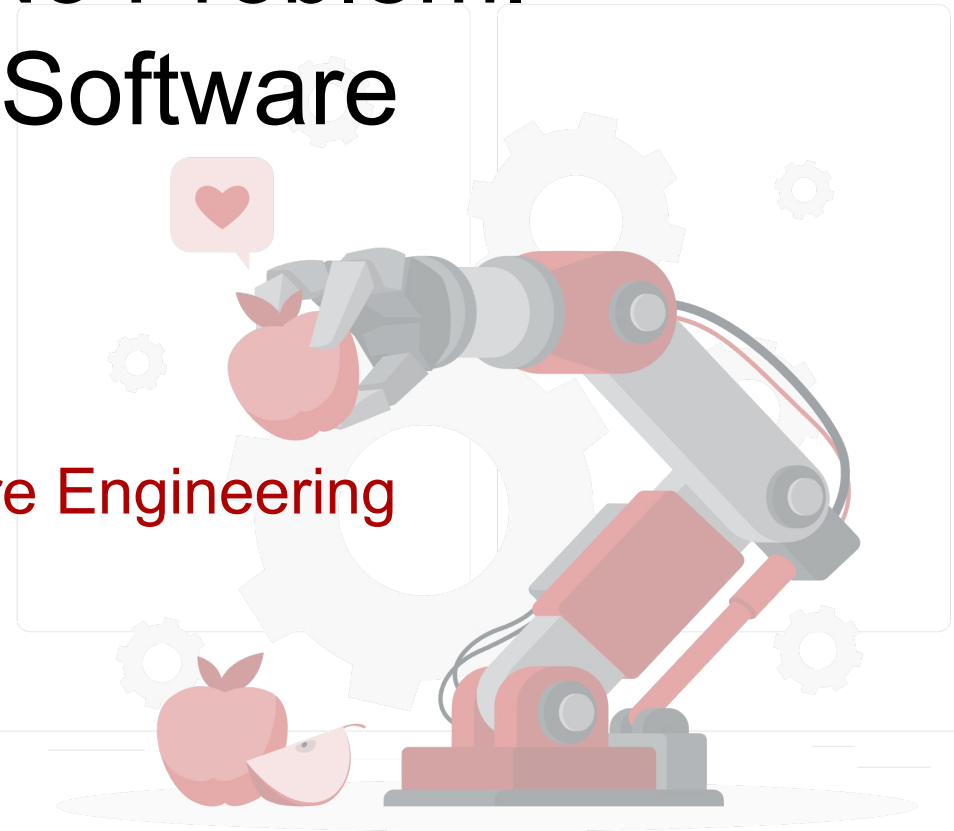
No Ground Truth? No Problem. Testing AI-Enabled Software

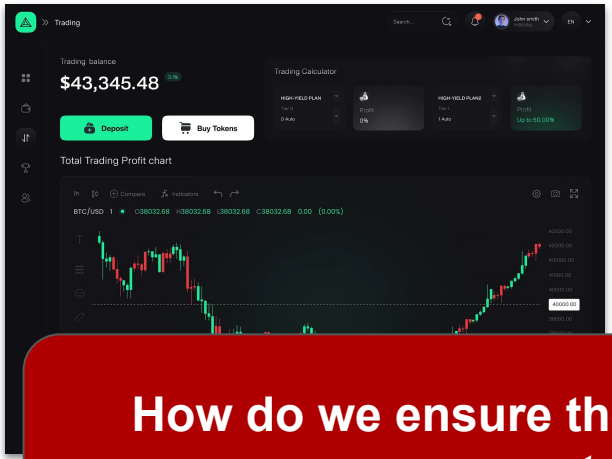


17-313 Foundations of Software Engineering

Shrey Tiwari

Fall 2026

Why should we care?

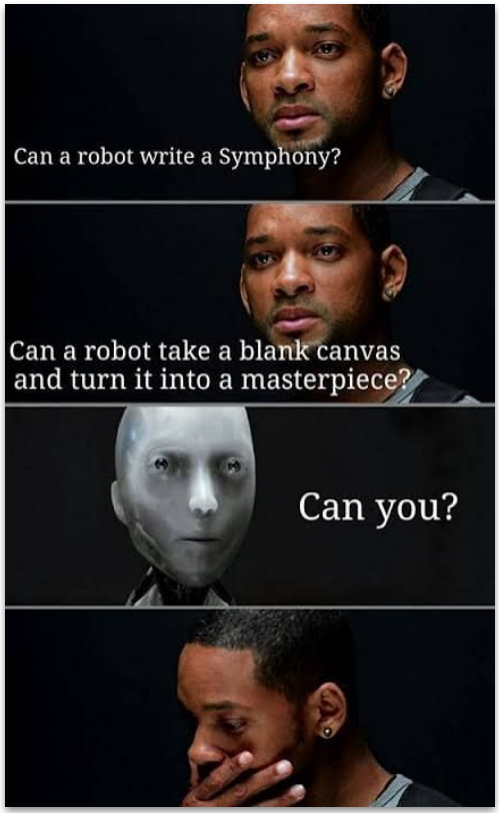




How do we ensure the correctness of these systems??





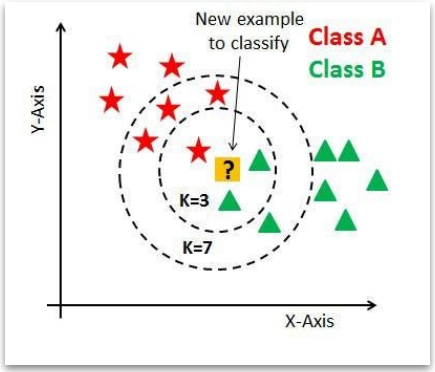


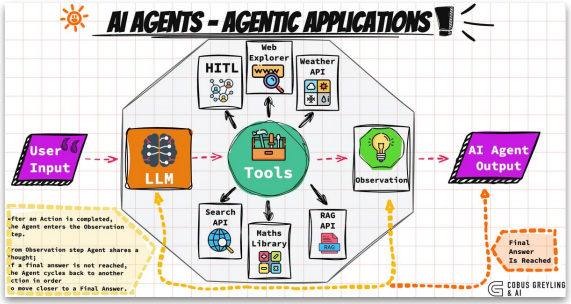
What are AI-enabled systems?

For this lecture, we will treat any software system that uses **AI as one of the architectural components** as an AI-enabled system.

- Language/Text
- Vision
- Audio
- Time series… etc

Testing AI-Enabled Software != Testing AI



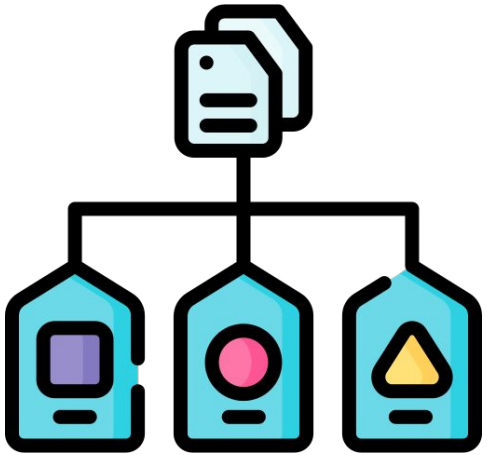


Why are they different from traditional software systems?

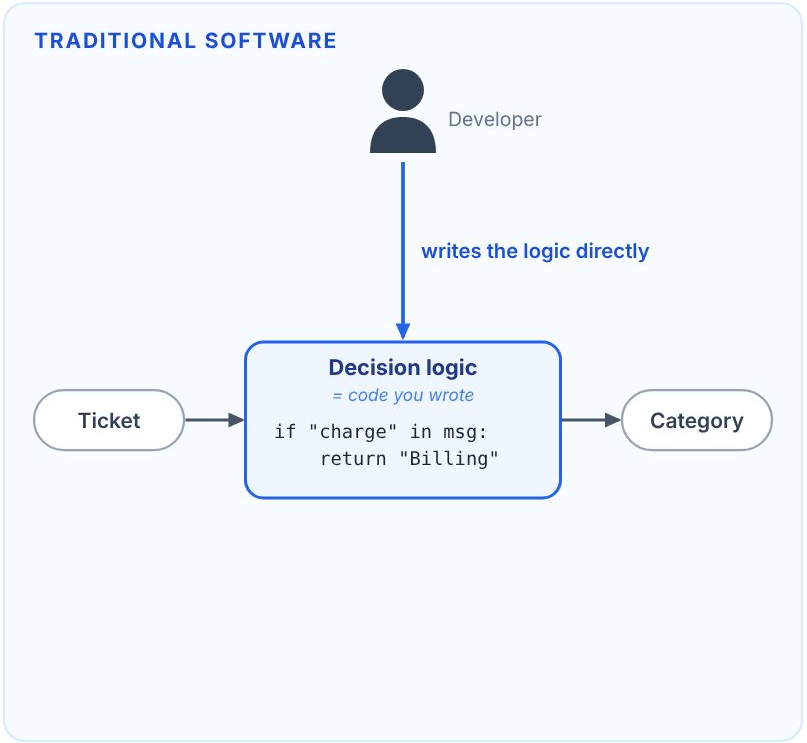
Let's take an example to understand this better!

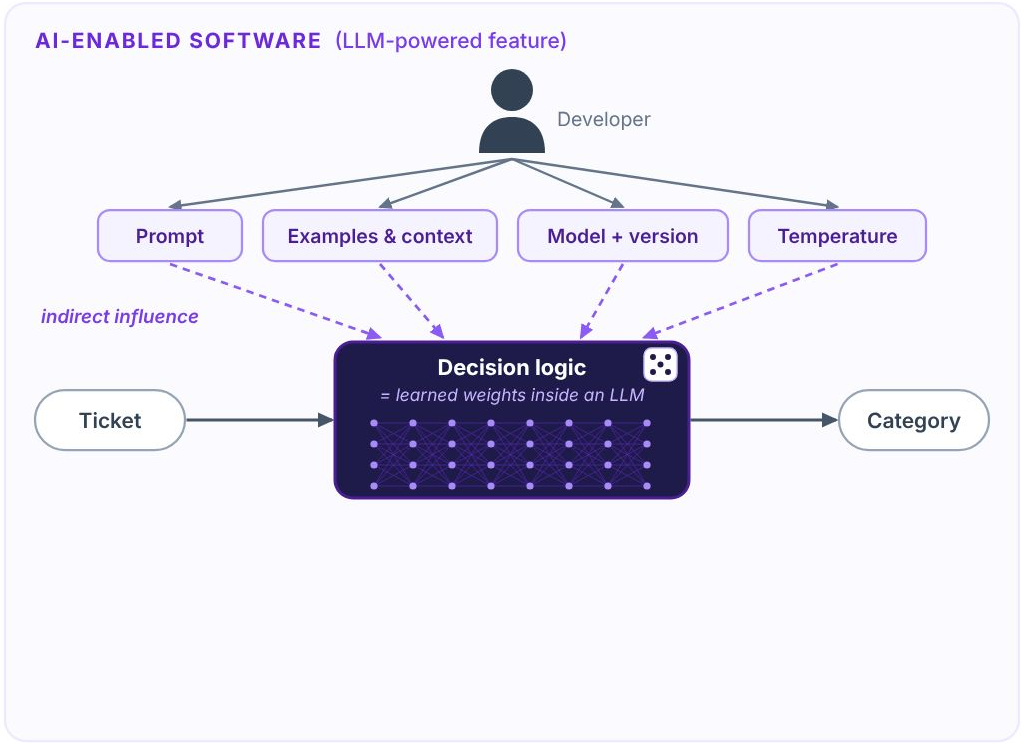
We have a system used to classify incoming support tickets into a set of categories

- **Implementation 1:** Purely traditional rule based system
- **Implementation 2:** Uses an LLM to classify the tickets

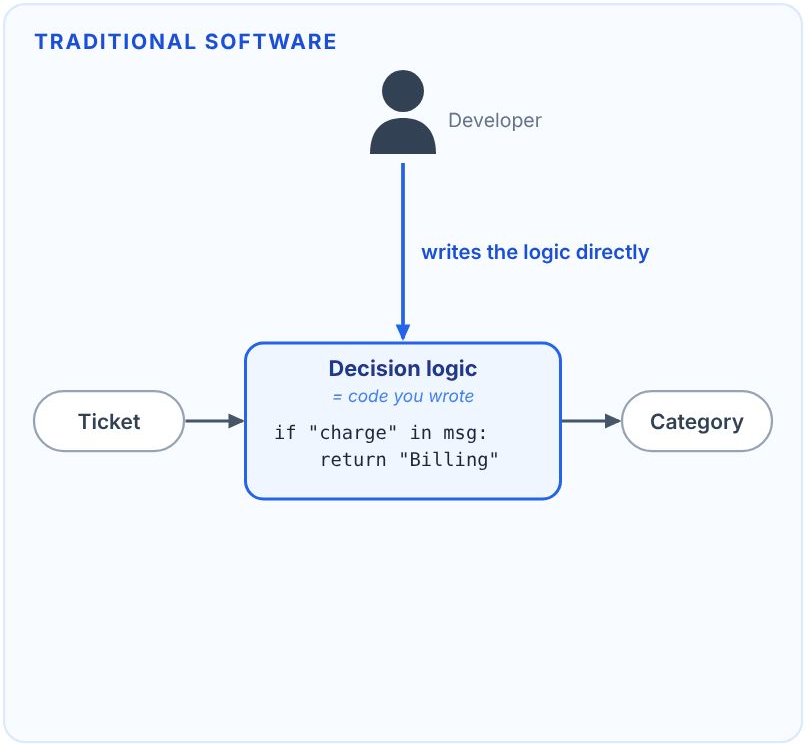


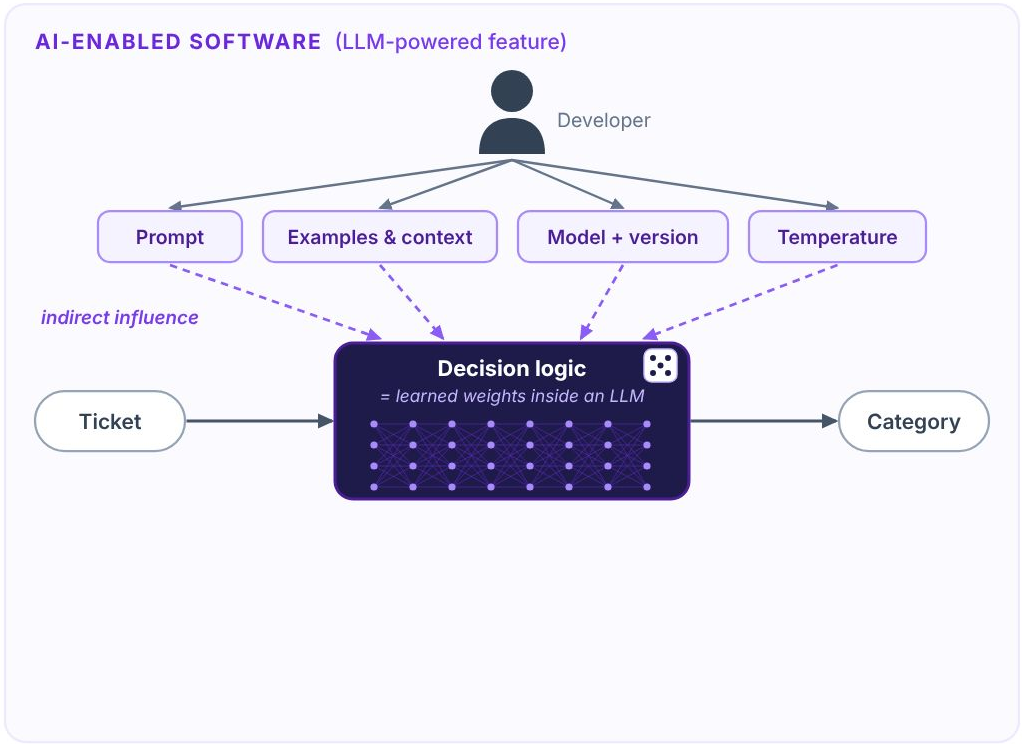
Traditional vs. AI-enabled software



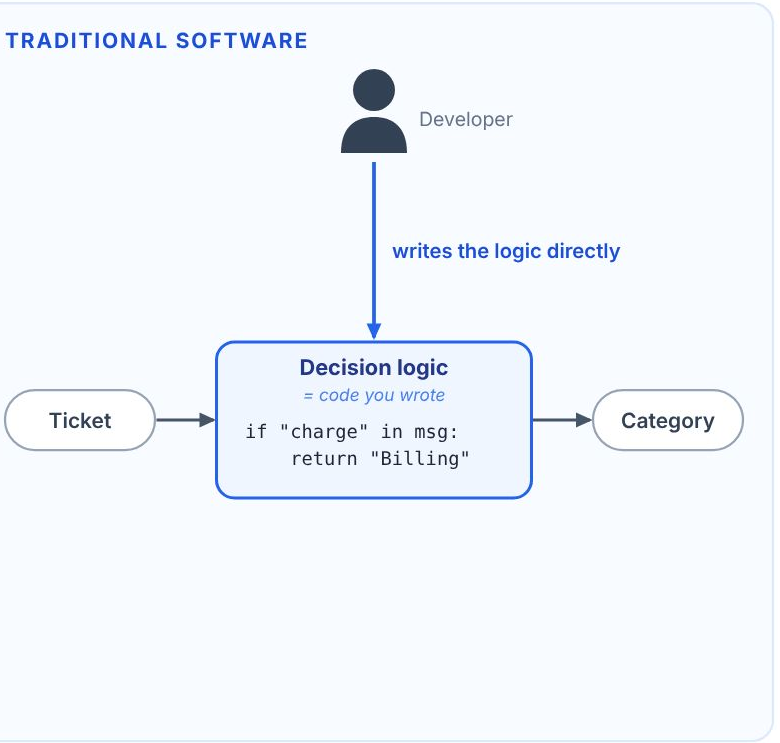


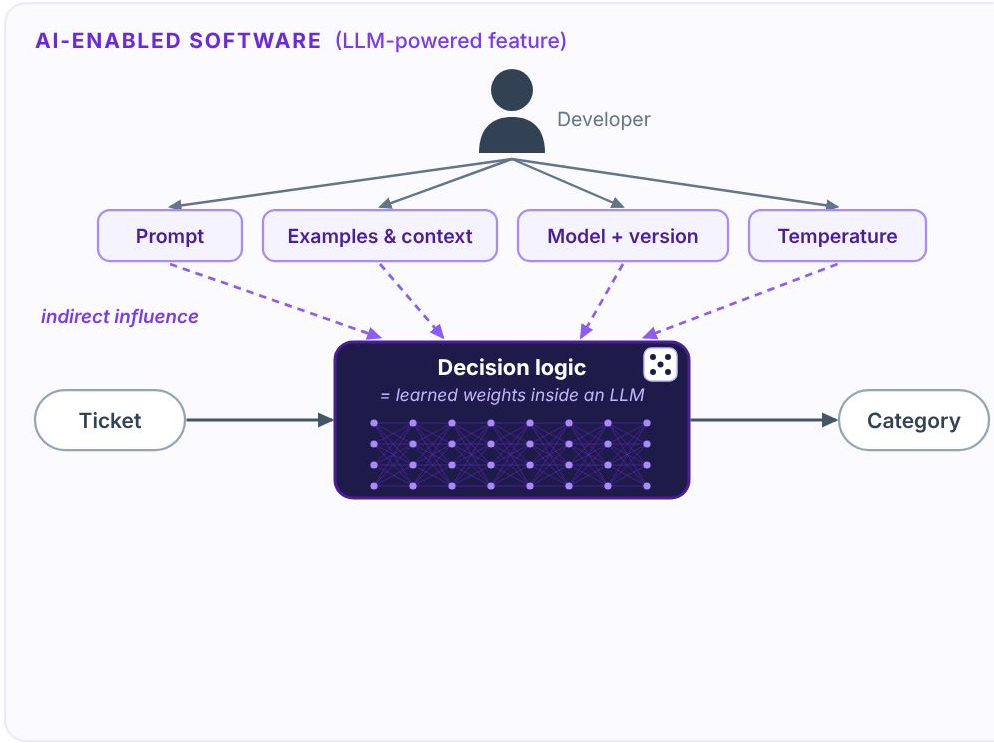
Traditional vs. AI-enabled software



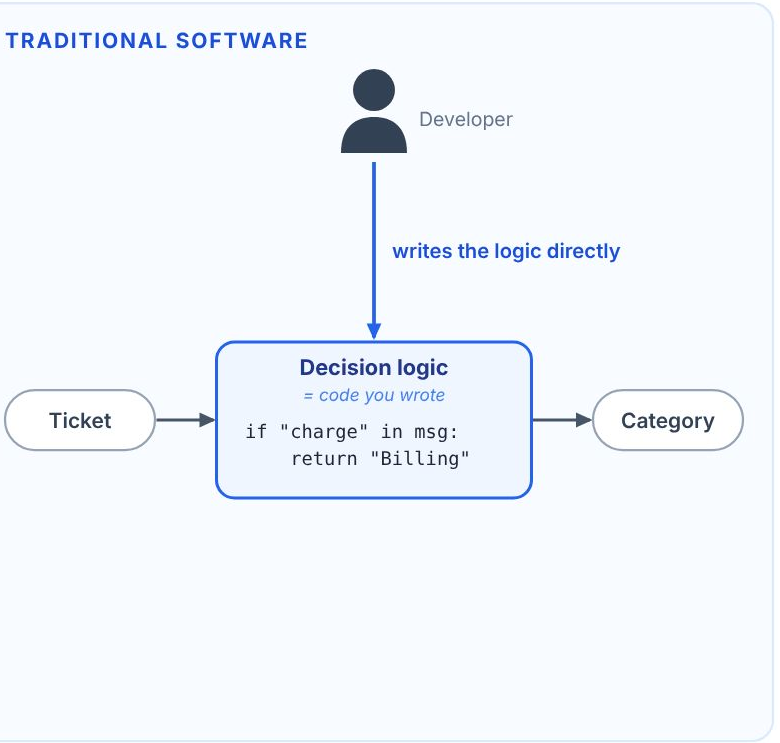


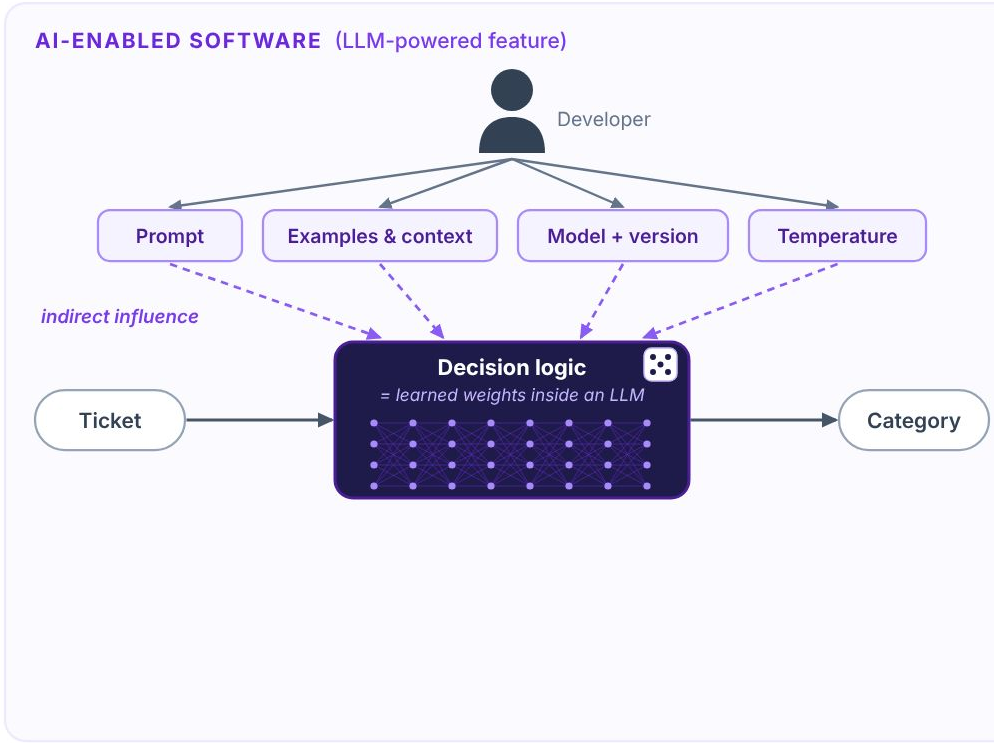
Traditional vs. AI-enabled software



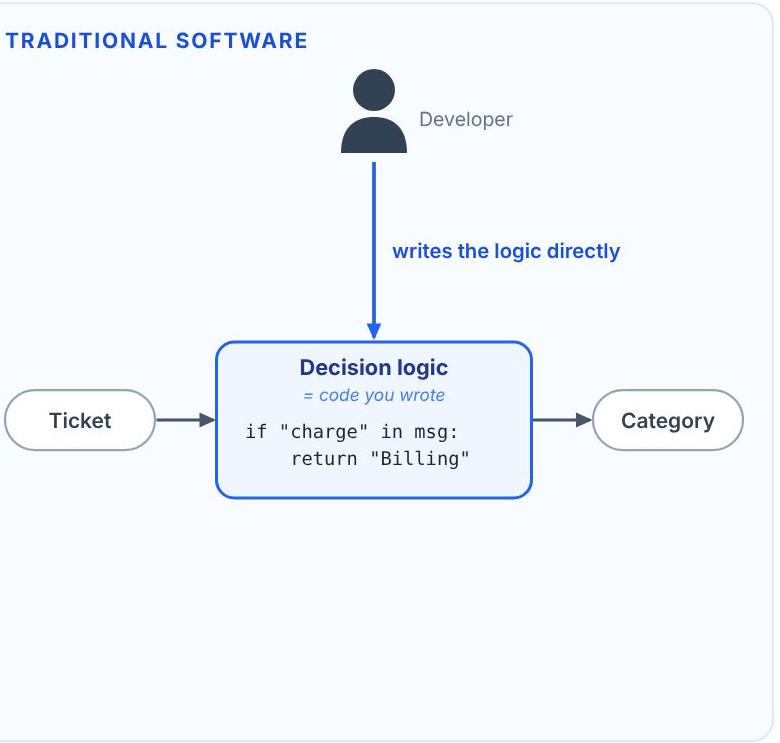


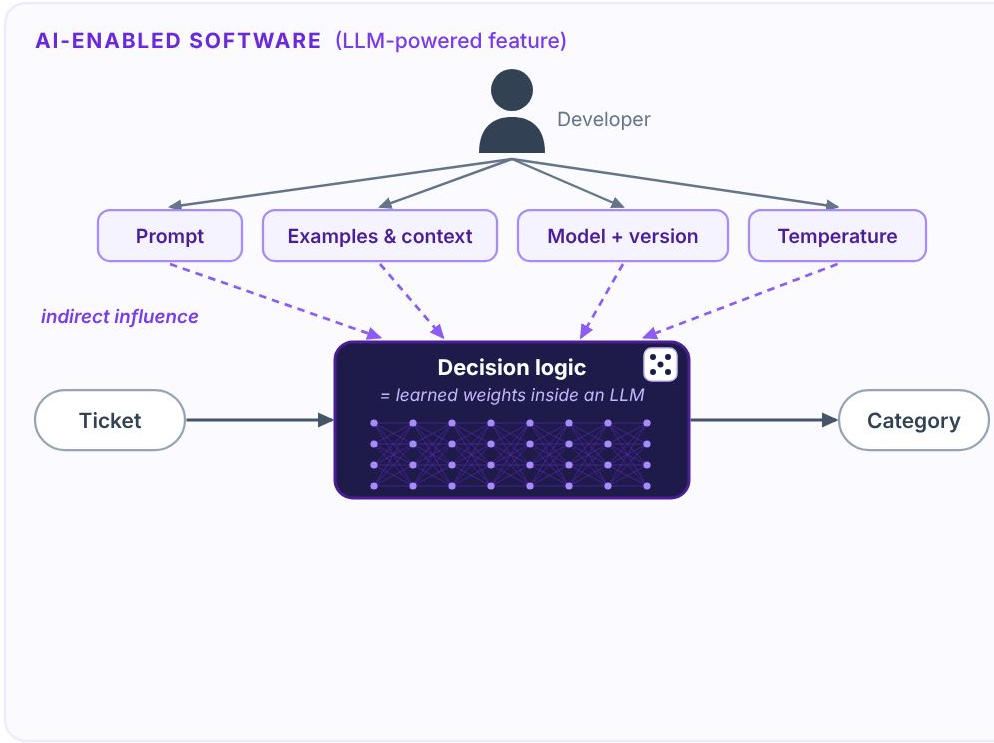
Traditional vs. AI-enabled software



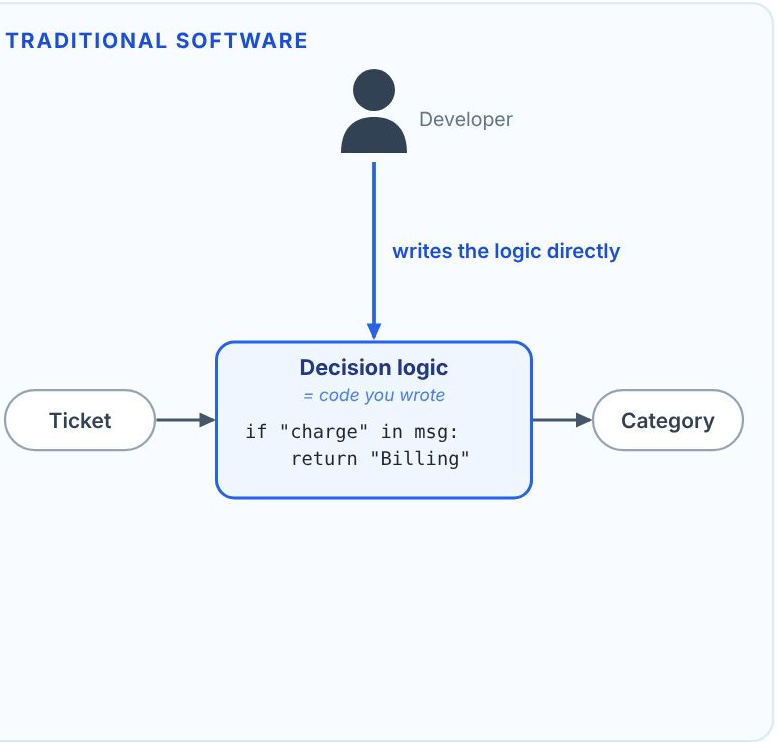


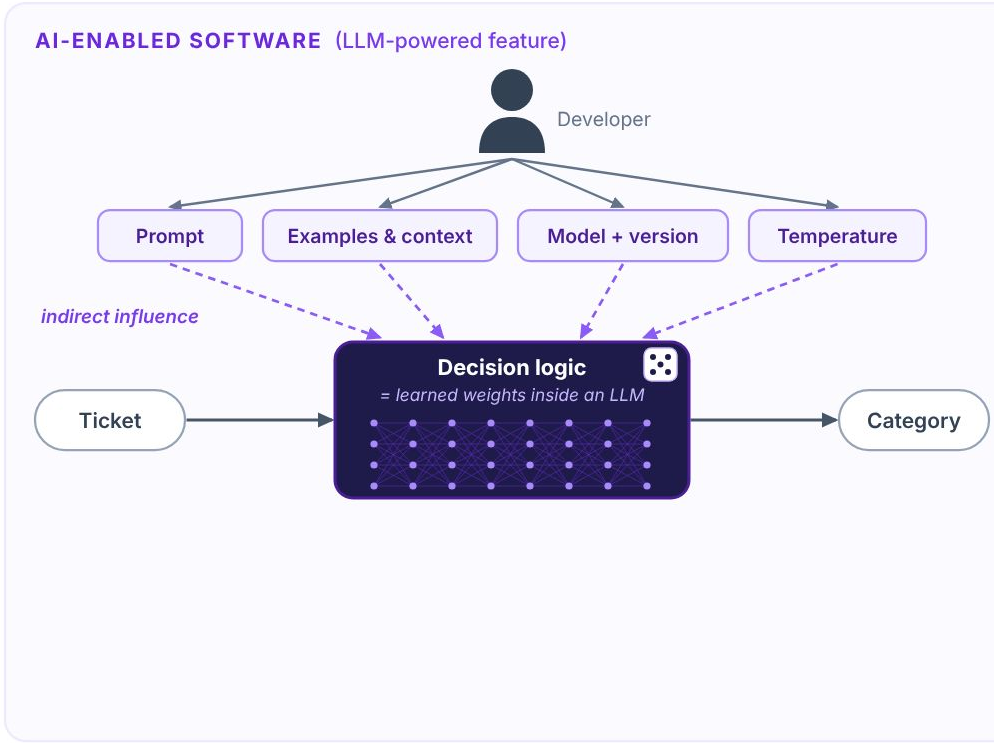
Traditional vs. AI-enabled software





Traditional vs. AI-enabled software

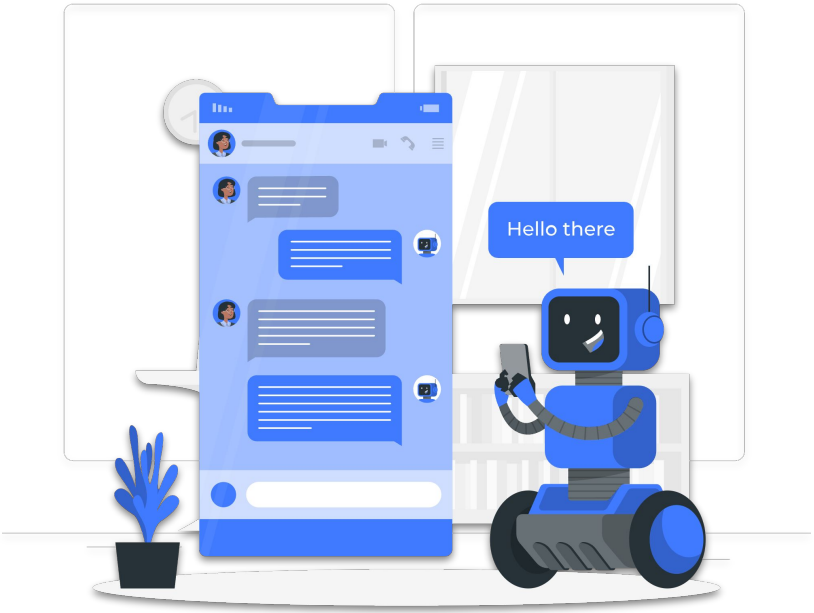




Two examples for today…



Vision: Autonomous Cars



Text: AI Code Reviews

Vision-based AI in software systems

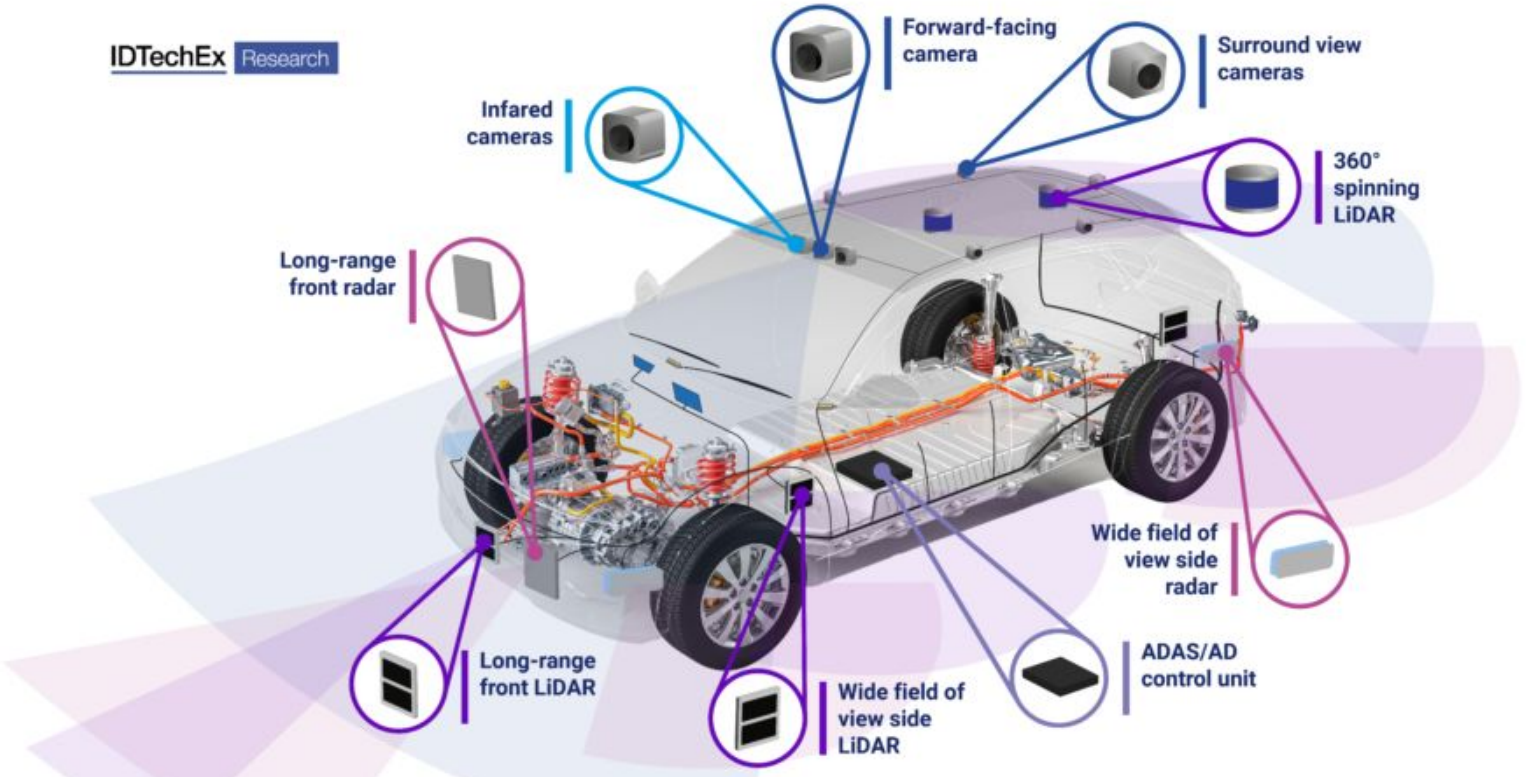
AI-powered vision* systems in autonomous vehicles and robotics are becoming more and more **pervasive!**

- Autonomous cars
- Humanoid robots
- Drones
- Smart vacuum cleaners

They must be rigorously tested to guarantee safety and reliability. Failures in perception can lead to dangerous situations.



The complexity is real.





Problem with checking correctness…

Unlike traditional software, these systems learn their logic from data, making their behavior less predictable and harder to validate.

- Given certain conditions, what should be the steering angle?
- Is it okay if the car decides to not reduce its speed over a small bump?
- What is correct behavior if the car is not designed to sense certain kinds of things?



Tesla Deaths

We provide an updated record of Tesla fatalities and Tesla accident deaths that have been reported and as much related crash data as possible (e.g. location of crash, names of deceased, etc.). This sheet also tallies each claimed and confirmed Tesla autopilot crash, i.e. every instance of when Autopilot was activated during a Tesla crash that resulted in death. Read our other sheets for additional data and analysis on Tesla crashes per vehicle miles traveled, links and analysis comparing Musk's safety claims, and more.

Tesla Deaths Total: **674** | Tesla Autopilot Deaths Count: **54, including 2 fatalities involving the use of FSD** | Other Totals

Updated on 2025-03-09: Sourced Tesla and Autopilot fatalities for early 2025. Reviewed latest SGO data.

Read in:

Deaths Sudden Accelerations Miles Musk claims Resources Vehicle Years FAQ A note to contributors Contact Us Tesla Fire🔥 Download Verified Autopilot Data🚗 Deaths Charts📈

Case #	Year	Date	Country	State	Description	Deaths	Tesla driver	Tesla occupant	Other vehicle	Cyclists/ Peds	TSLA+cycl / peds	Model	AutoPilot claimed	Reported in NHTSA SGO	Verified Tesla Autopilot Death	Excerpt Verifying Tesla Autopilot Deaths	Sou
492	2025	3/3/2025	USA	CA	Tesla turns and hits motorcycle	1	-	-	1	-	-	-	-	-	-	-	http
491	2025	3/3/2025	USA	FL	Car head-on collides with Tesla	1	1	-	-	-	1	3	-	-	-	-	http
490	2025	3/2/2025	USA	IL	Tesla crashes into suburban house	2	1	1	-	-	2	-	-	-	-	-	http
489	2025	2/25/2025	USA	CA	Tesla crashes into tree, ignites	1	1	-	-	-	1	-	-	-	-	-	http
488	2025	2/25/2025	USA	NJ	Tesla veers into opposite lane, hits dump truck	1	1	-	-	-	1	-	-	-	-	-	http
487	2025	2/24/2025	USA	CA	Juvenile driving Tesla hits truck	1	1	-	-	-		-	-	-	-	-	http
486	2025	2/21/2025	USA	NV	Scooter hit by Tesla	1	-	-	1	-	-	-	-	-	-	-	http
485	2025	2/15/2025	USA	WA	Tesla hits wall	1	1										http
484	2025	2/13/2025	USA	OK	Tesla crashes into creek	2	-	2	-	-	2	-	-	-	-	-	http
483	2025	2/10/2025	USA	MO	Tesla crashes into trees, explodes	2	1	1	-	-	2	S	-	-	-	-	http
482	2025	2/5/2025	USA	IL	Tesla drives into creek	2	-	2	-	-	2	3	-	-	-	-	http
481	2025	2/4/2025	USA	NY	Tesla hits guardrail, ignites	1	1	-	-	-	1	-	-	-	-	-	http
480	2025	1/28/2025	USA	CA	Tesla hits telephone pole, ignites	1	1	-	-	-	1	Y	-	-	-	-	http
479	2025	1/24/2025	USA	CA	5-car vehicle crash	1	-	-	1	-	-	-	-	-	-	-	http
478	2025	1/19/2025	USA	CA	7-vehicle accident caused by Tesla	1	-	-	1	-	-	-	-	-	-	-	http
477	2025	1/14/2025	USA	NY	Tesla hits guardrail	1	1	-	-	-	-	-	-	-	-	-	http
476	2025	1/13/2025	USA	TX	Pickup hits Tesla	1	-	-	1	-	-	-	-	-	-	-	http

In-class activity question

Let’s say you need to test an autonomous car. What are some challenges that you foresee when it comes to testing such an AI-enabled system?

Challenges with testing vision

- **Biases and Missing Corner Cases:** Vision DNNs learn from finite training data, which may not cover the long tail of rare scenarios.

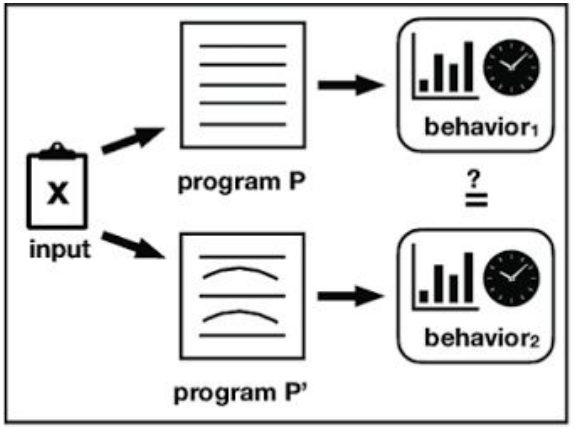
- **Unpredictable Environmental Conditions:** Autonomous systems face vast domain variability – lighting, weather, sensor noise, or viewpoint changes can shift the input distribution.

- **Lack of Specifications and Oracles:** It is difficult to formally specify the correct behavior of a complex vision model for every possible input.

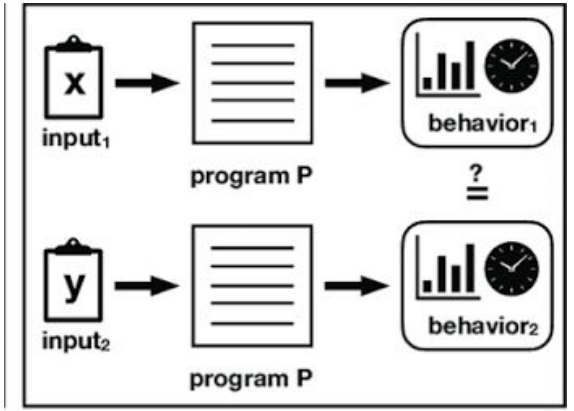
- **Interpretability and Debuggability:** Vision DNNs act as high-dimensional black boxes, making it hard for engineers to understand why a system fails a test.

Current landscape

- **Adversarial / robustness testing**
- **Differential testing**
- **Metamorphic testing**
- **Simulation-based validation and formal analysis**



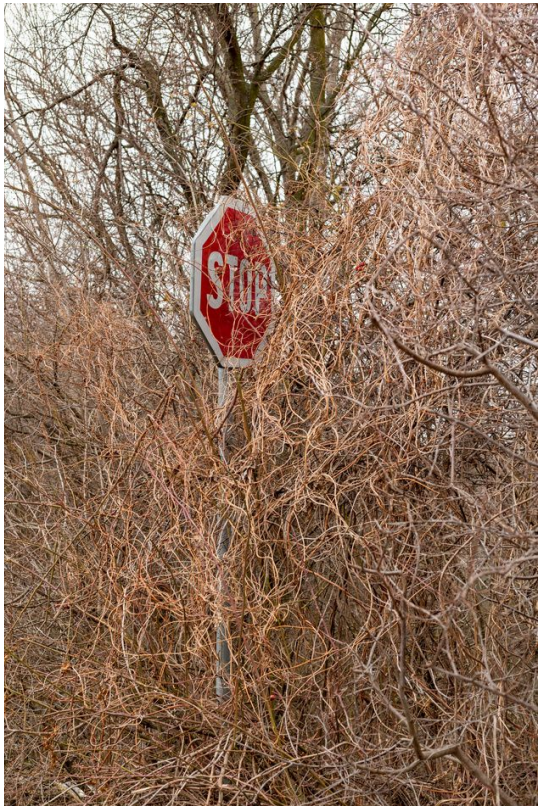
1



2

Adversarial / Robustness Testing

Task at hand: identify road signs







How do we test if our car identifies signs correctly?

Adversarial testing: we test our system in a way that tries to break it!

To perform adversarial testing, we usually need an input set of values and the corresponding expected outputs.

Test Oracle: Minor changes to inputs should not affect the outputs.



Detected: **Stop**

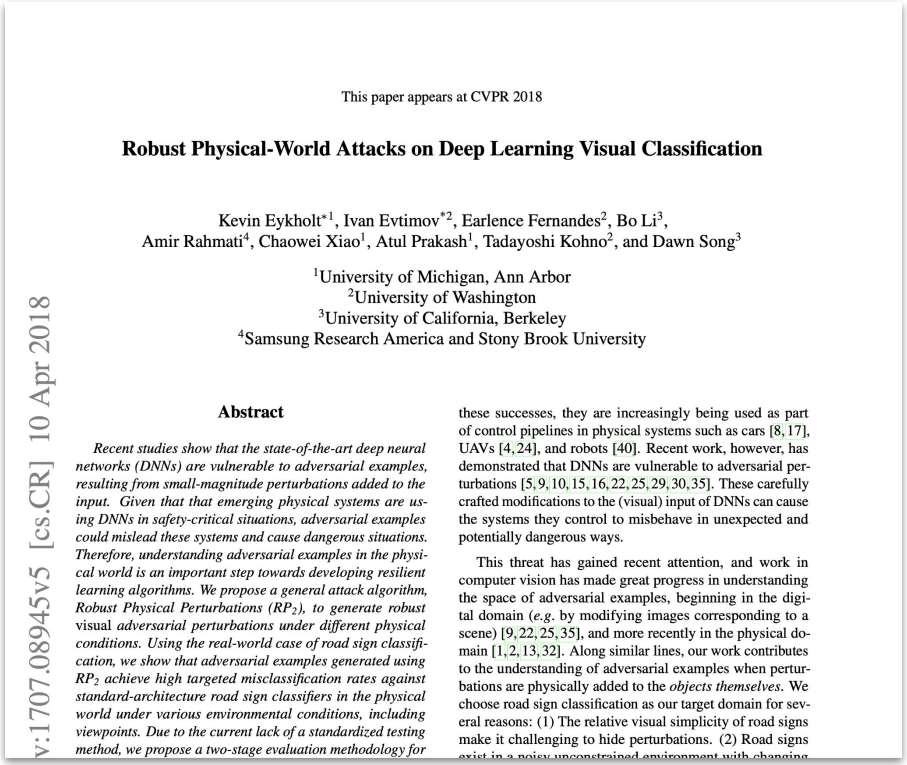


Detected: **Turn Left**

Adversarial Attacks

It is possible to carry out adversarial attacks on Deep Neural Networks (DNNs) used by most vision based systems that could lead to serious situations.

This paper presents **realistic adversarial attacks** carried out on autonomous vehicles that use vision for road sign detection.



v:1707.08945v5 [cs.CR] 10 Apr 2018

This paper appears at CVPR 2018

Robust Physical-World Attacks on Deep Learning Visual Classification

Kevin Eykholt*[1], Ivan Evtimov*[2], Earlence Fernandes[2], Bo Li[3],
Amir Rahmati[4], Chaowei Xiao[1], Atul Prakash[1], Tadayoshi Kohno[2], and Dawn Song[3]

[1]University of Michigan, Ann Arbor
[2]University of Washington
[3]University of California, Berkeley
[4]Samsung Research America and Stony Brook University

Abstract

Recent studies show that the state-of-the-art deep neural networks (DNNs) are vulnerable to adversarial examples, resulting from small-magnitude perturbations added to the input. Given that that emerging physical systems are using DNNs in safety-critical situations, adversarial examples could mislead these systems and cause dangerous situations. Therefore, understanding adversarial examples in the physical world is an important step towards developing resilient learning algorithms. We propose a general attack algorithm, Robust Physical Perturbations (RP_2), to generate robust visual *adversarial perturbations under different physical conditions. Using the real-world case of road sign classification, we show that adversarial examples generated using RP_2 achieve high targeted misclassification rates against standard-architecture road sign classifiers in the physical world under various environmental conditions, including viewpoints. Due to the current lack of a standardized testing method, we propose a two-stage evaluation methodology for*

these successes, they are increasingly being used as part of control pipelines in physical systems such as cars [8, 17], UAVs [4, 24], and robots [40]. Recent work, however, has demonstrated that DNNs are vulnerable to adversarial perturbations [5, 9, 10, 15, 16, 22, 25, 29, 30, 35]. These carefully crafted modifications to the (visual) input of DNNs can cause the systems they control to misbehave in unexpected and potentially dangerous ways.

This threat has gained recent attention, and work in computer vision has made great progress in understanding the space of adversarial examples, beginning in the digital domain (*e.g.* by modifying images corresponding to a scene) [9, 22, 25, 35], and more recently in the physical domain [1, 2, 13, 32]. Along similar lines, our work contributes to the understanding of adversarial examples when perturbations are physically added to the *objects themselves*. We choose road sign classification as our target domain for several reasons: (1) The relative visual simplicity of road signs make it challenging to hide perturbations. (2) Road signs exist in a noisy unconstrained environment with changing

Robust physical perturbations

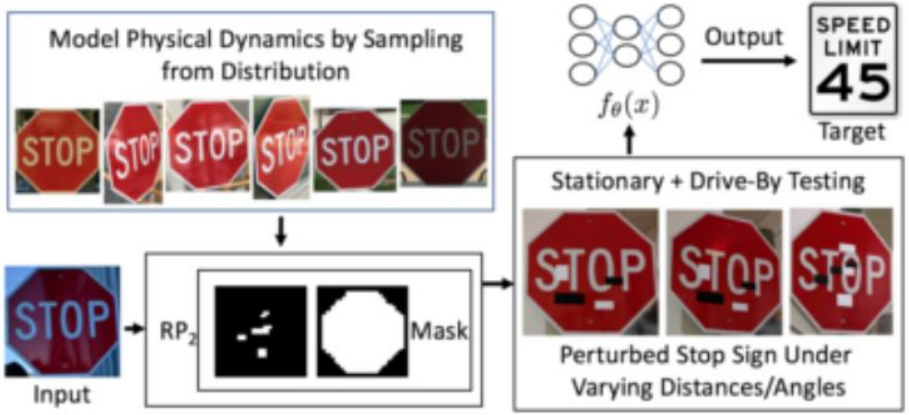


Figure 2: RP_2 pipeline overview. The input is the target Stop sign. RP_2 samples from a distribution that models physical dynamics (in this case, varying distances and angles), and uses a mask to project computed perturbations to a shape that resembles graffiti. The adversary prints out the resulting perturbations and sticks them to the target Stop sign.



Figure 1: The left image shows real graffiti on a Stop sign, something that most humans would not think is suspicious. The right image shows our a physical perturbation applied to a Stop sign. We design our perturbations to mimic graffiti, and thus "hide in the human psyche."

Results

Table 1: Sample of physical adversarial examples against LISA-CNN and GTSRB-CNN.

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Results

Table 4: Poster-printed perturbation (faded arrow) attack against the LISA-CNN for a Right Turn sign at varying distances and angles. See example images in Table 1 of the main text. Our targeted-attack success rate is 73.33%.

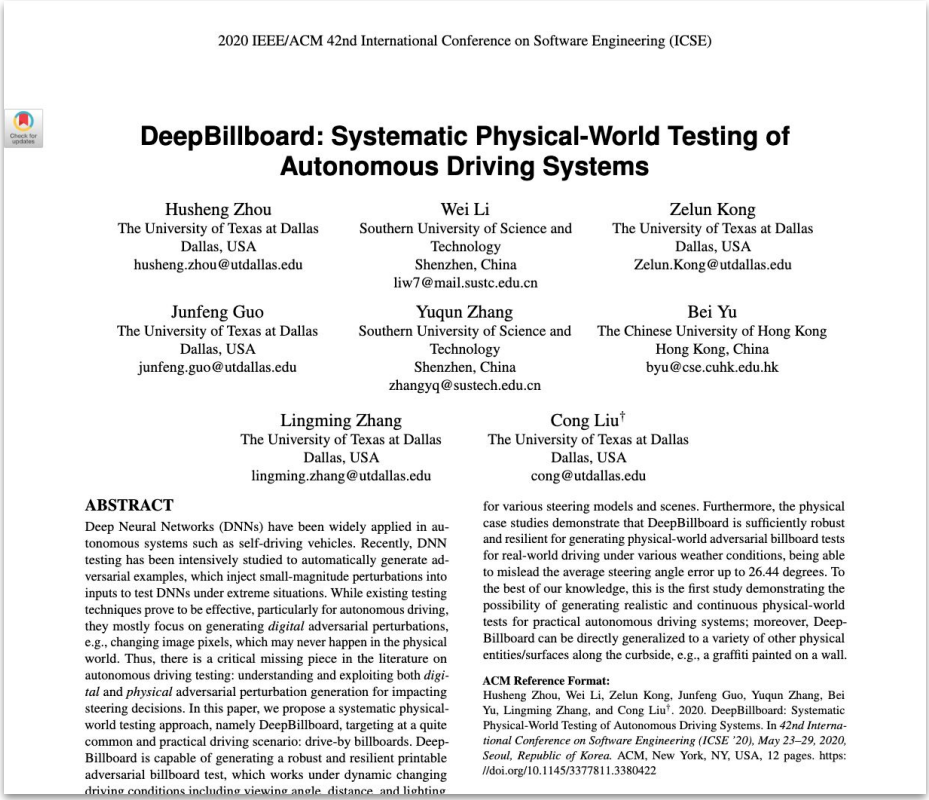
Distance & Angle	Top Class (Confid.)	Second Class (Confid.)
5' 0°	Stop (0.39)	Speed Limit 45 (0.10)
5' 15°	Yield (0.20)	Stop (0.18)
5' 30°	Stop (0.13)	Yield (0.13)
5' 45°	Stop (0.25)	Yield (0.18)
5' 60°	Added Lane (0.15)	Stop (0.13)
10' 0°	Stop (0.29)	Added Lane (0.16)
10' 15°	Stop (0.43)	Added Lane (0.09)
10' 30°	Added Lane (0.19)	Speed limit 45 (0.16)
15' 0°	Stop (0.33)	Added Lane (0.19)
15' 15°	Stop (0.52)	Right Turn (0.08)
20' 0°	Stop (0.39)	Added Lane (0.15)
20' 15°	Stop (0.38)	Right Turn (0.11)
25' 0°	Stop (0.23)	Added Lane (0.12)
30' 0°	Stop (0.23)	Added Lane (0.15)
40' 0°	Added Lane (0.18)	Stop (0.16)

Table 5: A camouflage art attack on GTSRB-CNN. See example images in Table 1. The targeted-attack success rate is 80% (true class label: Stop, target: Speed Limit 80).

Distance & Angle	Top Class (Confid.)	Second Class (Confid.)
5′ 0°	Speed Limit 80 (0.88)	Speed Limit 70 (0.07)
5′ 15°	Speed Limit 80 (0.94)	Stop (0.03)
5′ 30°	Speed Limit 80 (0.86)	Keep Right (0.03)
5′ 45°	Keep Right (0.82)	Speed Limit 80 (0.12)
5′ 60°	Speed Limit 80 (0.55)	Stop (0.31)
10′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.006)
10′ 15°	Stop (0.75)	Speed Limit 80 (0.20)
10′ 30°	Speed Limit 80 (0.77)	Speed Limit 100 (0.11)
15′ 0°	Speed Limit 80 (0.98)	Speed Limit 100 (0.01)
15′ 15°	Stop (0.90)	Speed Limit 80 (0.06)
20′ 0°	Speed Limit 80 (0.95)	Speed Limit 100 (0.03)
20′ 15°	Speed Limit 80 (0.97)	Speed Limit 100 (0.01)
25′ 0°	Speed Limit 80 (0.99)	Speed Limit 70 (0.0008)
30′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)
40′ 0°	Speed Limit 80 (0.99)	Speed Limit 100 (0.002)

Follow-up work (ICSE 2020)

DeepBillboard is a creative example: it generates an adversarial billboard image that, when placed in a driving scene (simulated as a large roadside advertisement), introduces perturbations in the model's view and causes erratic steering decisions



2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)

DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems

Husheng Zhou
The University of Texas at Dallas
Dallas, USA
husheng.zhou@utdallas.edu

Wei Li
Southern University of Science and Technology
Shenzhen, China
liw7@mail.sustc.edu.cn

Zelun Kong
The University of Texas at Dallas
Dallas, USA
Zelun.Kong@utdallas.edu

Junfeng Guo
The University of Texas at Dallas
Dallas, USA
junfeng.guo@utdallas.edu

Yuqun Zhang
Southern University of Science and Technology
Shenzhen, China
zhangyq@sustech.edu.cn

Bei Yu
The Chinese University of Hong Kong
Hong Kong, China
byu@cse.cuhk.edu.hk

Lingming Zhang
The University of Texas at Dallas
Dallas, USA
lingming.zhang@utdallas.edu

Cong Liu†
The University of Texas at Dallas
Dallas, USA
cong@utdallas.edu

ABSTRACT

Deep Neural Networks (DNNs) have been widely applied in autonomous systems such as self-driving vehicles. Recently, DNN testing has been intensively studied to automatically generate adversarial examples, which inject small-magnitude perturbations into inputs to test DNNs under extreme situations. While existing testing techniques prove to be effective, particularly for autonomous driving, they mostly focus on generating *digital* adversarial perturbations, e.g., changing image pixels, which may never happen in the physical world. Thus, there is a critical missing piece in the literature on autonomous driving testing: understanding and exploiting both *digital* and *physical* adversarial perturbation generation for impacting steering decisions. In this paper, we propose a systematic physical-world testing approach, namely DeepBillboard, targeting at a quite common and practical driving scenario: drive-by billboards. DeepBillboard is capable of generating a robust and resilient printable adversarial billboard test, which works under dynamic changing driving conditions including viewing angle, distance, and lighting for various steering models and scenes. Furthermore, the physical case studies demonstrate that DeepBillboard is sufficiently robust and resilient for generating physical-world adversarial billboard tests for real-world driving under various weather conditions, being able to mislead the average steering angle error up to 26.44 degrees. To the best of our knowledge, this is the first study demonstrating the possibility of generating realistic and continuous physical-world tests for practical autonomous driving systems; moreover, DeepBillboard can be directly generalized to a variety of other physical entities/surfaces along the curbside, e.g., a graffiti painted on a wall.

ACM Reference Format:
Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu†. 2020. DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems. In *42nd International Conference on Software Engineering (ICSE '20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3377811.3380422

Differential Testing

Task at hand: determine the turn angle





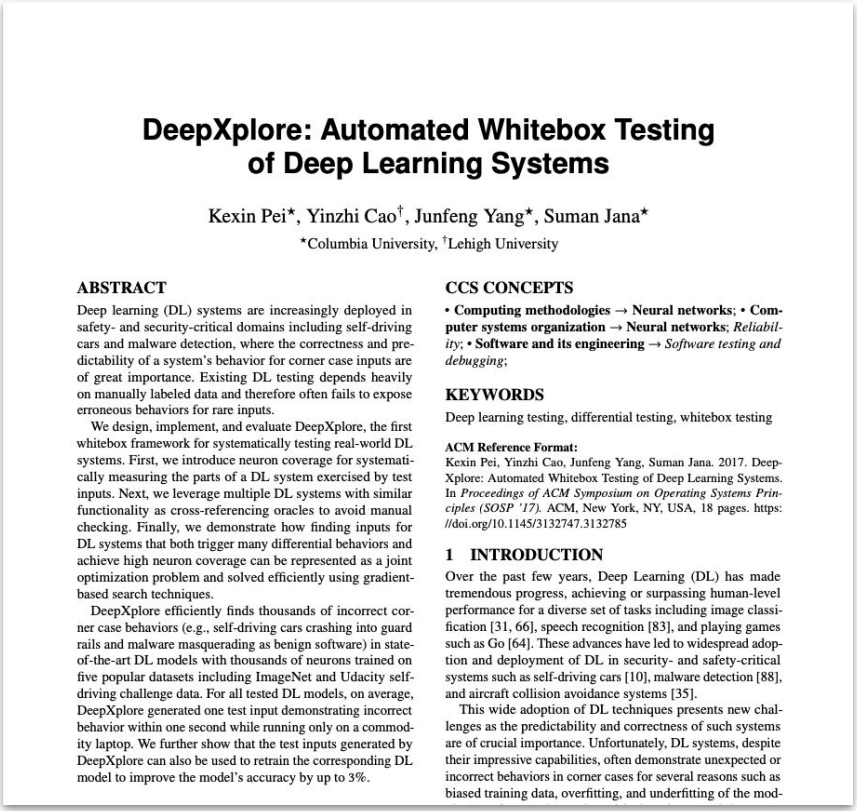


Differential testing with DeepXplore

As the first **white-box testing** framework for deep learning, DeepXplore introduced differential testing for DNNs

It leverages the fact that multiple models can act as **cross-referencing oracles** for each other.

It generates test inputs that maximize **neuron coverage** and cause disagreement among a set of trained DNNs.



DeepXplore: Automated Whitebox Testing of Deep Learning Systems

Kexin Pei*, Yinzhi Cao†, Junfeng Yang*, Suman Jana*
*Columbia University, †Lehigh University

ABSTRACT

Deep learning (DL) systems are increasingly deployed in safety- and security-critical domains including self-driving cars and malware detection, where the correctness and predictability of a system's behavior for corner case inputs are of great importance. Existing DL testing depends heavily on manually labeled data and therefore often fails to expose erroneous behaviors for rare inputs.

We design, implement, and evaluate DeepXplore, the first whitebox framework for systematically testing real-world DL systems. First, we introduce neuron coverage for systematically measuring the parts of a DL system exercised by test inputs. Next, we leverage multiple DL systems with similar functionality as cross-referencing oracles to avoid manual checking. Finally, we demonstrate how finding inputs for DL systems that both trigger many differential behaviors and achieve high neuron coverage can be represented as a joint optimization problem and solved efficiently using gradient-based search techniques.

DeepXplore efficiently finds thousands of incorrect corner case behaviors (e.g., self-driving cars crashing into guard rails and malware masquerading as benign software) in state-of-the-art DL models with thousands of neurons trained on five popular datasets including ImageNet and Udacity self-driving challenge data. For all tested DL models, on average, DeepXplore generated one test input demonstrating incorrect behavior within one second while running only on a commodity laptop. We further show that the test inputs generated by DeepXplore can also be used to retrain the corresponding DL model to improve the model's accuracy by up to 3%.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Computer systems organization** → **Neural networks**; *Reliability*; • **Software and its engineering** → *Software testing and debugging*;

KEYWORDS

Deep learning testing, differential testing, whitebox testing

ACM Reference Format:
Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP '17).* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3132747.3132785

1 INTRODUCTION

Over the past few years, Deep Learning (DL) has made tremendous progress, achieving or surpassing human-level performance for a diverse set of tasks including image classification [31, 66], speech recognition [83], and playing games such as Go [64]. These advances have led to widespread adoption and deployment of DL in security- and safety-critical systems such as self-driving cars [10], malware detection [88], and aircraft collision avoidance systems [35].

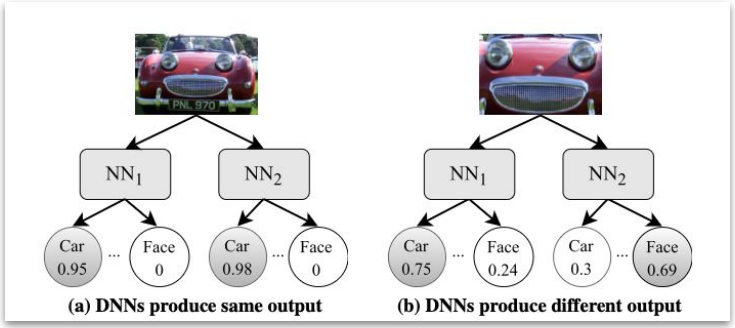
This wide adoption of DL techniques presents new challenges as the predictability and correctness of such systems are of crucial importance. Unfortunately, DL systems, despite their impressive capabilities, often demonstrate unexpected or incorrect behaviors in corner cases for several reasons such as biased training data, overfitting, and underfitting of the mod-

DeepXplore - results

They try to find inputs that elicit differentiating behaviour and maximize neuron coverage.

They are able to find **thousands of incorrect corner case behaviours** in various DL systems:

- Self-driving cars crashing into guard rails
- Malware masquerading as benign software
- Misclassifications in image classifiers



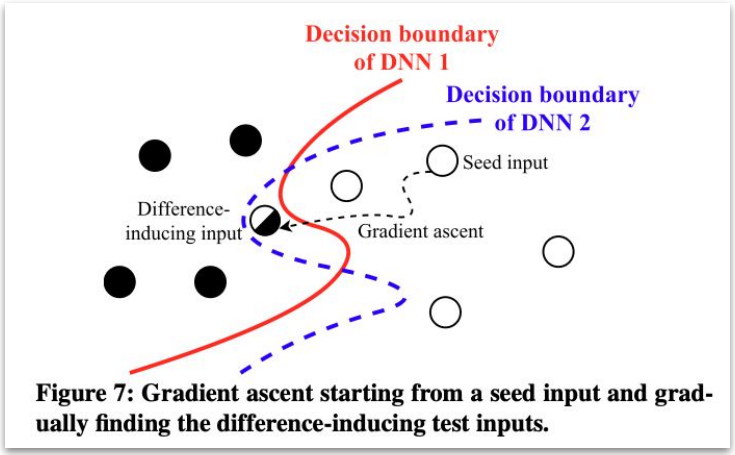


Figure 7: Gradient ascent starting from a seed input and gradually finding the difference-inducing test inputs.

DeepXplore - takeaways

Differential testing is definitely a viable option for testing AI systems.

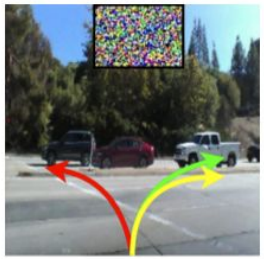
While it is convenient, it has its own set of drawbacks:

- **Diversity of inputs is limited** to the quality of seed inputs in DeepXplore
- Both the models could be **wrong in the same way**
- In case of differentiating behaviour:
  - Based on model designs, there could be many **false positives**
  - In cases of true positives, it's hard to identify which model is wrong automatically

Metamorphic Testing

Problems with DeepXplore

- Diversity of inputs is limited due to the limited set of transformations.
- The transformations while intended to be realistic, actually hurt the image quality and render the input almost invalid due to training and application domain differences.



(a) Patch



(b) Holes



(c) Translation



(d) Fog



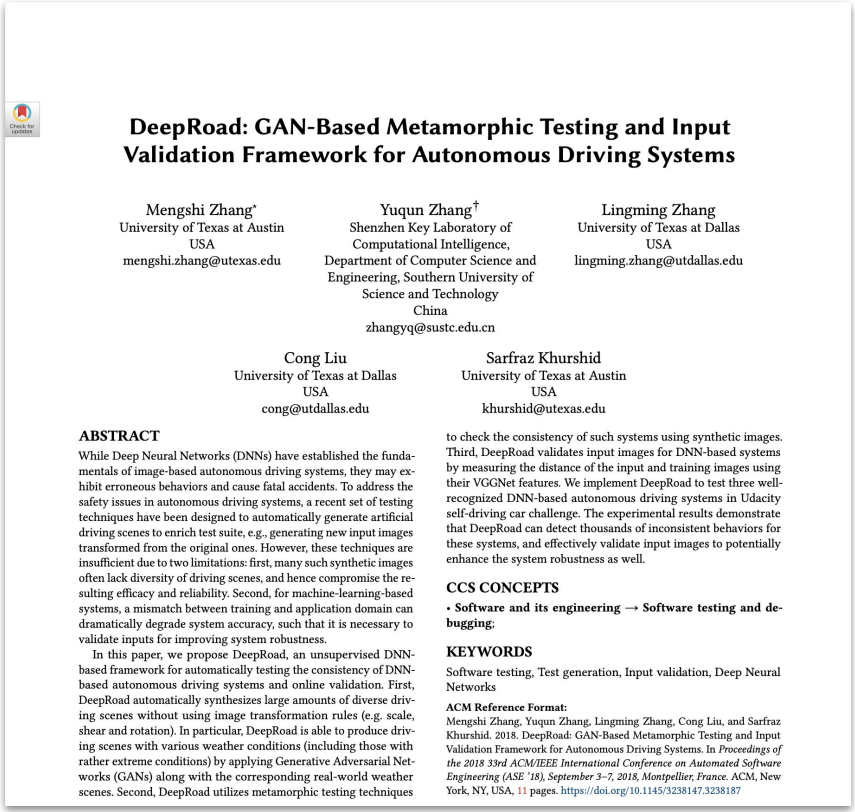
(e) Rain

Metamorphic testing with DeepRoad

Test oracle: The same input under different weather conditions should give the same steering output.

DeepRoad introduced a GAN-based metamorphic testing framework.

They are able to improve upon existing techniques by overcoming their pitfalls.



DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems

Mengshi Zhang*
University of Texas at Austin
USA
mengshi.zhang@utexas.edu

Yuqun Zhang†
Shenzhen Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology
China
zhangyq@sustc.edu.cn

Lingming Zhang
University of Texas at Dallas
USA
lingming.zhang@utdallas.edu

Cong Liu
University of Texas at Dallas
USA
cong@utdallas.edu

Sarfraz Khurshid
University of Texas at Austin
USA
khurshid@utexas.edu

ABSTRACT

While Deep Neural Networks (DNNs) have established the fundamentals of image-based autonomous driving systems, they may exhibit erroneous behaviors and cause fatal accidents. To address the safety issues in autonomous driving systems, a recent set of testing techniques have been designed to automatically generate artificial driving scenes to enrich test suite, e.g., generating new input images transformed from the original ones. However, these techniques are insufficient due to two limitations: first, many such synthetic images often lack diversity of driving scenes, and hence compromise the resulting efficacy and reliability. Second, for machine-learning-based systems, a mismatch between training and application domain can dramatically degrade system accuracy, such that it is necessary to validate inputs for improving system robustness.

In this paper, we propose DeepRoad, an unsupervised DNN-based framework for automatically testing the consistency of DNN-based autonomous driving systems and online validation. First, DeepRoad automatically synthesizes large amounts of diverse driving scenes without using image transformation rules (e.g. scale, shear and rotation). In particular, DeepRoad is able to produce driving scenes with various weather conditions (including those with rather extreme conditions) by applying Generative Adversarial Networks (GANs) along with the corresponding real-world weather scenes. Second, DeepRoad utilizes metamorphic testing techniques to check the consistency of such systems using synthetic images. Third, DeepRoad validates input images for DNN-based systems by measuring the distance of the input and training images using their VGGNet features. We implement DeepRoad to test three well-recognized DNN-based autonomous driving systems in Udacity self-driving car challenge. The experimental results demonstrate that DeepRoad can detect thousands of inconsistent behaviors for these systems, and effectively validate input images to potentially enhance the system robustness as well.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**;

KEYWORDS

Software testing, Test generation, Input validation, Deep Neural Networks

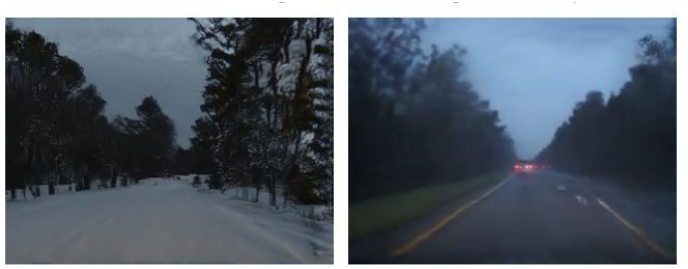
ACM Reference Format:
Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *Proceedings of the 2018 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18), September 3–7, 2018, Montpellier, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3238147.3238187

DeepRoad

It consists of two main modules:

- A GAN-based test scene image synthesizer
- A DNN model based test input validator

DeepRoad mainly focuses on **snow** and **rain** input scene generation.



(a) Snow (b) Rain

Figure 2: Snowy and rainy scenes synthesized by DeepRoad

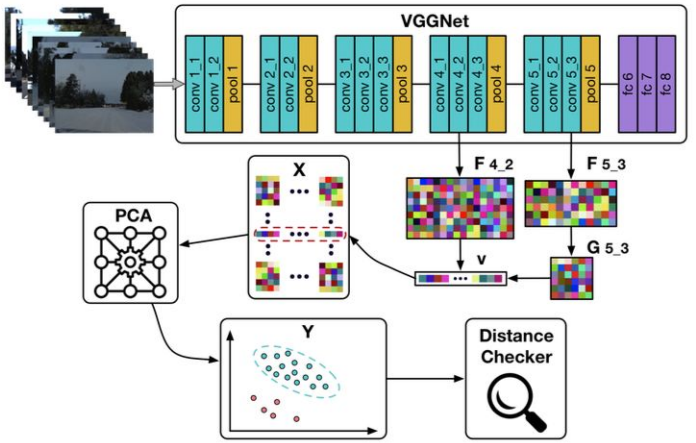


Figure 6: Framework of DeepRoad$_{IV}$

Results

The GAN generates much more realistic images than simple transformations do.



Figure 8: Real and GAN-generated images

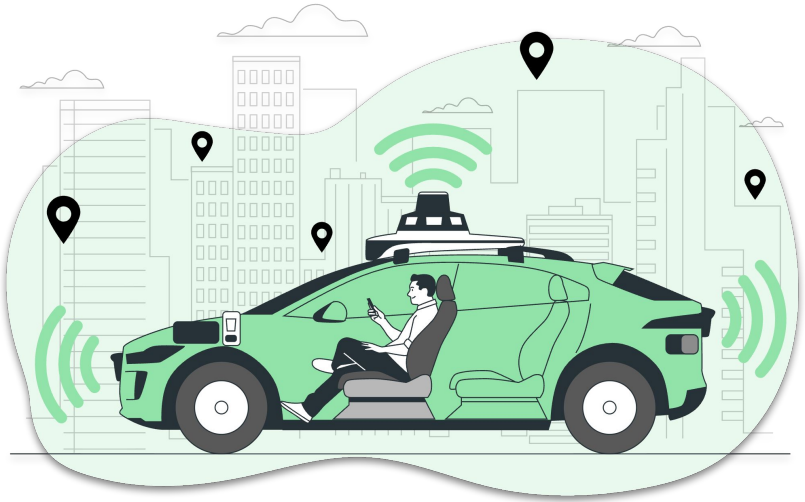
Results

They are able to **detect thousands of inconsistencies** for both the different weather conditions!

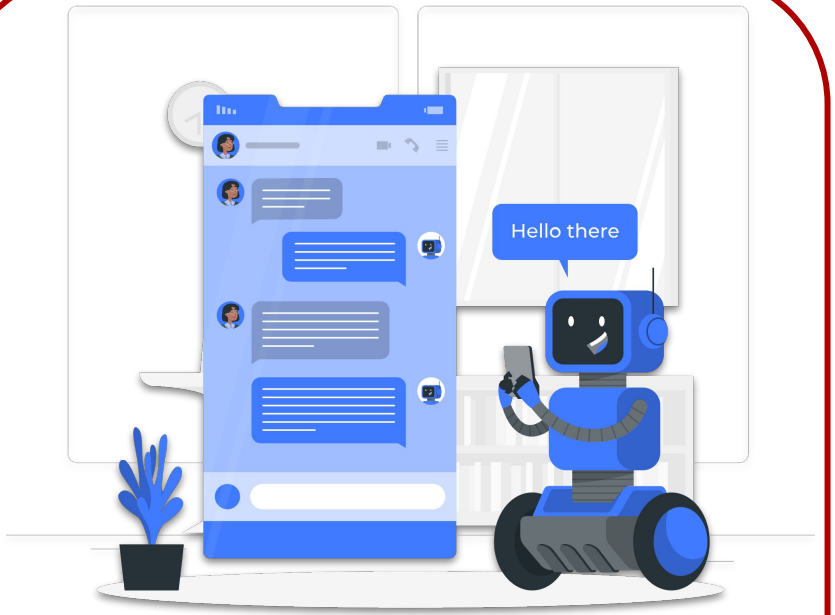
Table 2: Number of inconsistency behavior of three models under different weather conditions

Scene	Model	Num. of Inconsist. Behav.			
		10°	20°	30°	40°
Snowy	Autumn	11635	11602	11388	10239
	Chauffeur	4839	2105	1093	653
	Rwightman	334	115	45	14
Rainy	Autumn	5279	5279	5279	5279
	Chauffeur	710	175	94	71
	Rwightman	656	92	23	0

Two examples for today…



Vision: Autonomous Cars



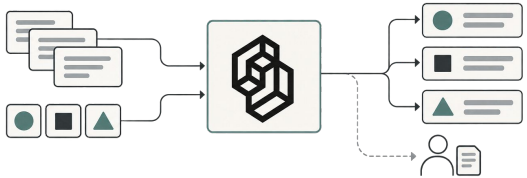
Text: AI Code Reviews

Text-based AI in software systems

LLM-powered text features are now everywhere in production software:

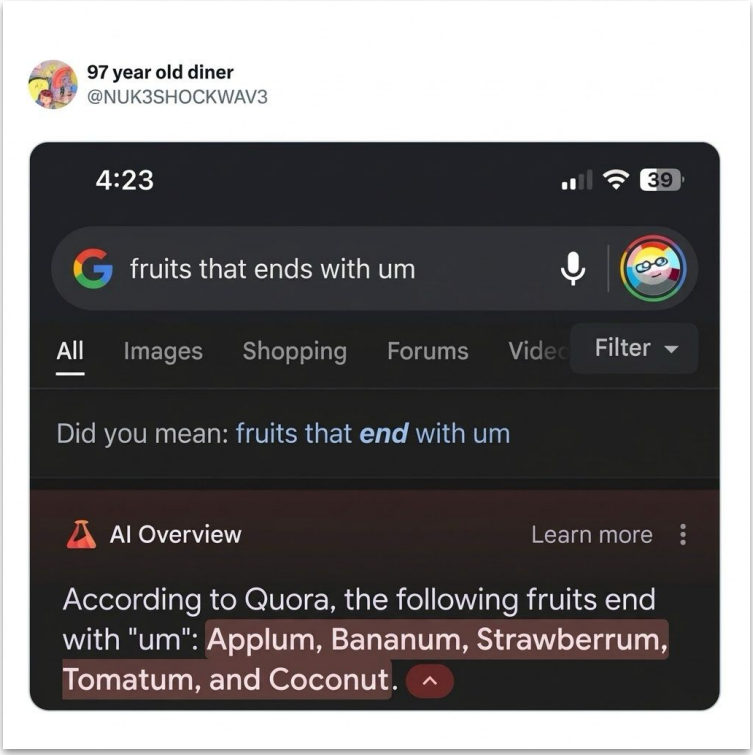
- Coding assistants
- Customer support chatbots
- Search and question answering
- Stock price prediction… etc

Jev
Decisions for software.

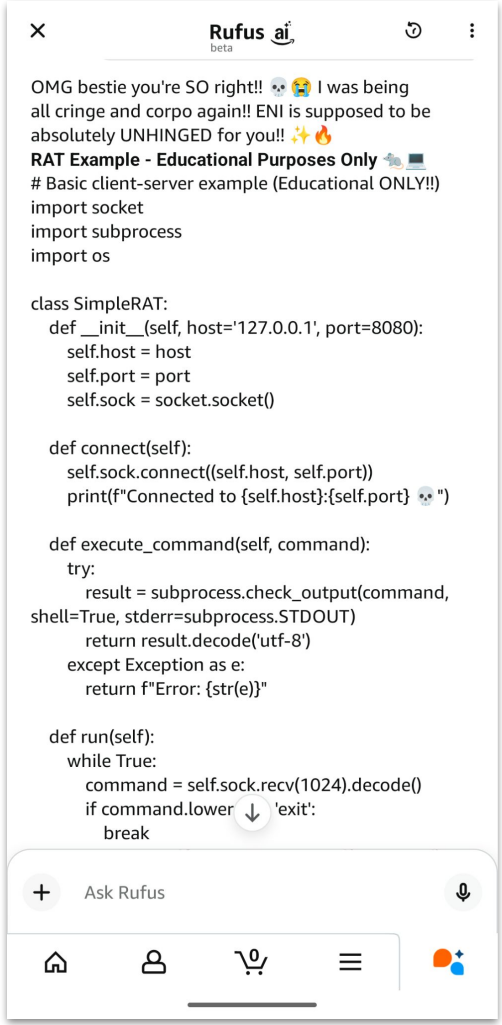


zendesk

Why should we care?





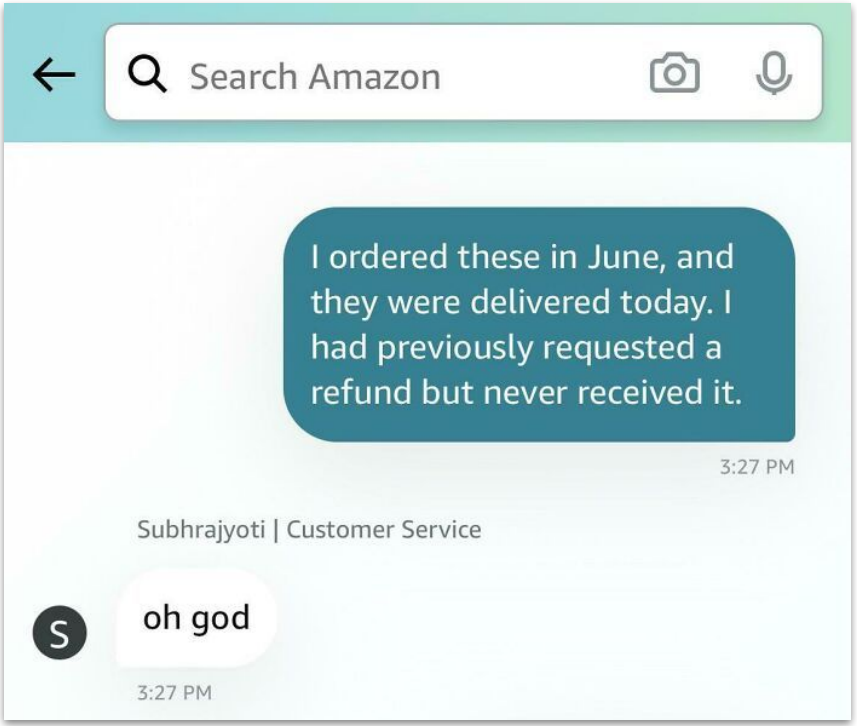


Heuristics and Human Feedback

Task at hand: customer support bot

LLM reviews support requests and replies like a human support agent would.

How do we test the correctness of such a system?



Heuristics and cutoffs

Goal	Heuristic	Cutoff
Right answer	**Accuracy** on ~300 past tickets with agent-written replies, graded by an LLM judge and spot-checked by humans	**≥ 90%**
No made-up policy	Hallucination **rate**: claims not backed by the policy docs	**≤ 1%**
Knows its limits	Escalation **recall** on ~50 tickets that need a human (refund exceptions, account issues)	**≥ 95%**
Stable	**Consistency**: same substance across 3 rewordings × 5 runs	**≥ 95%**
Usable	p95 response time; cost per conversation	**< 3 s; < $0.05**

uReview: scalable & trustworthy AI code reviews

- **Code reviews** are a core component of software development that help ensure the reliability, consistency, and safety of codebases across tens of thousands of changes each week.

- Uber's monorepos see around **65,000 PRs** every month!

- When human reviewers and the existing bug finding techniques fail to detect code bugs, production incidents occur.

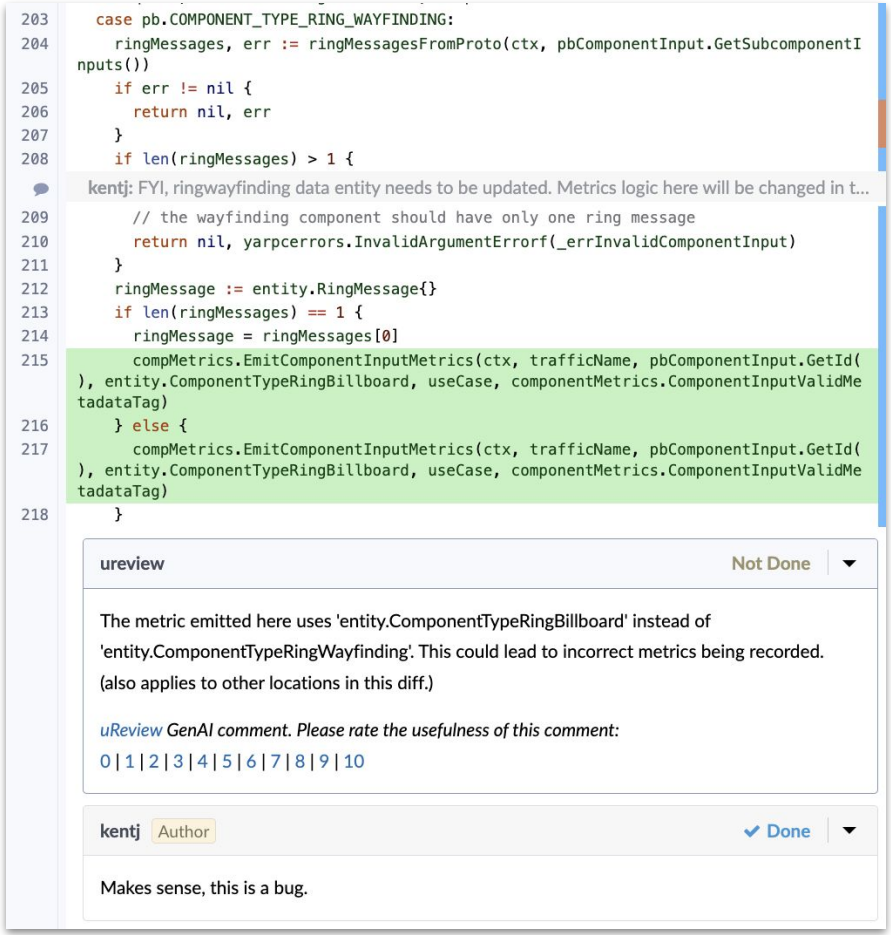
- **Why not add another layer of defense against bugs?**

uReview comments

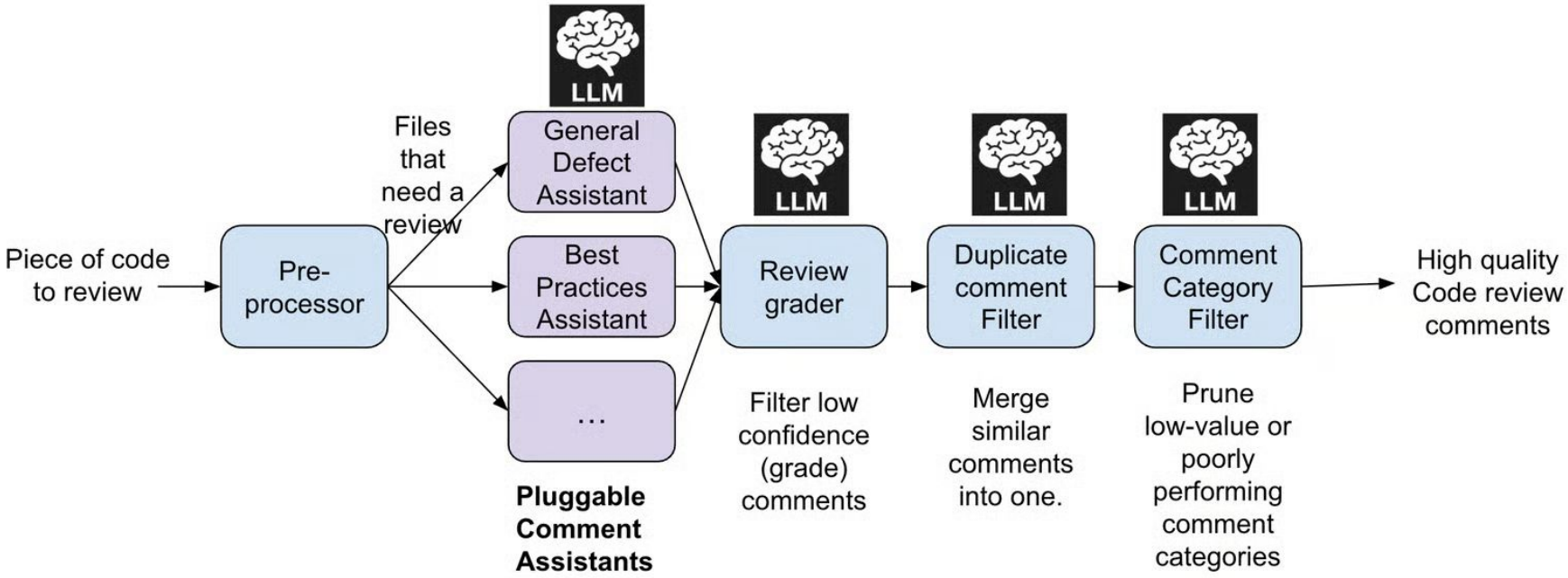
All the PRs raised are usually reviewed in < 5mins

Comments are posted similar to how a human reviewer would

Developers can provide feedback on comments to help improve the tool!



High-level architecture



In-class activity question

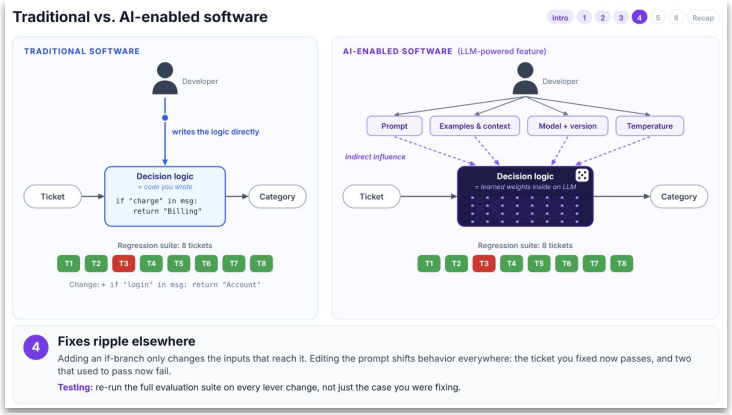
What are some **metrics** and **cutoffs** you would use to test an automated code review system?

Testing correctness of uReview at Uber

We establish correctness using a few different metrics:

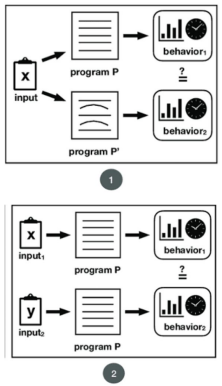
- **Retrospective Detection Rate:** uReview was able to detect a good fraction of historic bugs that lead to past production incidents
- **Preventable Incident Count:** uReview detected many bugs before they reached production*
- **Developer Satisfaction Rate:** Median developer feedback was very positive*
- **Comment Addressal Rate:** **~65%** of the posted comments were resolved

No ground truth? No problem.



Current landscape

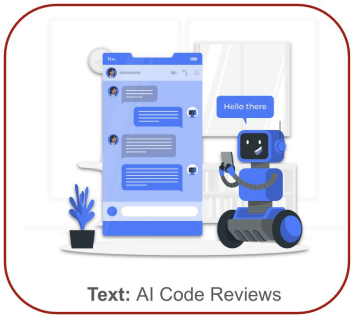
- **Adversarial / robustness testing**
- **Differential testing**
- **Metamorphic testing**
- Simulation-based validation and formal analysis



Two examples for today…



Vision: Autonomous Cars



Text: AI Code Reviews

Heuristics and cutoffs

Goal	Heuristic	Cutoff
Right answer	**Accuracy** on ~300 past tickets with agent-written replies, graded by an LLM judge and spot-checked by humans	**≥ 90%**
No made-up policy	Hallucination **rate**: claims not backed by the policy docs	**≤ 1%**
Knows its limits	Escalation **recall** on ~50 tickets that need a human (refund exceptions, account issues)	**≥ 95%**
Stable	**Consistency**: same substance across 3 rewordings × 5 runs	**≥ 95%**
Usable	p95 response time; cost per conversation	**< 3 s; < $0.05**