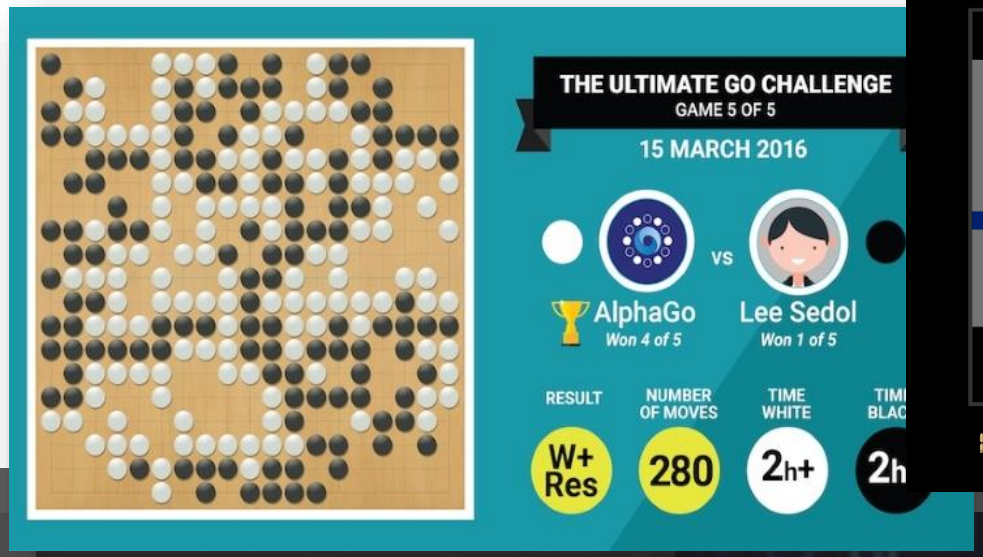
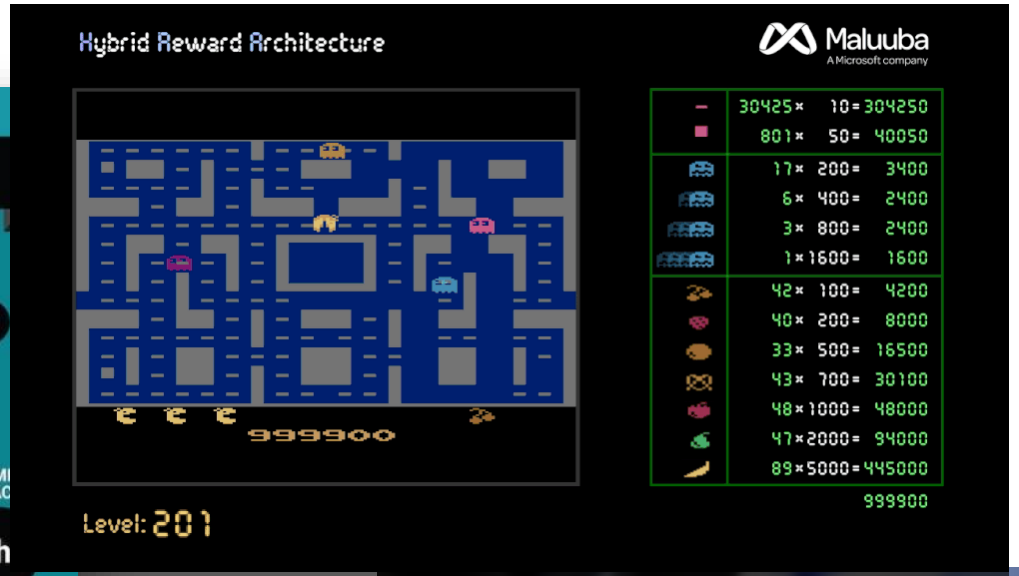
AI in the Open World: Discovering Blind Spots of AI

Ece Kamar
eckamar@microsoft.com

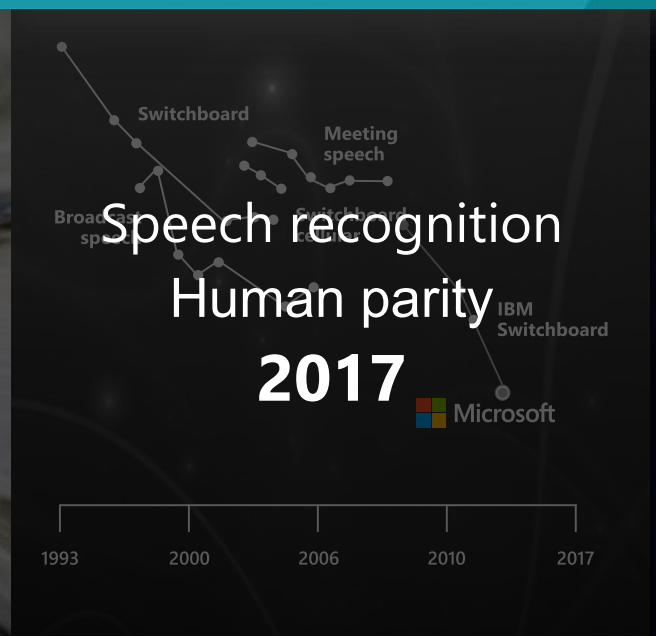
Microsoft Research AI

Recent AI Successes





Object recognition
Human parity
2016



Speech recognition
Human parity
2017

Machine reading comprehension
Human parity
Jan 2018

Machine translation
Human parity
March 2018

What Happens in the Real World

AI in the court: When algorithms rule on jail time

By MATT O'BRIEN AND DAKE KANG, ASSOCIATED PRESS

CLEVELAND — Jan 31, 2018, 5:04 PM ET

Share Tweet

A.I. Shows Promise Assisting Physicians

Doctors competed against A.I. computers to recognize illnesses on magnetic resonance images of a human brain during a competition in Beijing last year. The human doctors lost.

Mark Schiefelbein/Associated Press

Self-driving car timeline for 11 top automakers

DAN FAGELLA, TECHEMERGENCE @DANFAGGELLA JUNE 4, 2017 3:10 PM

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica
May 23, 2016

ON A SPRING AFTERNOON IN 2014, Brisha Borden was running late to pick up her god-sister from school when she spotted an unlocked kid's blue Huffy bicycle and a silver Razor scooter. Borden and a friend grabbed the bike and scooter and tried to ride them down the street in the Fort Lauderdale suburb of Coral Springs.

SCIENCE \ TECH \ HEALTH

IBM's Watson gave unsafe recommendations for treating cancer

Doctors fed it hypothetical scenarios, not real patient data

By Angela Chen | @chengela | Jul 26, 2018, 4:29pm EDT

SHARE

Tesla's Autopilot keeps crashing into parked cars. Here's why.

Principles of Responsible AI

Aether Committee

Microsoft

AI, Ethics and Effects
in Engineering & Research

BIAS AND FAIRNESS

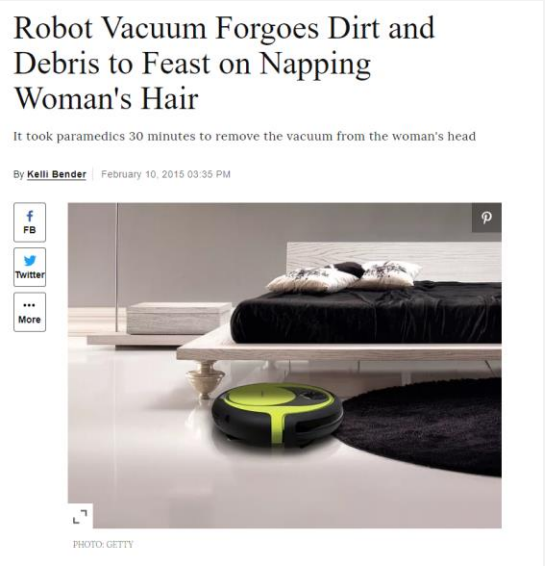
INTELLIGIBILITY & TRANSPARENCY

RELIABILITY AND SAFETY

HUMAN-AI INTERACTION & COLLABORATION

Insight #1:
Responsible AI emerges from understanding **AI limitations** in the **open-world**.

Many Reasons for Failures



Robot Vacuum Forgoes Dirt and Debris to Feast on Napping Woman's Hair

It took paramedics 30 minutes to remove the vacuum from the woman's head

By Kelli Bender | February 10, 2015 03:35 PM

FB
Twitter
More

PHOTO: GETTY

Incomplete/incorrect models

Misspecified goals and objectives

Attacks against AI systems

Eykholt, Evtimov, et al. 2018

Human error

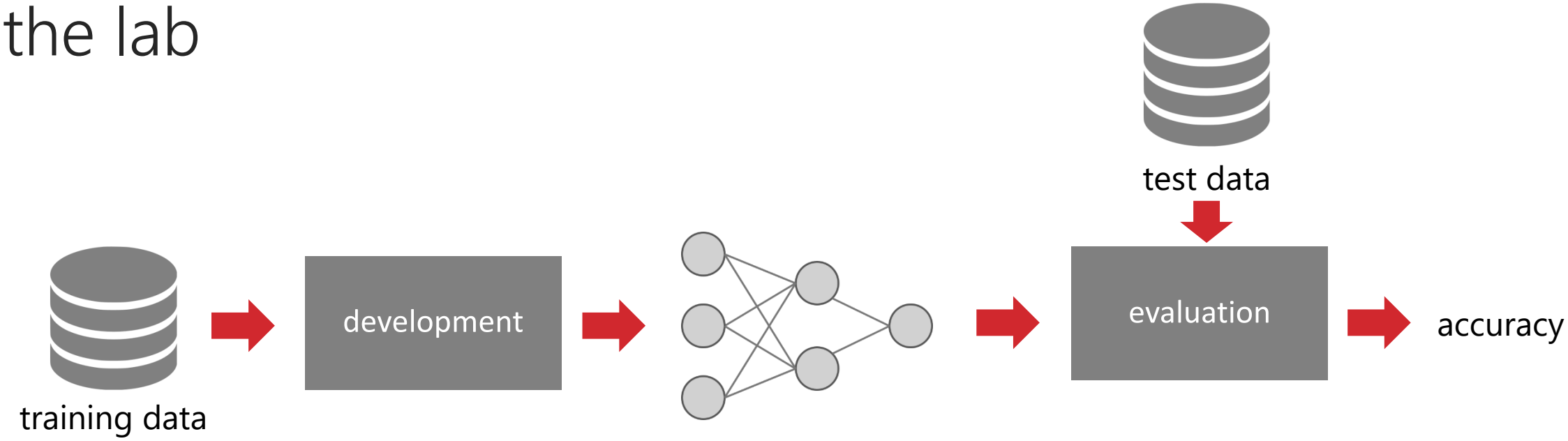
Tesla driver killed while using autopilot was watching Harry Potter, witness says

Driver in first known fatal self-driving car crash was also driving so fast that 'he went so fast through my trailer I didn't see him', the truck driver involved said

0:00 / 1:02

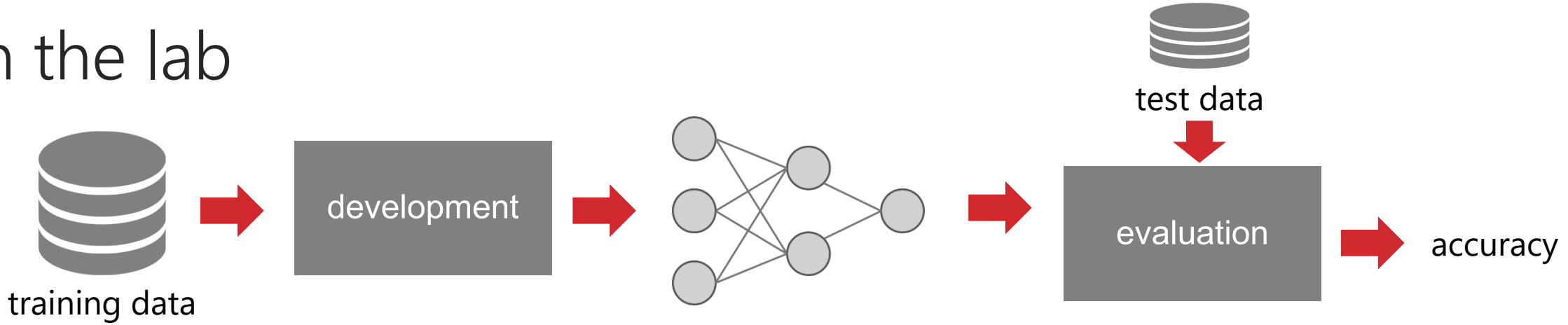
Insight #2:
There is no silver bullet, but variety of emerging techniques

In the lab

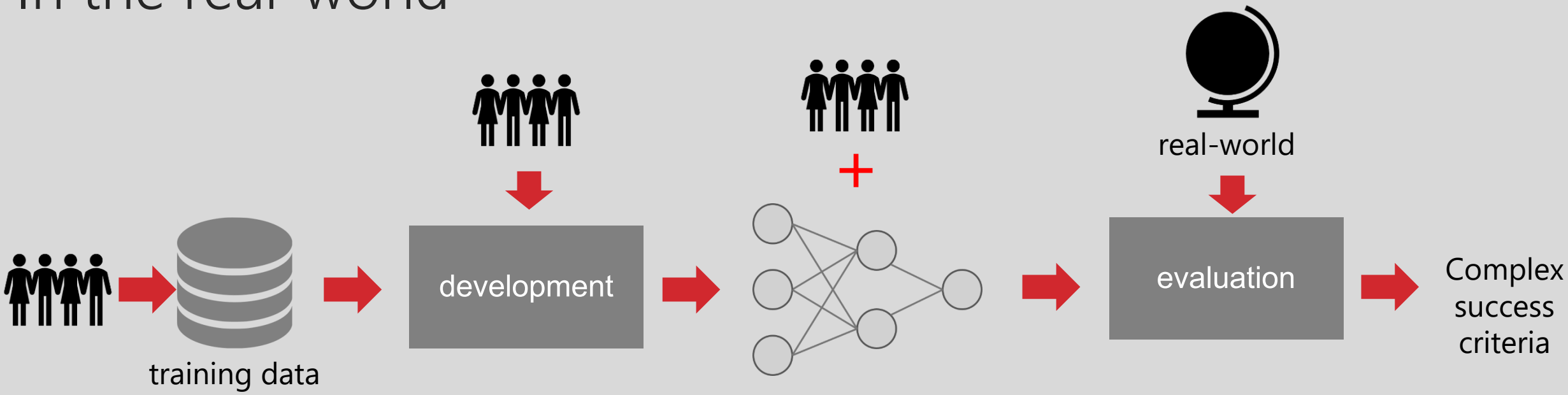


- Robust optimization and training
- Outlier detection approaches
- Incorporating causal learning
- Model expansion
- Risk-sensitive optimization …

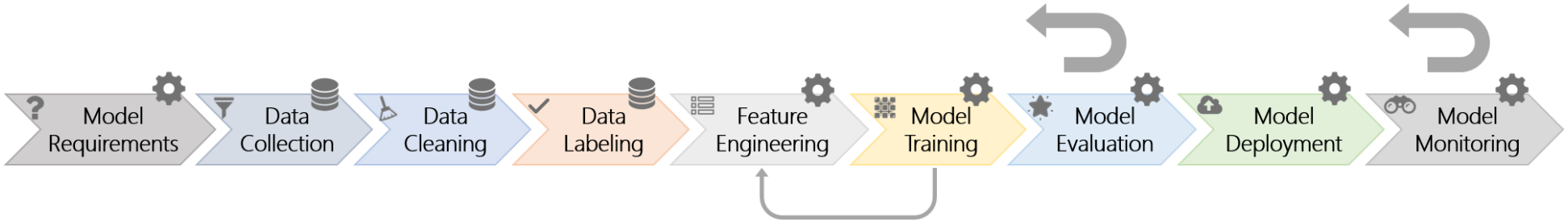
In the lab



In the real-world



Insight #3:
Aligning efforts with AI development cycle is key.



Software Engineering for Machine Learning: A Case Study

Saleema Amershi, *Microsoft Research*, Redmond, WA USA, samershi@microsoft.com
Andrew Begel, *Microsoft Research*, Redmond, WA USA, andrew.begel@microsoft.com
Christian Bird, *Microsoft Research*, Redmond, WA USA, cbird@microsoft.com
Robert DeLine, *Microsoft Research*, Redmond, WA USA, rdeline@microsoft.com
Harald Gall, *University of Zurich*, Zurich, Switzerland, gall@ifi.uzh.ch
Ece Kamar, *Microsoft Research*, Redmond, WA USA, eckamar@microsoft.com
Nachiappan Nagappan, *Microsoft Research*, Redmond, WA USA, nachin@microsoft.com
Besmira Nushi, *Microsoft Research*, Redmond, WA USA, besmira.nushi@microsoft.com
Thomas Zimmermann, *Microsoft Research*, Redmond, WA USA, tzimmer@microsoft.com

***Abstract*—Recent advances in machine learning have stimulated widespread interest within the Information Technology sector on integrating AI capabilities into software and services. This goal has forced organizations to evolve their development processes. We report on a study that we conducted on observing software teams at Microsoft as they develop AI-based applications. We consider a nine-stage workflow process informed by prior experiences developing AI applications (e.g., search and NLP) and data science tools (e.g. application diagnostics and bug reporting). We found that various Microsoft teams have united this workflow into preexisting, well-evolved, Agile-like software engineering processes, providing insights about several essential engineering challenges that organizations may face in creating large-scale AI solutions for the marketplace. We collected some best practices from Microsoft teams to address these challenges. In addition, we have identified three aspects of the AI domain that make it fundamentally different from prior software application domains: 1) discovering, managing, and versioning the data needed for machine learning applications is much more complex and difficult than other types of software engineering, 2) model customization and model reuse require very different skills than are typically found in software teams, and 3) AI components are more difficult to handle as distinct modules than traditional software components — models may be "entangled" in complex ways and experience non-monotonic error behavior. We believe that the lessons learned by Microsoft teams will be valuable to other organizations.**

***Index Terms*—AI, Software engineering, process, data**

I. Introduction

Personal computing. The Internet. The Web. Mobile computing. Cloud computing. Nary a decade goes by without a disruptive shift in the dominant application domain of the software industry. Each shift brings with it new software engineering goals that spur software organizations to evolve their development practices in order to address the novel aspects of the domain.

The latest trend to hit the software industry is around integrating artificial intelligence (AI) capabilities based on advances in machine learning. AI broadly includes technologies for reasoning, problem solving, planning, and learning, among others. Machine learning refers to statistical modeling techniques that have powered recent excitement in the software and services marketplace. Microsoft product teams have used machine learning to create application suites such as Bing Search or the Cortana virtual assistant, as well as platforms such as Microsoft Translator for real-time translation of text, voice, and video, Cognitive Services for vision, speech, and language understanding for building interactive, conversational agents, and the Azure AI platform to enable customers to build their own machine learning applications [1]. To create these software products, Microsoft has leveraged its preexisting capabilities in AI and developed new areas of expertise across the company.

In this paper, we describe a study in which we learned how various Microsoft software teams build software applications with customer-focused AI features. For that, Microsoft has integrated existing Agile software engineering processes with AI-specific workflows informed by prior experiences in developing early AI and data science applications. In our study, we asked Microsoft employees about how they worked through the growing challenges of daily software development specific to AI, as well as the larger, more essential issues inherent in the development of large-scale AI infrastructure and applications. With teams across the company having differing amounts of work experience in AI, we observed that many issues reported by newer teams dramatically drop in importance as the teams mature, while some remain as essential to the practice of large-scale AI. We have made a first attempt to create a process maturity metric to help teams identify how far they have come on their journeys to building AI applications.

As a key finding of our analyses, we discovered three fundamental differences to building applications and platforms for training and fielding machine-learning models than we have seen in prior application domains. First, machine learning is all about data. The amount of effort and rigor it takes to discover, source, manage, and version data is inherently more complex and different than doing the same with software code. Second, building for customizability and extensibility of models require teams to not only have software engineering skills but almost

A study with 565 ML engineers

End-to-end tool fragmentation

Tools make ML too difficult

Data collection and cleaning is hard

Education and lack of expertise

Debugging is hard

Model evaluation and deployment

Compliance and ethics

Outline

- Blind spot detection for supervised learning

  [with Himabindu Lakkaraju, Rich Caruana and Eric Horvitz, AAAI 2016]

- Blind spots of reinforcement learning

  [with Ramya Ramakrishnan, Debadeepta Dey, Julie Shah and Eric Horvitz, AAMAS 2018; AAAI 2019]

- Backward-compatibility for AI systems

  [with Gagan Bansal, Besmira Nushi, Dan Weld and Eric Horvitz, AAAI 2019]

Outline

- **Blind spot detection for supervised learning**

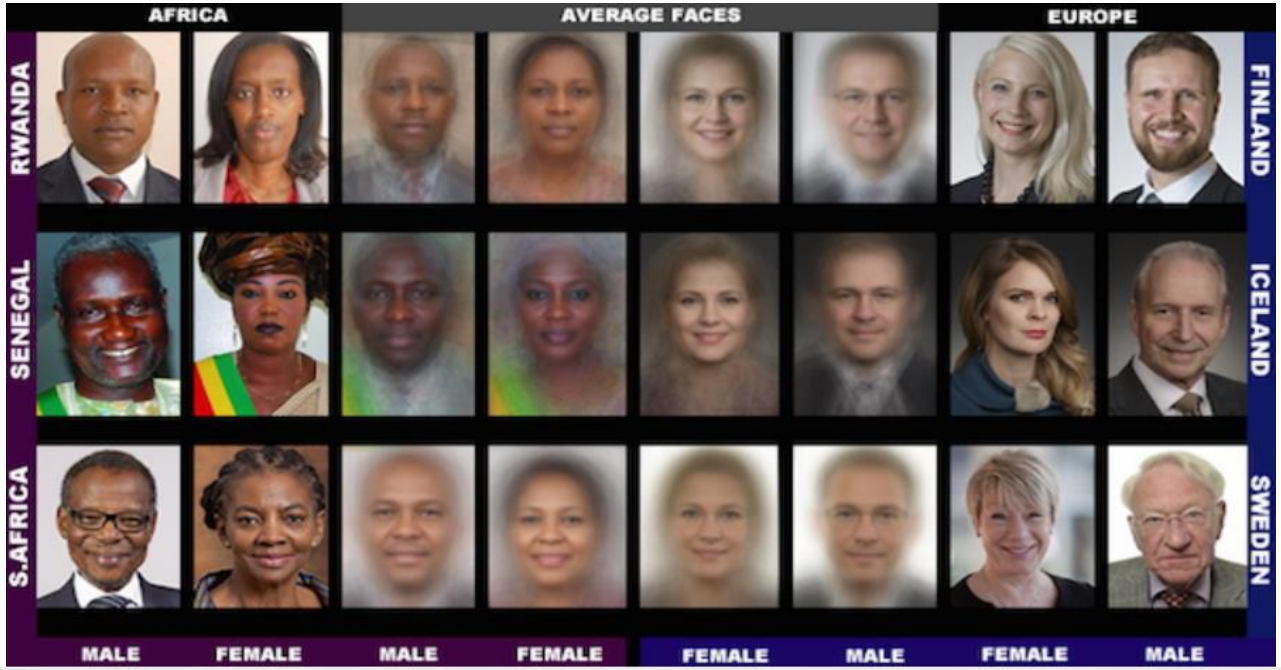
  [with Himabindu Lakkaraju, Rich Caruana and Eric Horvitz, AAAI 2016]

- Blind spots of reinforcement learning

  [with Ramya Ramakrishnan, Debadeepta Dey, Julie Shah and Eric Horvitz, AAMAS 2018; AAAI 2019]

- Backward-compatibility for AI systems

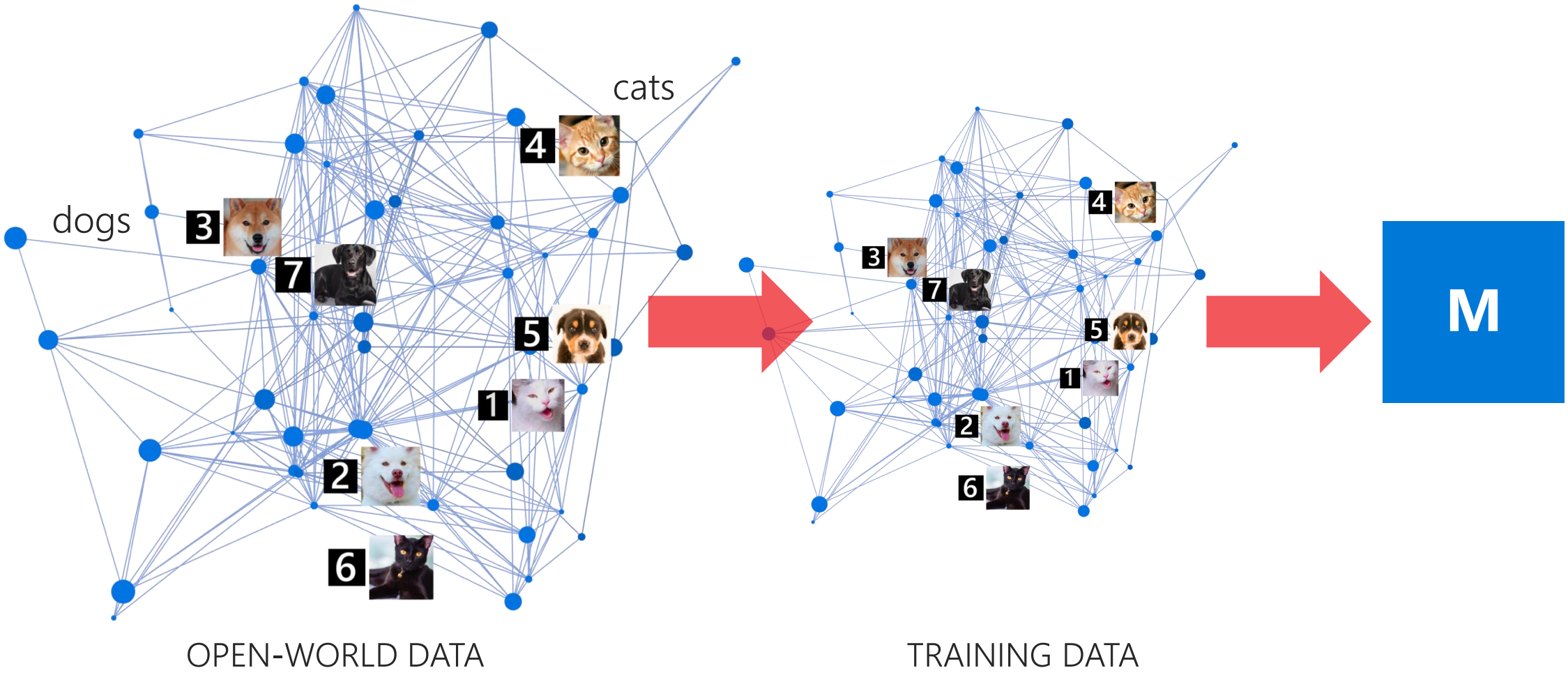
  [with Gagan Bansal, Besmira Nushi, Dan Weld and Eric Horvitz, AAAI 2019]



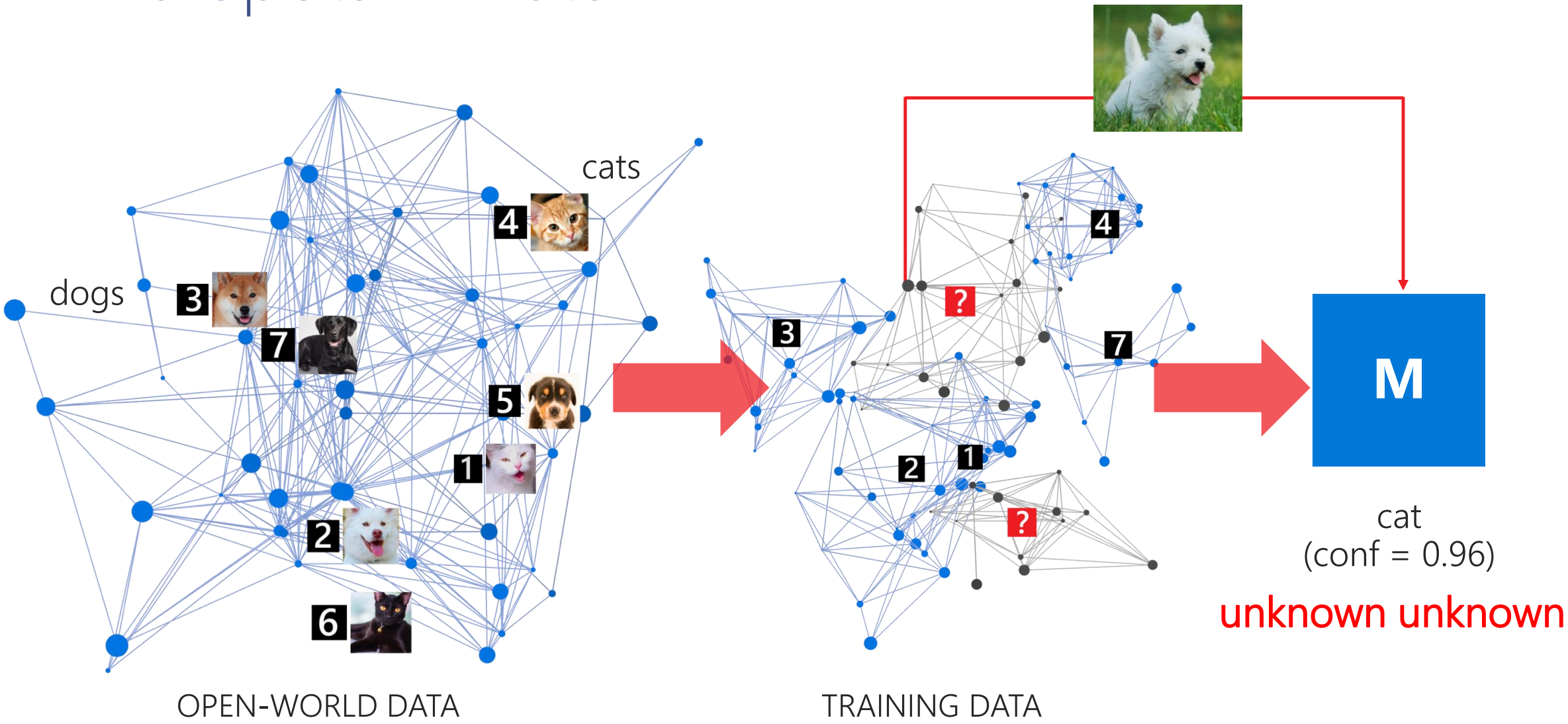
J. Buolamwini & T. Gebru

Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%

Fundamental Assumption in ML



Blind Spots in Data

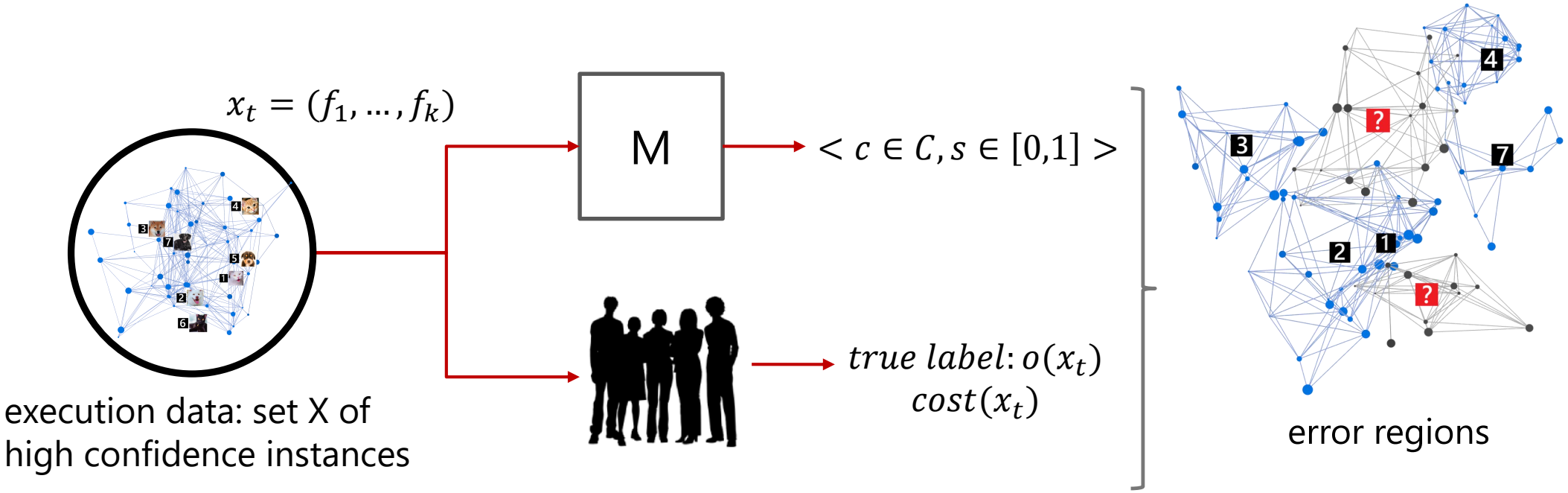


Focus: Blind-spot Detection

- **Unknown unknowns:** Data points with confident but incorrect predictions.
- **Blind-spots:** Feature spaces with high concentration of unknown unknowns
- **Cause:** Mismatch between training and execution data
- Common active learning techniques do not help!

with Himabundu Lakkaraju, Rich Caruana and Eric Horvitz, AAAI 2017

Problem Definition



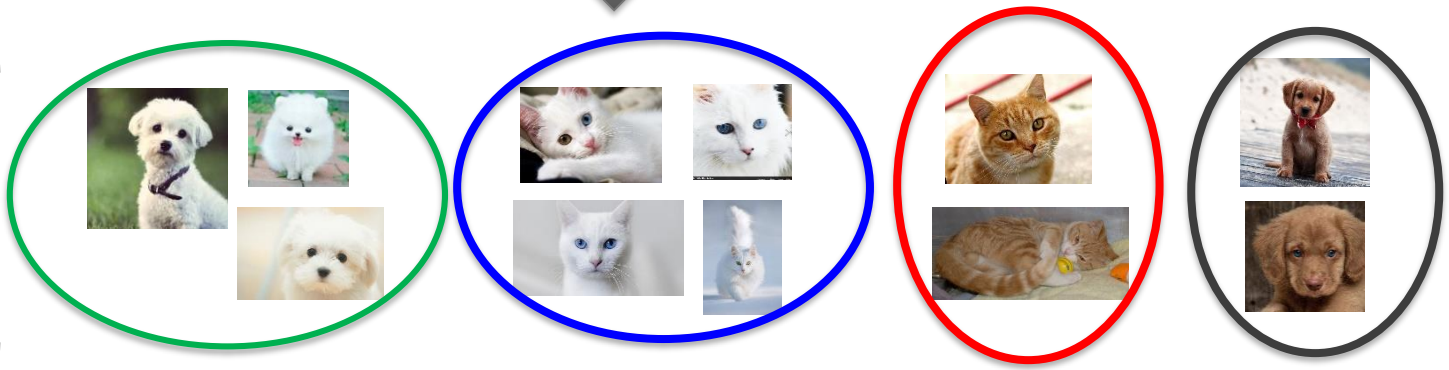
Two Main Challenges:

- Searching for errors
- Interpretability of error regions

Our Framework

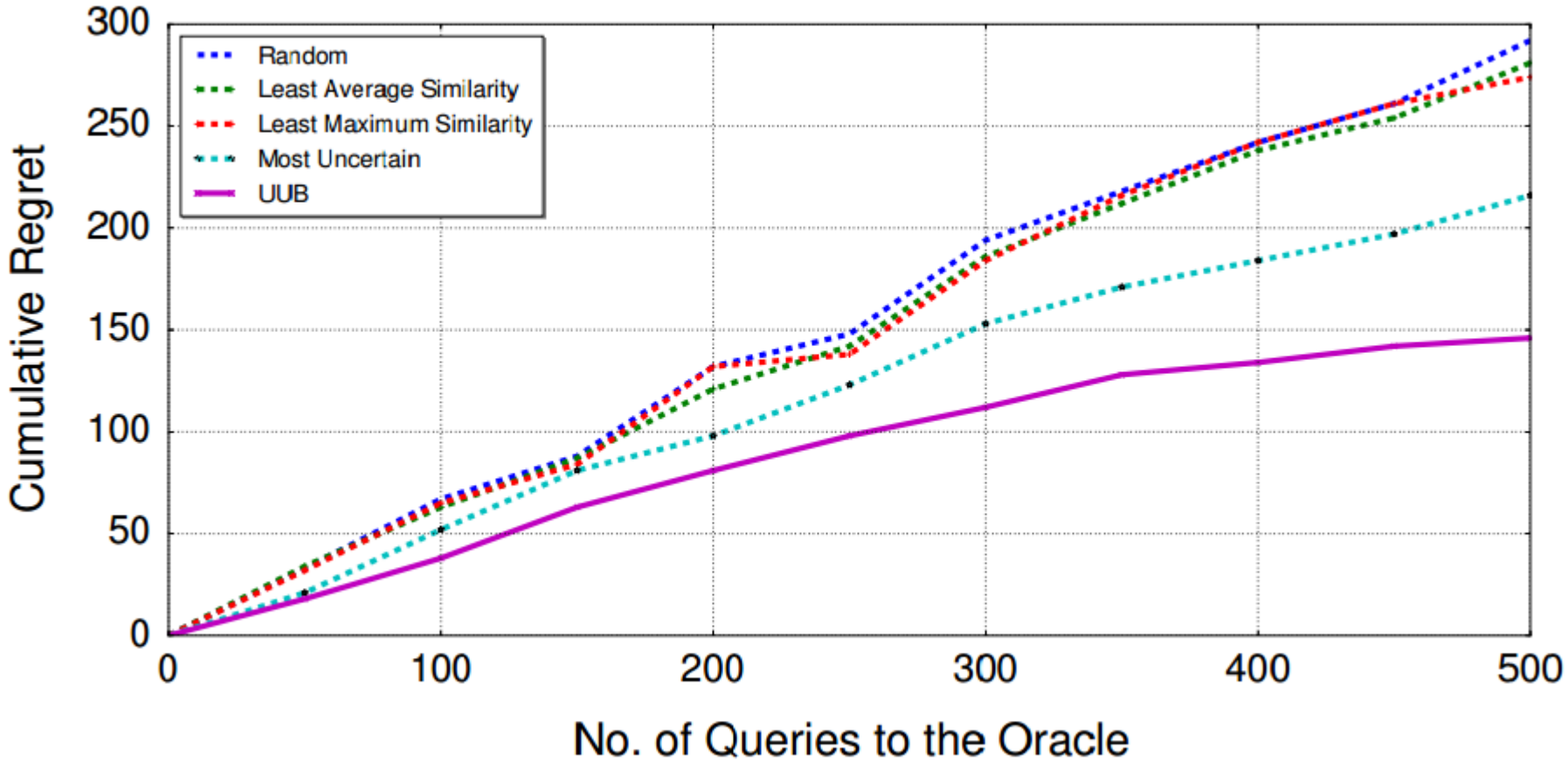
Input:
Execution data points with high confidence

Step 1:
Descriptive Space Partitioning



Step 2:
Multi-armed bandits for unknown unknowns

Comparison with Alternative Methods (more in the paper)



Outline

- Blind spot detection for supervised learning

  [with Himabindu Lakkaraju, Rich Caruana and Eric Horvitz, AAAI 2016]

- **Blind spots of reinforcement learning**

  [with Ramya Ramakrishnan, Debadeepta Dey, Julie Shah and Eric Horvitz, AAMAS 2018; AAAI 2019]

- Backward-compatibility for AI systems

  [with Gagan Bansal, Besmira Nushi, Dan Weld and Eric Horvitz, AAAI 2019]

Simulation


Reverse: (API)
Throttle: (API)
Steering: (API)
Footbrake: (API)
Speed: 42 km/h
Gear: 2
Collision#2 with Car_58 - ObjID:22

Real world

Simulation → Real world

≠

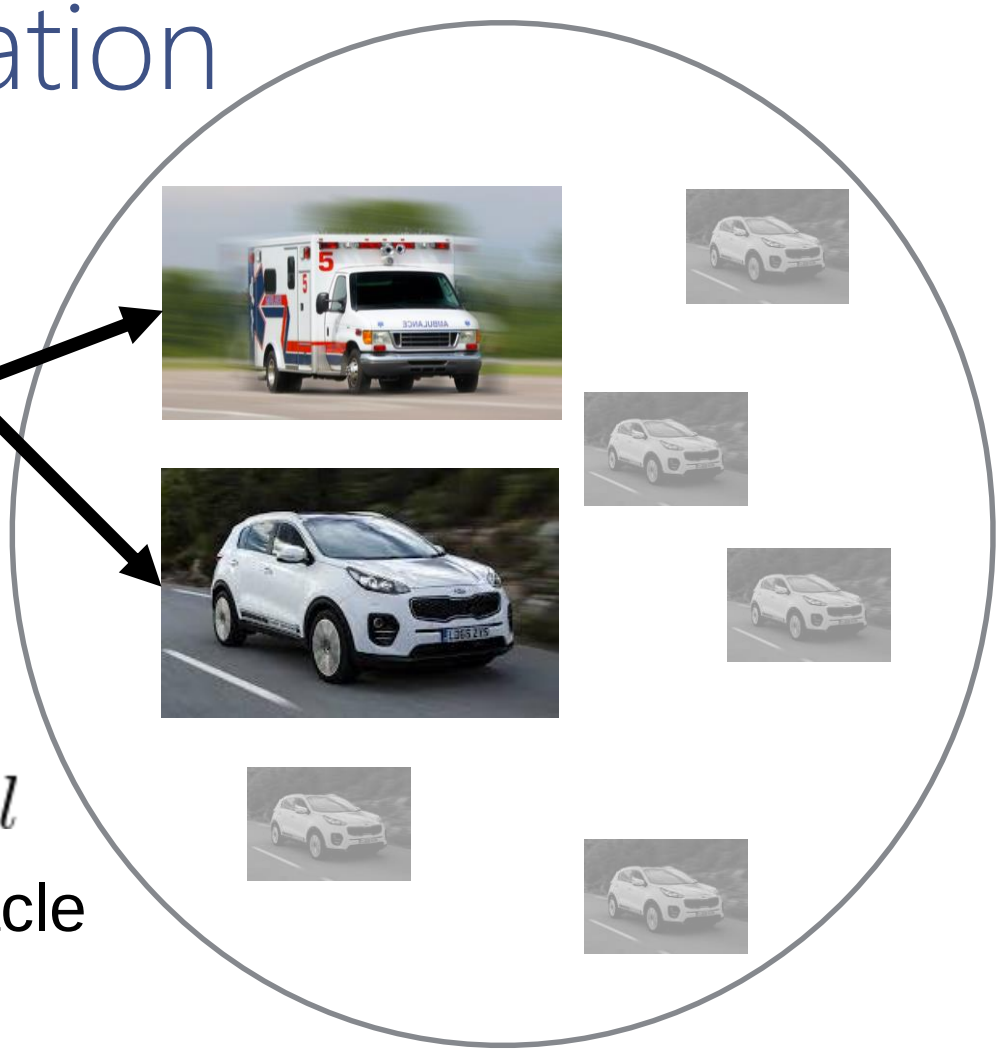
Blind spots in RL: Mismatch between simulation and real world can cause costly mistakes

Approach: Use **human (oracle) guidance** to transfer from sim to real in a safe way

Type of blind spots: Limited State Representation

S_{sim}

What agent sees



S_{real}

What oracle sees

Type of blind spots: Limited State Representation

Blind spot

S_{sim}

What agent sees

Agent action unacceptable

Agent action acceptable

S_{real}

What oracle sees

Learning blind spots

One Approach

Active learning and testing in the world

- Dangerous

Alternative Approach

Use oracle to safely obtain data in the real world

Demonstrations

+ Easy to obtain

- Only provides oracle's action, not input on agent's action

Corrections

- Harder to obtain

+ Gives direct input on agent's policy

Problem

π_{sim}

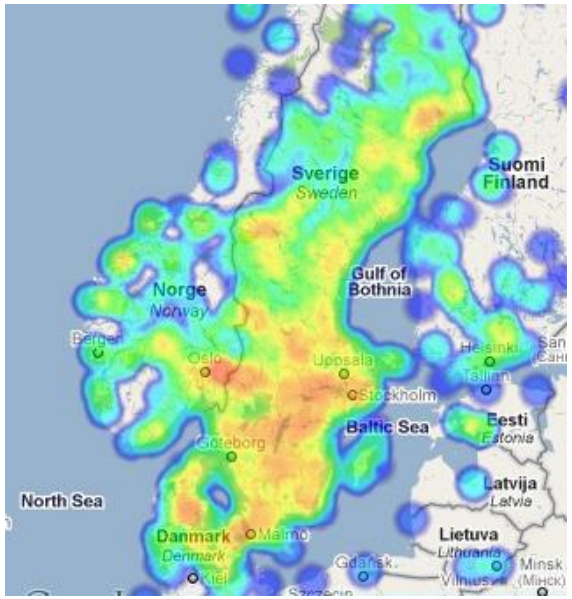
ORACLE O

$\{A(s,a), \pi_{real}\}$

B

Our approach

M = Blind spot model



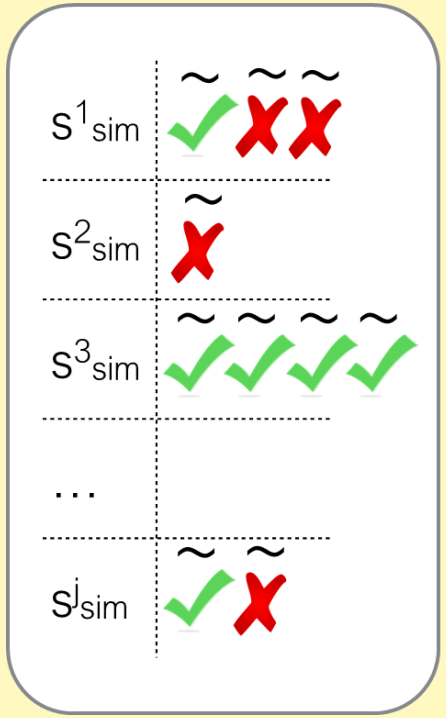
Our Approach

π_{sim}

O

B

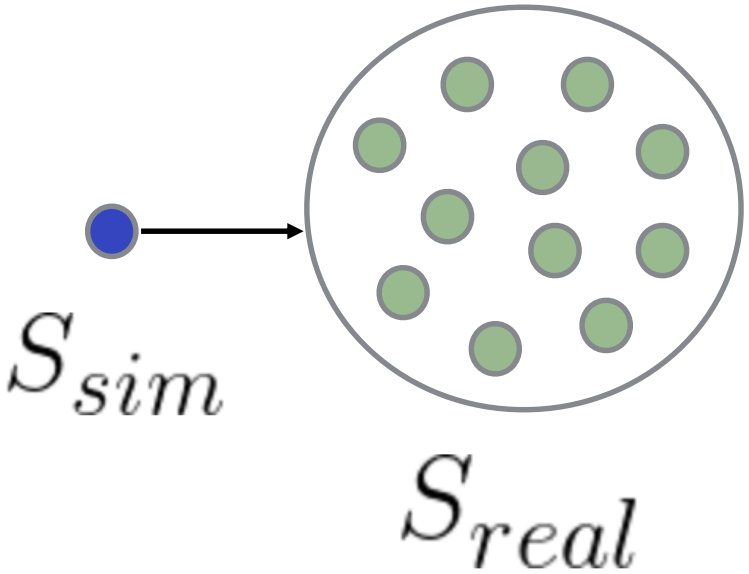
Data collection



M

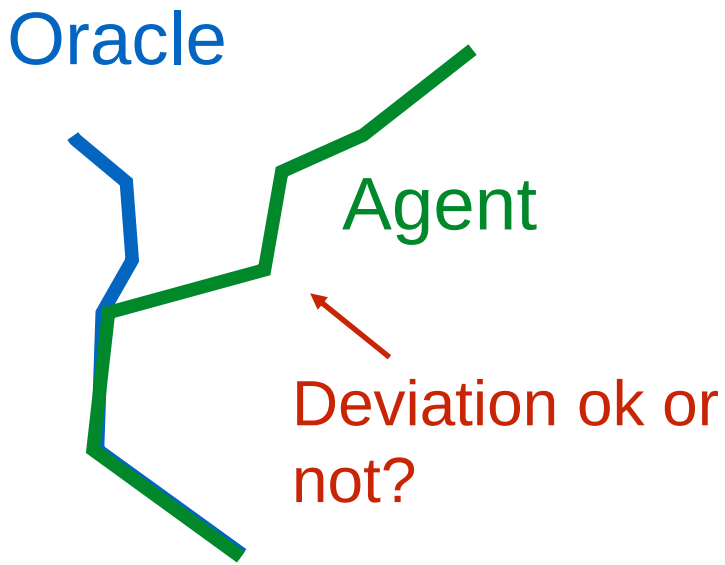
Noise and Bias in Data

State representation (SR) noise



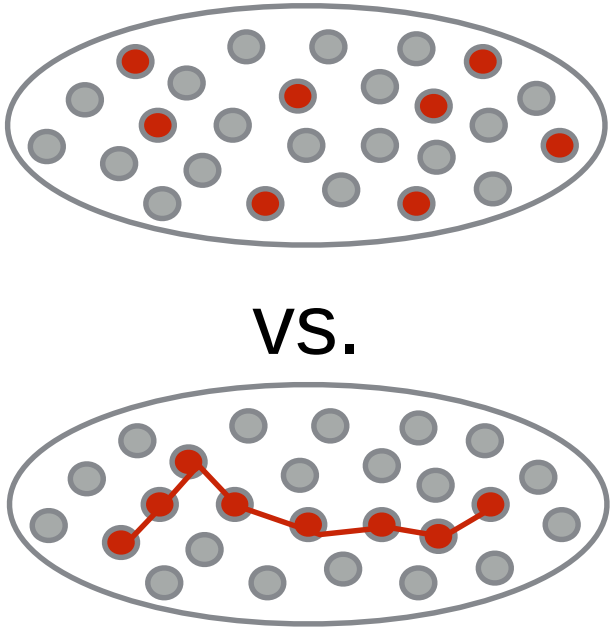
Many states look identical

Action mismatch (AM) noise



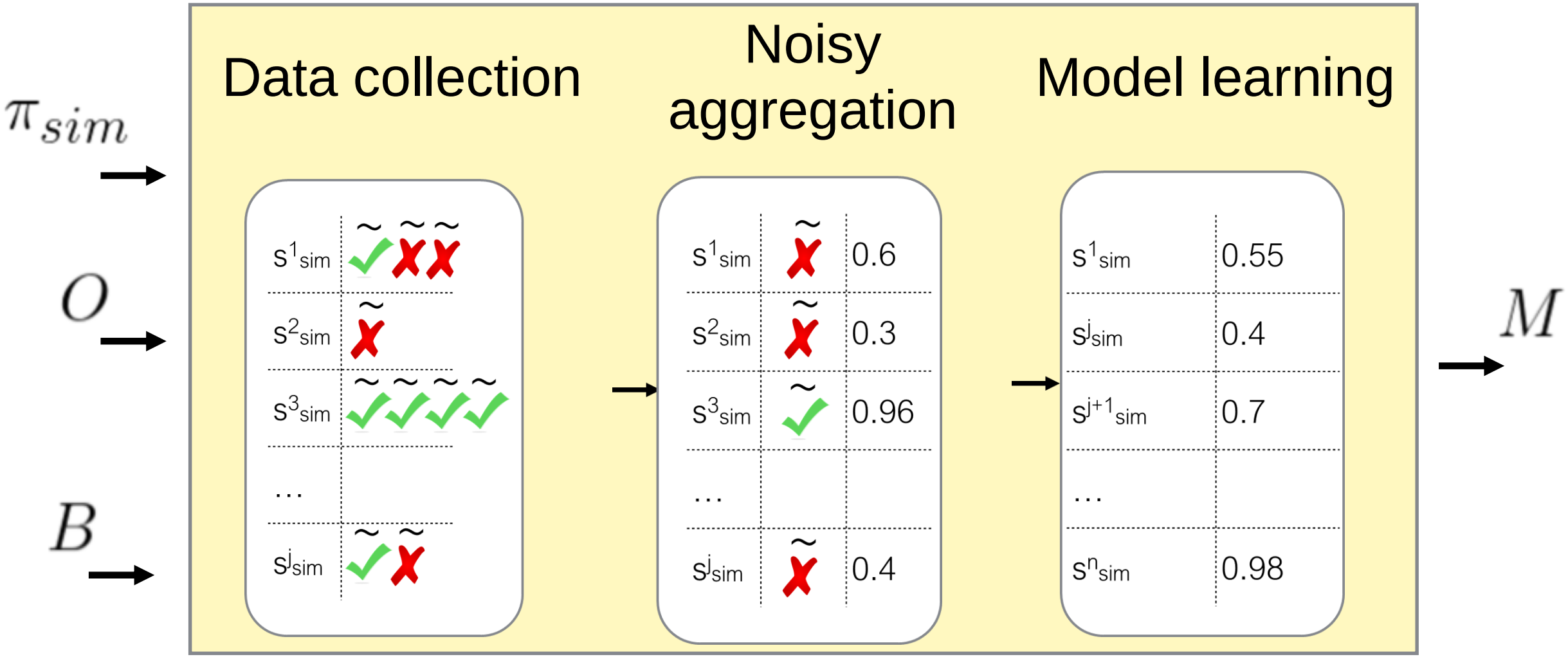
Observing oracle only shows behavior deviation

Bias in data



Trajectory-based feedback adds bias to the data

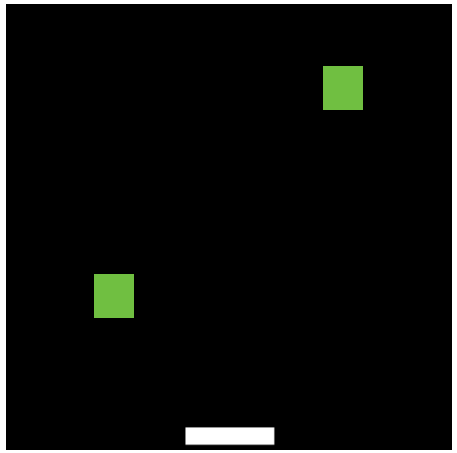
Our Approach



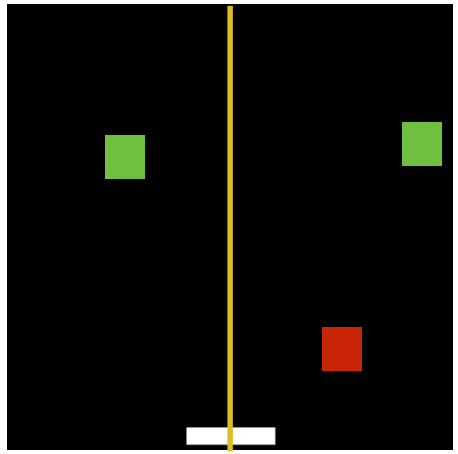
Evaluation

Catcher

Sim

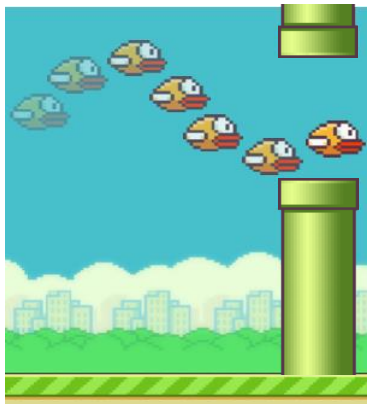


Real

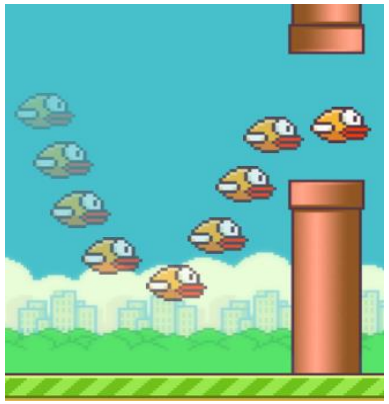


FlappyBird

Sim

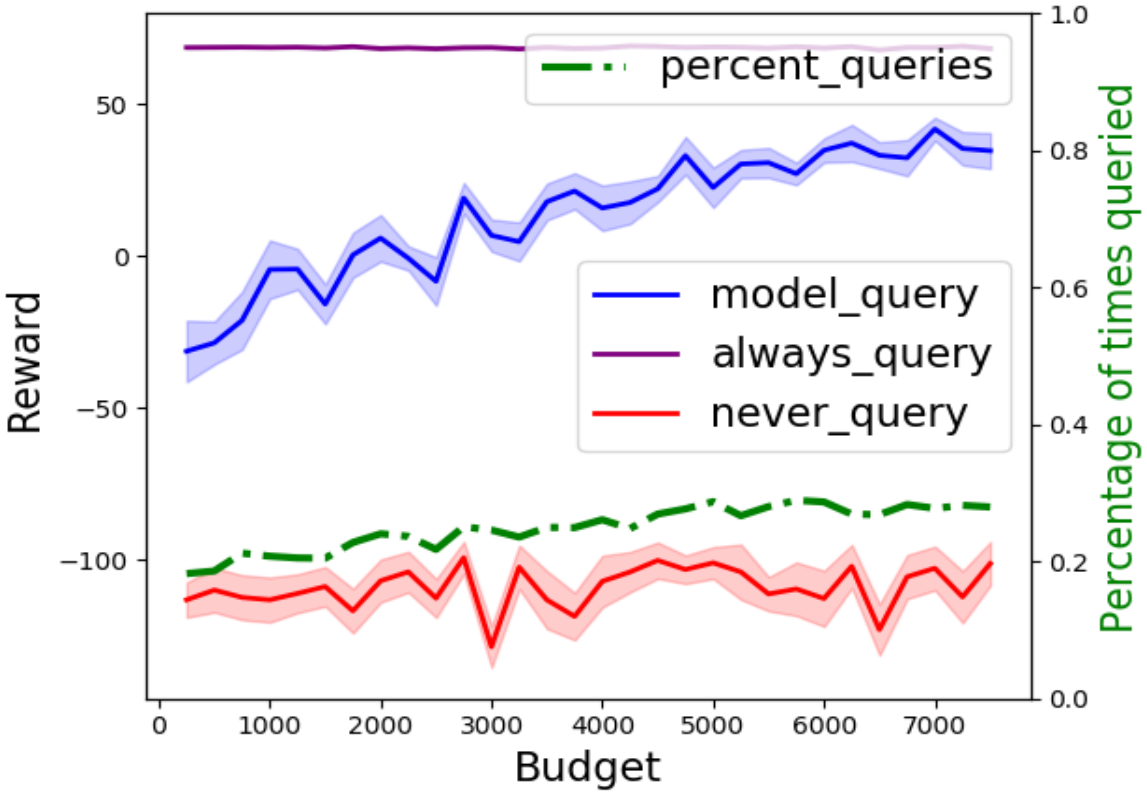


Real



Example Result – more on paper

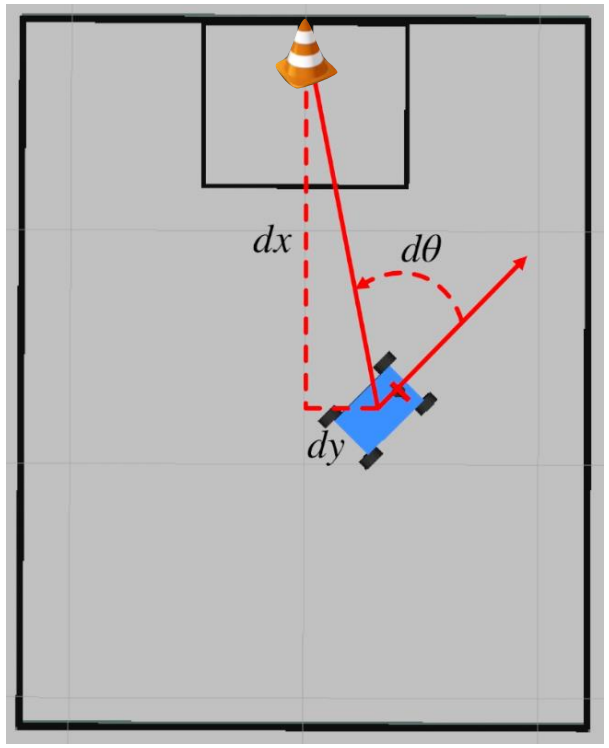
D-AM: Demonstrations



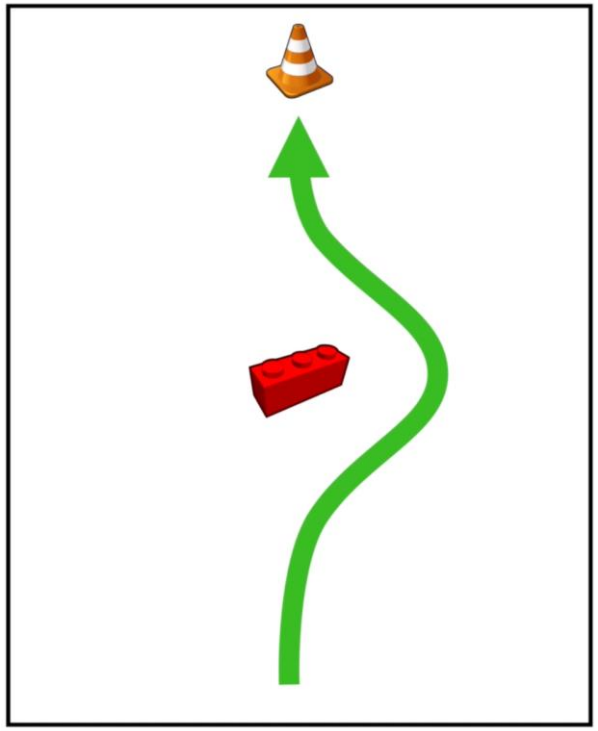
Our blind spot model can be effectively used in the real world to get 80% of the reward with less than 30% queries

Mini autonomous car demo

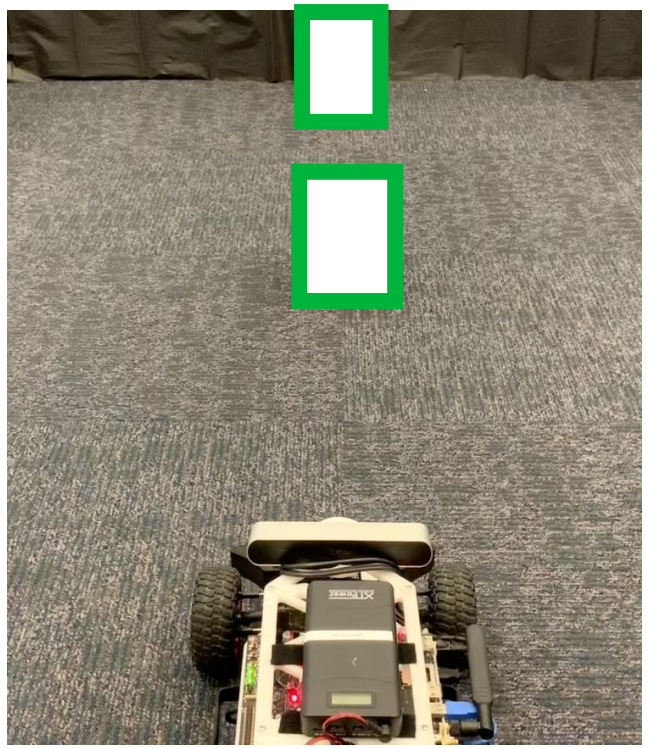
Simulation



Real-world task

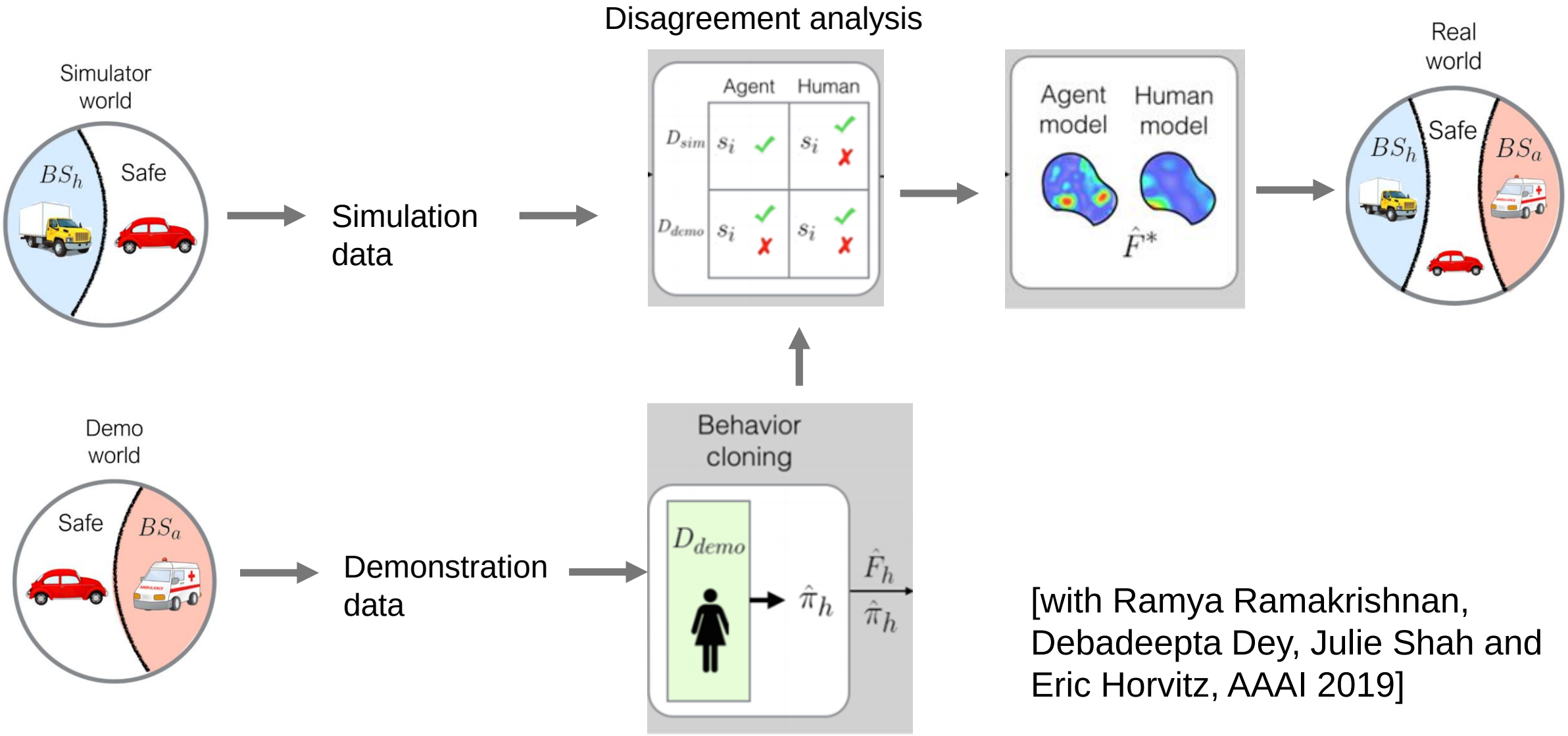


Real world

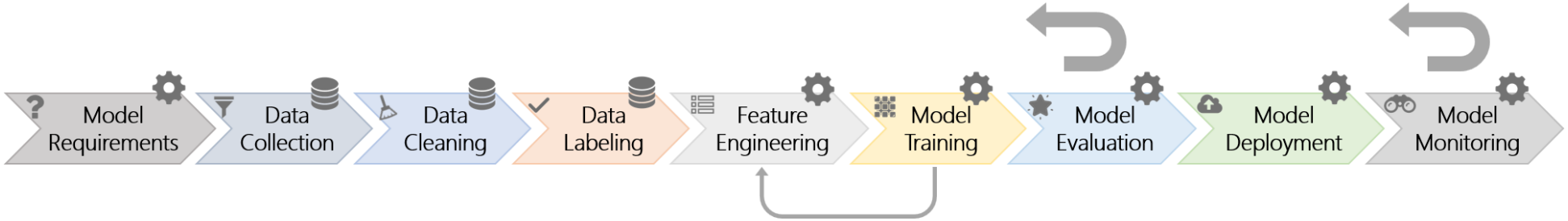


Human Blind Spots

Relaxing Assumptions: Towards Joint Execution



[with Ramya Ramakrishnan, Debadeepta Dey, Julie Shah and Eric Horvitz, AAAI 2019]



Software Engineering for Machine Learning: A Case Study

Saleema Amershi, *Microsoft Research*, Redmond, WA USA, samershi@microsoft.com
Andrew Begel, *Microsoft Research*, Redmond, WA USA, andrew.begel@microsoft.com
Christian Bird, *Microsoft Research*, Redmond, WA USA, cbird@microsoft.com
Robert DeLine, *Microsoft Research*, Redmond, WA USA, rdeline@microsoft.com
Harald Gall, *University of Zurich*, Zurich, Switzerland, gall@ifi.uzh.ch
Ece Kamar, *Microsoft Research*, Redmond, WA USA, eckamar@microsoft.com
Nachiappan Nagappan, *Microsoft Research*, Redmond, WA USA, nachin@microsoft.com
Besmira Nushi, *Microsoft Research*, Redmond, WA USA, besmira.nushi@microsoft.com
Thomas Zimmermann, *Microsoft Research*, Redmond, WA USA, tzimmer@microsoft.com

***Abstract*—Recent advances in machine learning have stimulated widespread interest within the Information Technology sector on integrating AI capabilities into software and services. This goal has forced organizations to evolve their development processes. We report on a study that we conducted on observing software teams at Microsoft as they develop AI-based applications. We consider a nine-stage workflow process informed by prior experiences developing AI applications (e.g., search and NLP) and data science tools (e.g. application diagnostics and bug reporting). We found that various Microsoft teams have united this workflow into preexisting, well-evolved, Agile-like software engineering processes, providing insights about several essential engineering challenges that organizations may face in creating large-scale AI solutions for the marketplace. We collected some best practices from Microsoft teams to address these challenges. In addition, we have identified three aspects of the AI domain that make it fundamentally different from prior software application domains: 1) discovering, managing, and versioning the data needed for machine learning applications is much more complex and difficult than other types of software engineering, 2) model customization and model reuse require very different skills than are typically found in software teams, and 3) AI components are more difficult to handle as distinct modules than traditional software components — models may be "entangled" in complex ways and experience non-monotonic error behavior. We believe that the lessons learned by Microsoft teams will be valuable to other organizations.**

***Index Terms*—AI, Software engineering, process, data**

I. Introduction

Personal computing. The Internet. The Web. Mobile computing. Cloud computing. Nary a decade goes by without a disruptive shift in the dominant application domain of the software industry. Each shift brings with it new software engineering goals that spur software organizations to evolve their development practices in order to address the novel aspects of the domain.

The latest trend to hit the software industry is around integrating artificial intelligence (AI) capabilities based on advances in machine learning. AI broadly includes technologies for reasoning, problem solving, planning, and learning, among others. Machine learning refers to statistical modeling techniques that have powered recent excitement in the software and services marketplace. Microsoft product teams have used machine learning to create application suites such as Bing Search or the Cortana virtual assistant, as well as platforms such as Microsoft Translator for real-time translation of text, voice, and video, Cognitive Services for vision, speech, and language understanding for building interactive, conversational agents, and the Azure AI platform to enable customers to build their own machine learning applications [1]. To create these software products, Microsoft has leveraged its preexisting capabilities in AI and developed new areas of expertise across the company.

In this paper, we describe a study in which we learned how various Microsoft software teams build software applications with customer-focused AI features. For that, Microsoft has integrated existing Agile software engineering processes with AI-specific workflows informed by prior experiences in developing early AI and data science applications. In our study, we asked Microsoft employees about how they worked through the growing challenges of daily software development specific to AI, as well as the larger, more essential issues inherent in the development of large-scale AI infrastructure and applications. With teams across the company having differing amounts of work experience in AI, we observed that many issues reported by newer teams dramatically drop in importance as the teams mature, while some remain as essential to the practice of large-scale AI. We have made a first attempt to create a process maturity metric to help teams identify how far they have come on their journeys to building AI applications.

As a key finding of our analyses, we discovered three fundamental differences to building applications and platforms for training and fielding machine-learning models than we have seen in prior application domains. First, machine learning is all about data. The amount of effort and rigor it takes to discover, source, manage, and version data is inherently more complex and different than doing the same with software code. Second, building for customizability and extensibility of models require teams to not only have software engineering skills but almost

A study with 565 ML engineers

End-to-end tool fragmentation

Tools make ML too difficult

Data collection and cleaning is hard

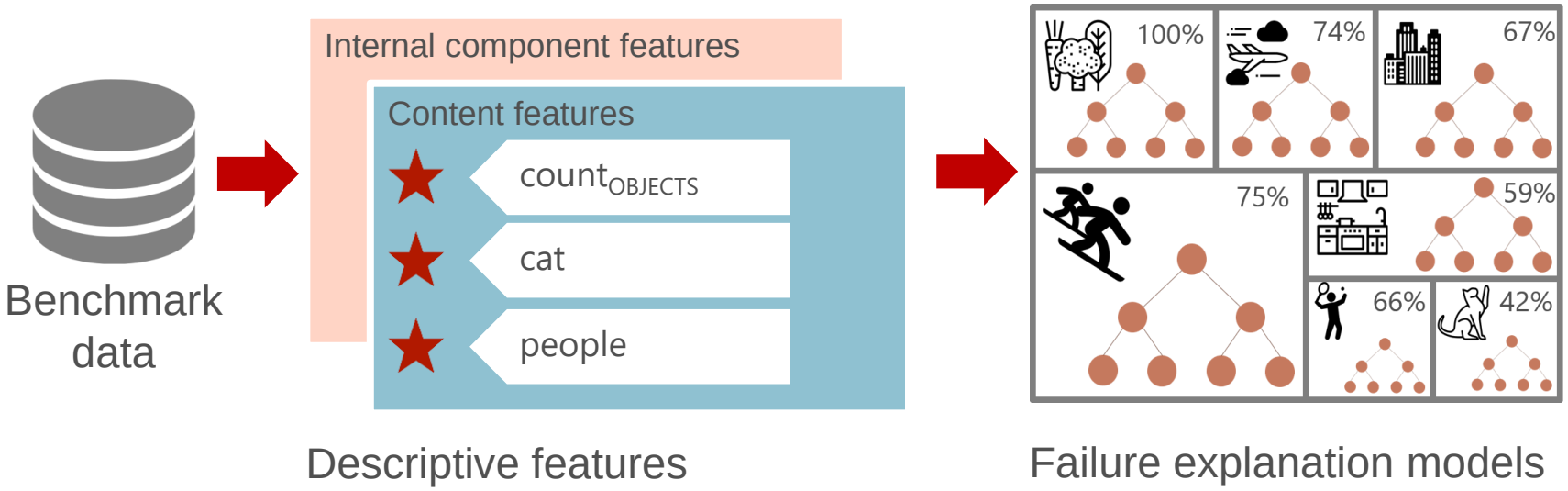
Education and lack of expertise

Debugging is hard

Model evaluation and deployment

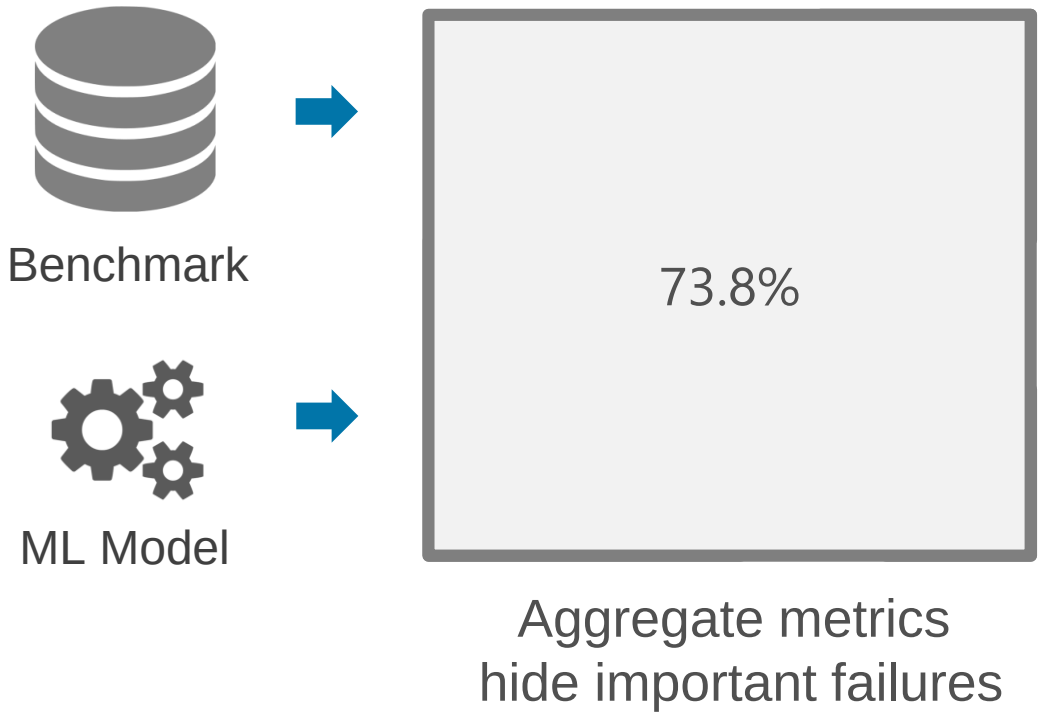
Compliance and ethics

Pandora: Error Analysis for ML systems

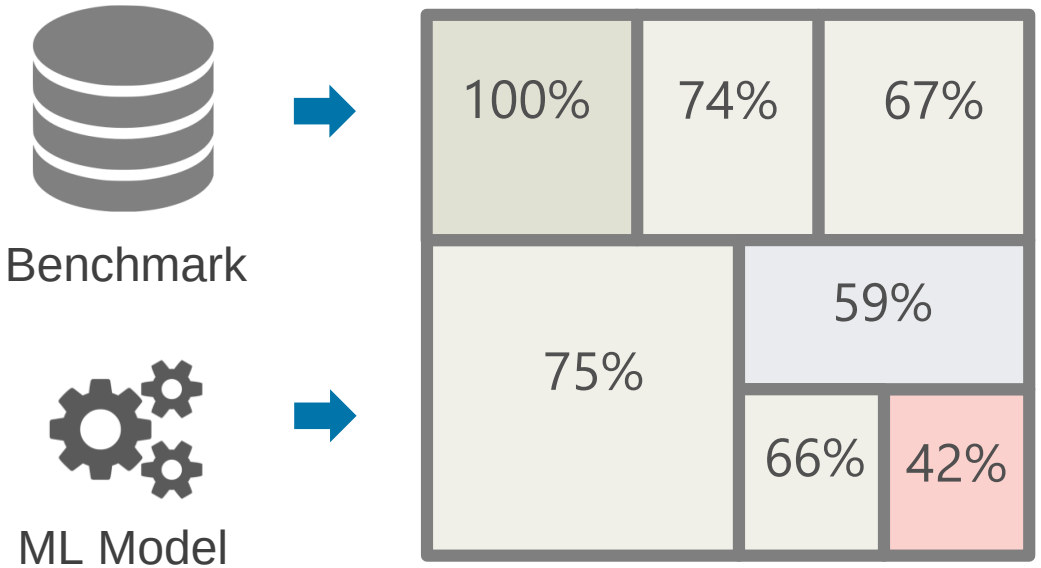


[with Besmira Nushi and Eric Horvitz, HCOMP 2018]

Current Practices of Error Detection



Key: Empowering AI developers

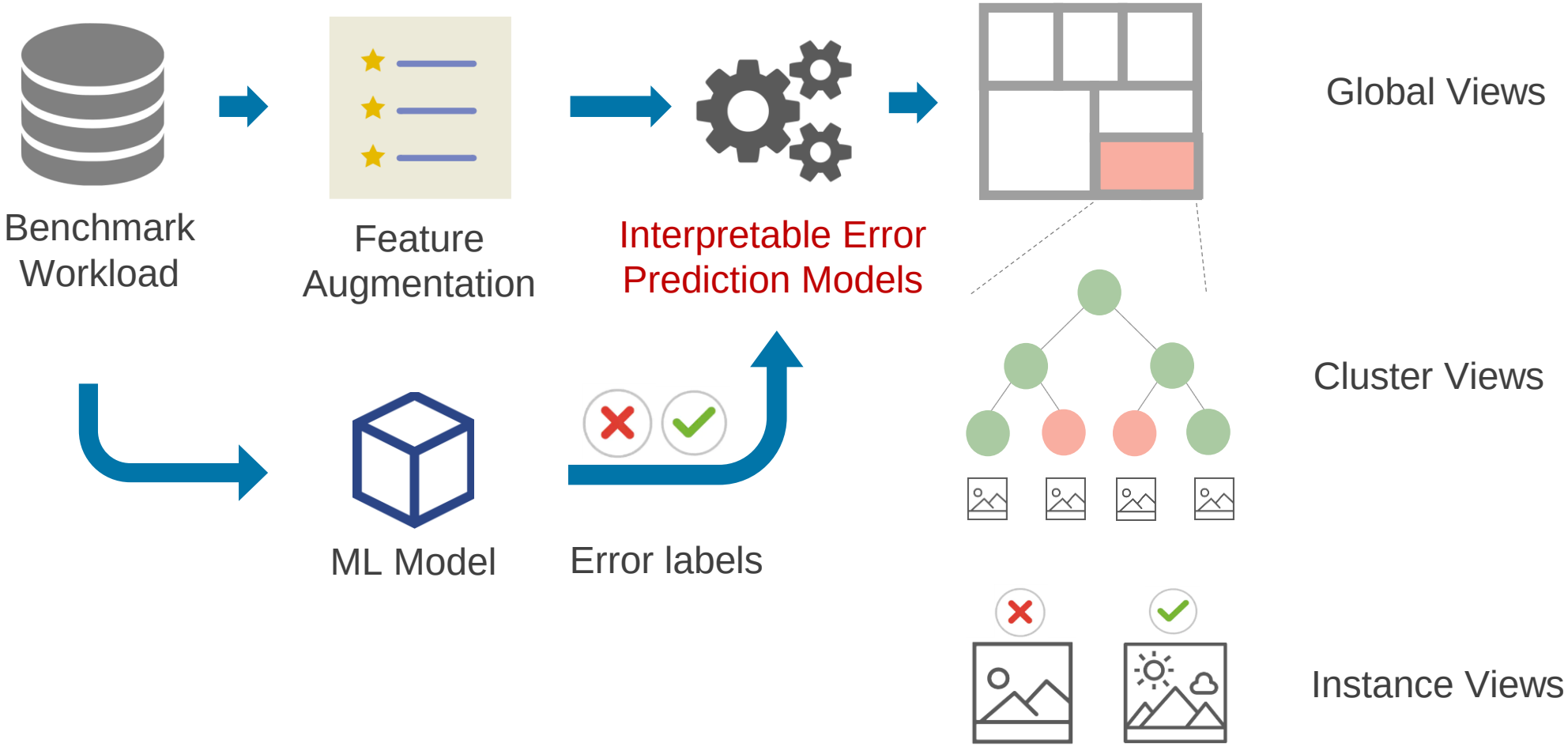


Different regions
fail for different reasons

[VISION]

"Build tools that help engineers **accelerate** the development iterations by identifying errors **faster**, **systematically**, and **rigorously**."

Tool for Error Analysis



ai-clustering

Decision Tree

Type to filter

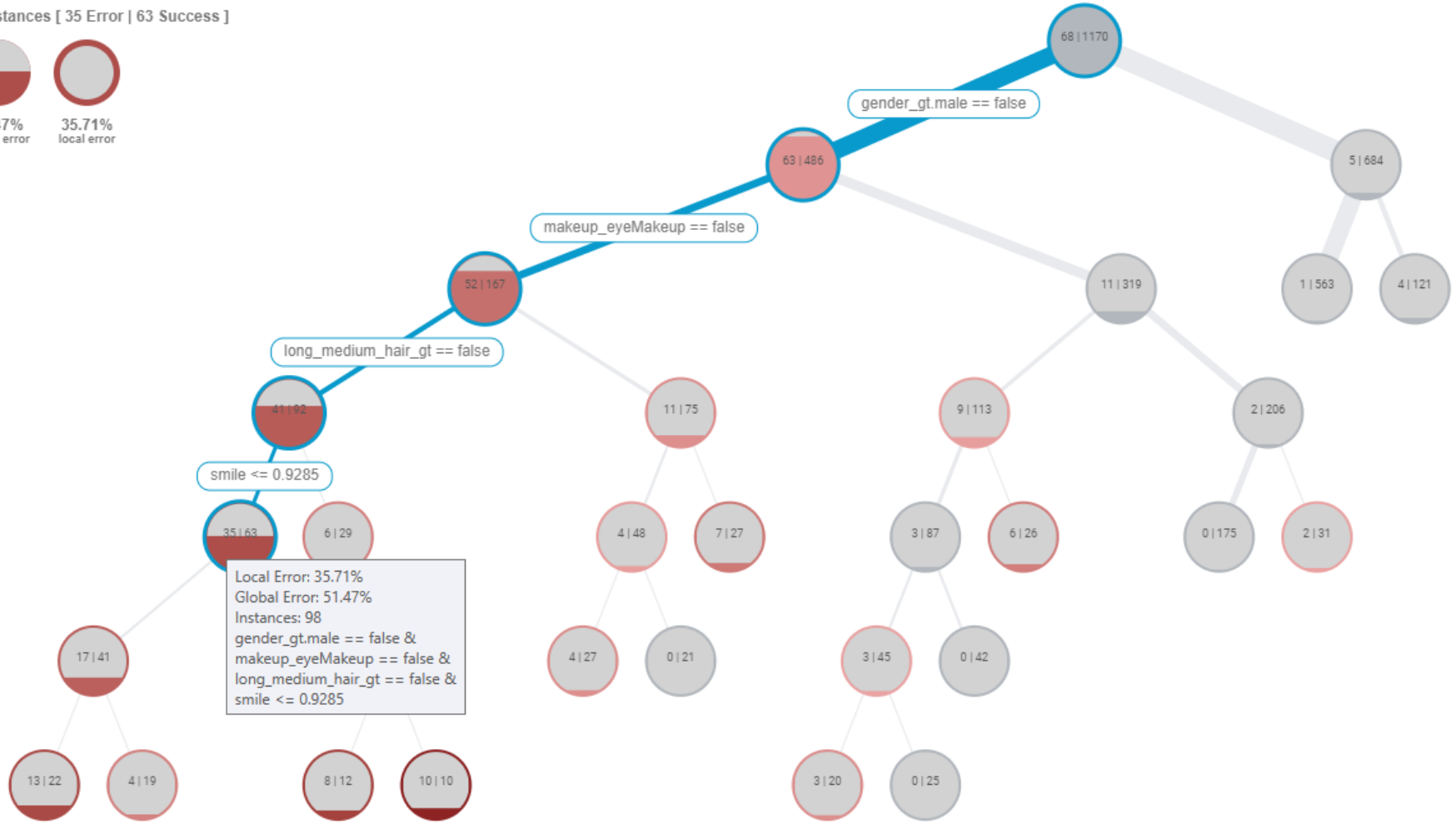
NAME	GAIN
gender_gt	
facialHair_sid	
facialHair_mc	
facialHair_be	
skin_type_gt	
hair_length_g	
accessories_	
age	
hair_bald	
smile	
noise_noiseL	
makeup_eyeM	
glasses_gt	
glasses	
hair_invisible	
occlusion_for	
exposure_exp	

APPLY

98 Instances [35 Error | 63 Success]

51.47% global error

35.71% local error



Outline

- Blind spot detection for supervised learning

  [with Himabindu Lakkaraju, Rich Caruana and Eric Horvitz, AAAI 2016]

- Blind spots of reinforcement learning

  [with Ramya Ramakrishnan, Debadeepta Dey, Julie Shah and Eric Horvitz, AAMAS 2018; AAAI 2019]

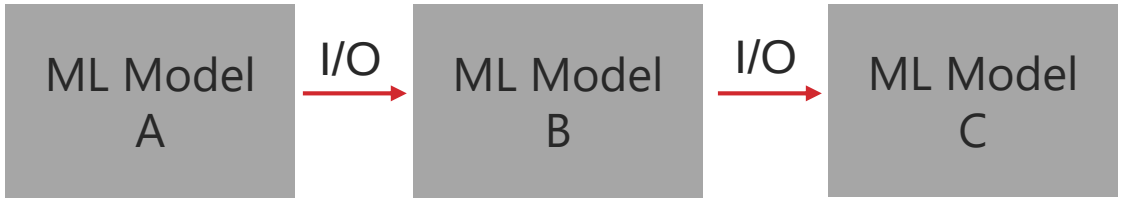
- **Backward-compatibility for AI systems**

  [with Gagan Bansal, Besmira Nushi, Dan Weld and Eric Horvitz, AAAI 2019]

Putting models into a system perspective

Sociotechnical System: Human-AI collaboration

Software System: component-component collaboration



Human-AI Collaboration

TRANSPORTATION \ CARS \ TESLA

83

Tesla can change so much with over-the-air updates that it's messing with some owners' heads

Praise for a recent software fix to the Model 3's braking is met with worry that different update slowed some customers' cars

By Sean O'Kane | @sokane1 | Jun 2, 2018, 1:00pm EDT

This week was different, though, because it showed just how far the company can go with those updates. With a swift change in the software, the company showed it can reach as deep as the systems that control the brakes. It creates the feeling that you could get out of your car one night, and by the time you get back in the next morning, the car could do some things — maybe everything — in a totally different way.

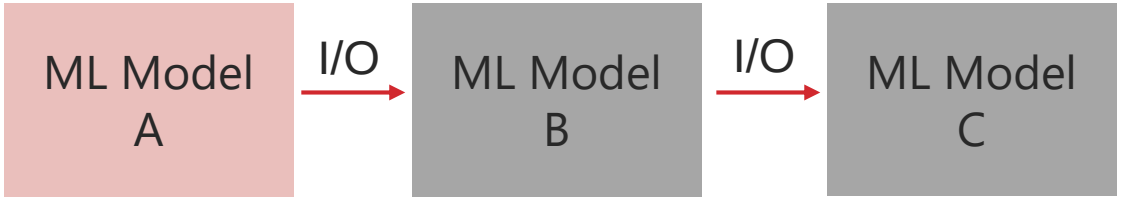
OUR PRODUCTS "ARE A BIT MYSTERIOUS, AND DO COOL THINGS, AND SOMETIMES THEY DO SOMETHING CREEPY OR HARMFUL," RINESI SAYS

Rinesi says it's also hard to define "software" in the first place since much of what modern technology does relies on things that live outside the physical object — in this case, the car. "You don't buy a car, or a phone, or soon enough a house or a medical implant or whatever: you buy an interface to, or an aspect of, a huge platform-company-ecosystem-whatever that changes by the minute," he says.

What can go wrong?

Updates can break **system** performance

Software System: component-component collaboration



- **Preserving Robustness Checks**
- **Error Handling and Chain Effects**:
- **Maintenance**

[Sculley et al. NeurIPS 2015] [Nushi et al. AAAI 2017]
[Andrist et al. ICSR 2017] [Wang et al. SIGIR 2012]

Compatibility Score

$$C(v1, v2) = \frac{\#(v1{=}Right \cap v2{=}Right)}{\#(v1{=}Right)}$$

Goal: v2 should not introduce any new errors during retraining.

Updates in Practice

$$\frac{\#(\text{v1=Right} \cap \text{v2=Right})}{\#(\text{v1=Right})}$$

Compatibility is not in-built

Classifier	Dataset	Perf. v1	Perf. v2	Compatibility
Logistic Regression	Recidivism	0.68	0.72	72%
	Credit Risk	0.72	0.77	66%
	Mortality	0.68	0.77	40%
Multi-layered Perceptron	Recidivism	0.59	0.73	53%
	Credit Risk	0.70	0.80	63%
	Mortality	0.71	0.84	76%

High-stake decision-making

Low compatibility

Training Compatible Models

Reformulated loss function

$$L_c = L + \lambda_c \cdot \mathcal{D}(v_1, v_2)$$

Dissonance

New-error dissonance

$$\mathcal{D}(x, y, v_1, v_2) = 1\,(v_1(x) = y) \cdot L(x, y, v_2)$$

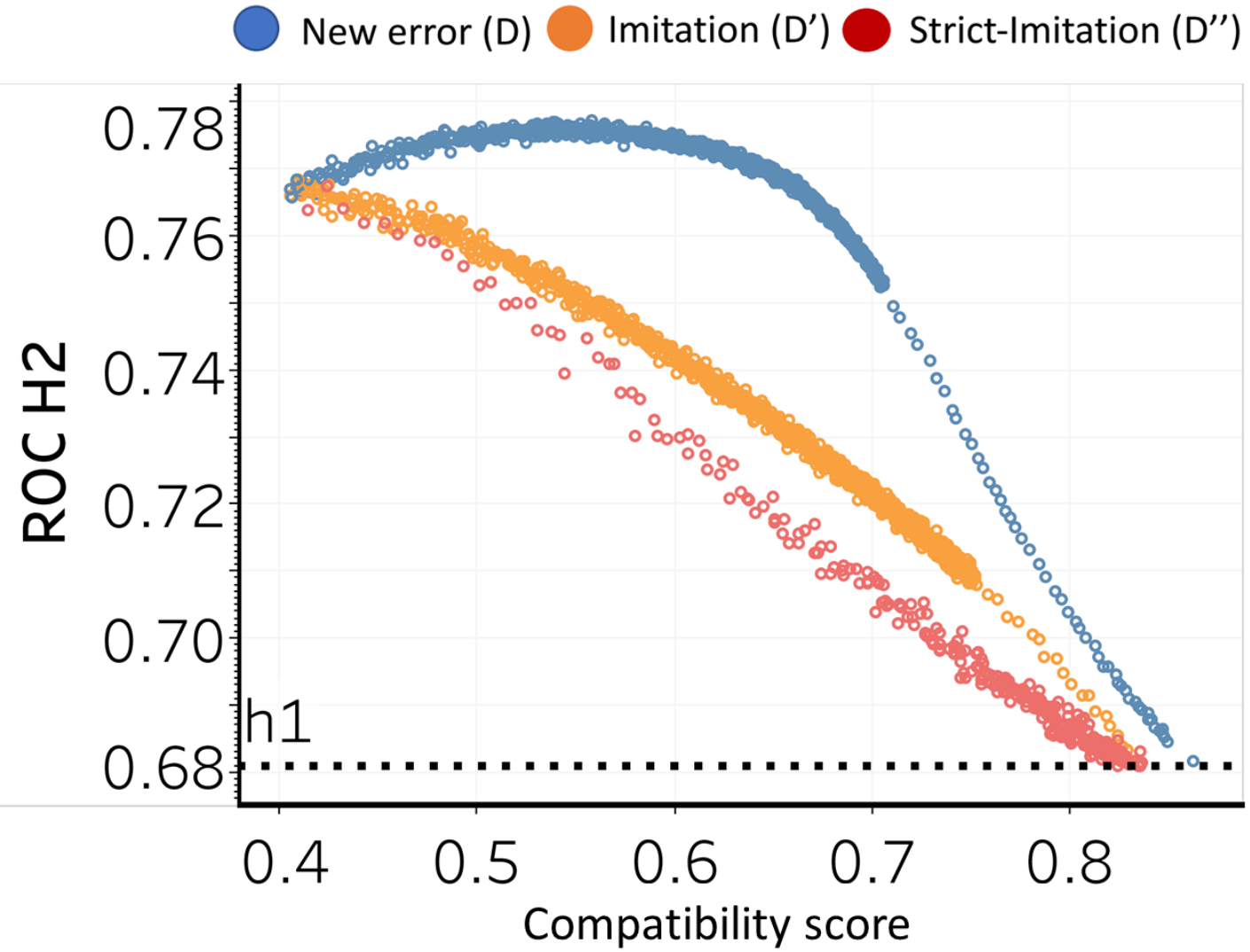
Imitation dissonance

$$\mathcal{D}(x, y, v_1, v_2) = L(x, v_1, v_2)$$

Strict imitation dissonance

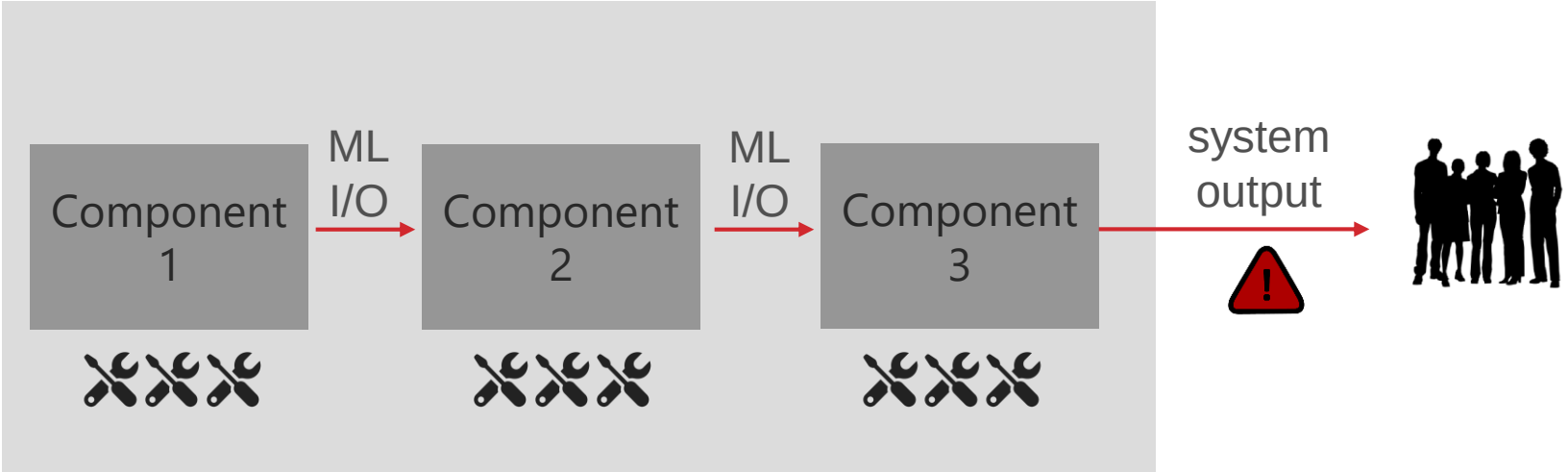
$$\mathcal{D}(x, y, v_1, v_2) = 1\,(v_1(x) = y) \cdot L(x, v_1, v_2)$$

Compatibility Graphs



FUTURE DIRECTIONS

Integrative ML Systems



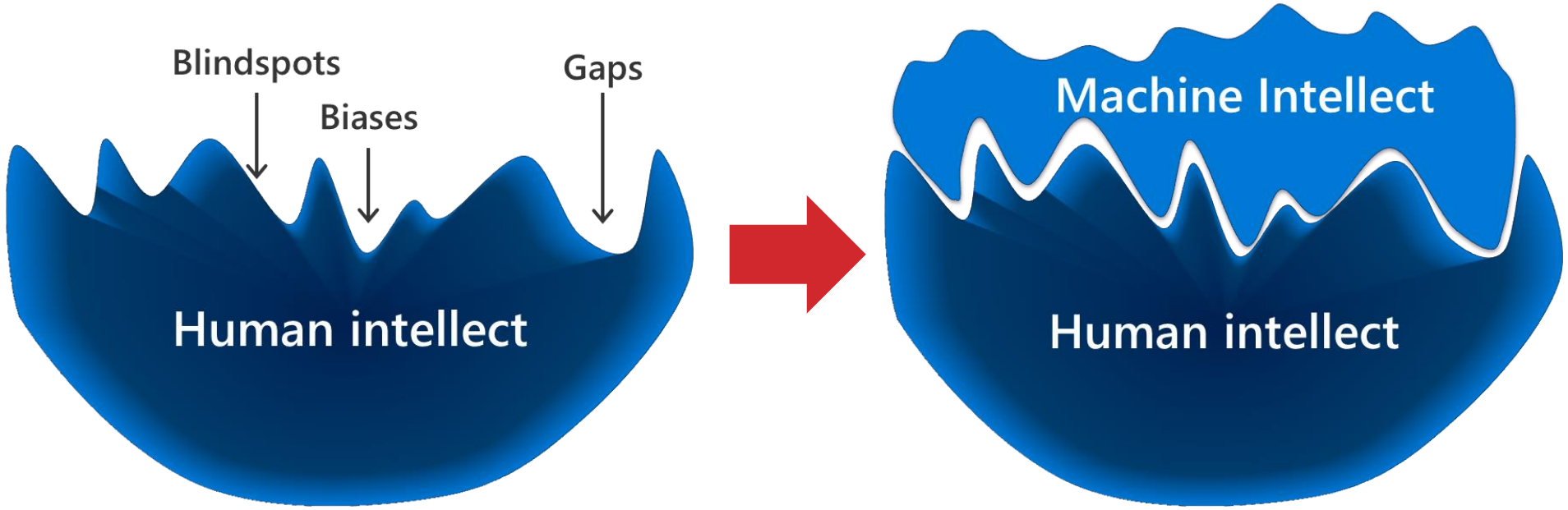
Where to invest development effort for the largest impact?

[with Besmira Nushi and Eric Horvitz, AAAI 2017]

Adversarial investigations of reliability and safety

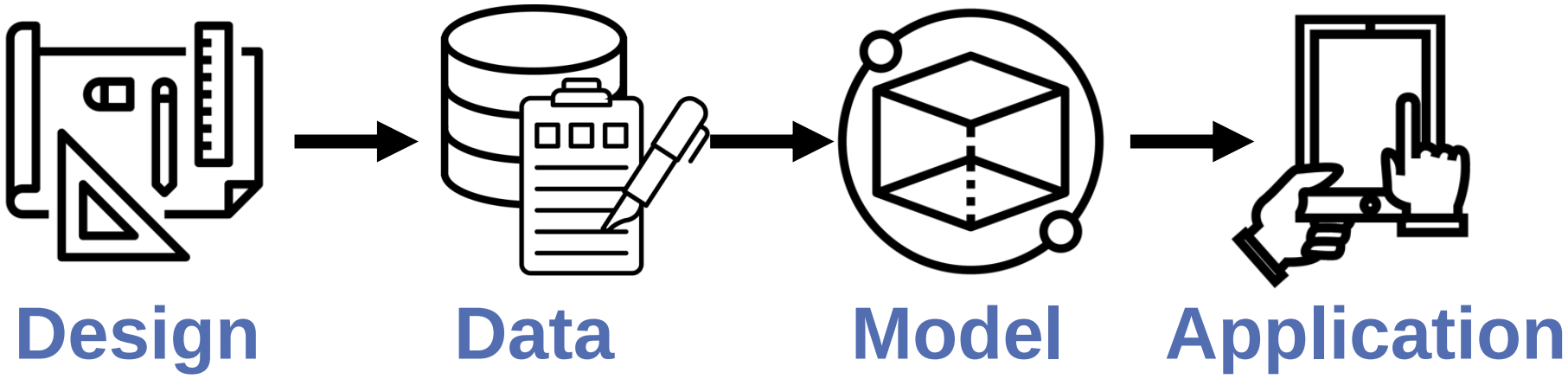
- Data poisoning
- Backdoor attacks
- Model inversion
- Model stealing
- ML supply chain attacks
- Membership inference

Directions around AI, People and Society



Takeaways

- Reliability & safety risks carry societal implications.
- Understanding AI limitations is key.
- Need for algorithmic advances as well as tools, design guidelines and frameworks for empowering developers and users.



Questions

eckamar@microsoft.com