft shut down Tay with in less than 24 hours
  - Tay learned how to be racist in less than 24 hours

Conclusion

AI for Security

18

- The tools and strategies that leverage AI to identify, prevent, and respond to emerging cyber threats



Security for AI

21

- The tools and strategies to identify, prevent, and respond to new cyber threats due to the emergence of AI