Data Science

Artificial Intelligence

Tinglong Dai, PhD

Bernard T. Ferrari Professor

Johns Hopkins Carey Business School

https://bit.ly/elevatorstrike

https://n.pr/3TVprEO

In Sweden, if you ask a union leader, "Are you afraid of new technology?" they will answer, "No, I'm afraid of *old* technology." The jobs disappear, and then we train people for new jobs. We won't protect jobs. But we will protect workers.

Ylva Johansson, Swedish Minister for Employment and Integration

https://bit.ly/nytswetech

Agenda

1. **Week 1: Foundations**
2. **Week 2: Tools & Techniques**

   Machine learning, deep learning, Keras, Python, and no-code tools.
3. **Week 3: CNNs & Business Models**

   Convolutional neural networks, using AI to create human value. Assignment #1 due.
4. **Week 4: MLOps, RNNs, & Transformers**

   MLOps, RNNs, transformers, evaluation of AI. AI Lab proposal due.
5. **Week 5: LLMs & GenAI**

   Large language models, generative AI, using AI to create business value. Assignment #2 due.
6. **Week 6: Reinforcement Learning**

   Reinforcement learning, human-AI interaction, system design.
7. **Week 7: Responsible AI**

   Responsible AI, future of AI. Lab final deliverable due.
8. **Week 8: Individual Final Exam; Peer Reviews of AI Lab**

4th TA Tutorial: Creating Agentic AI*

***Based on Suhas' Real-World Expereince at Amazon**

🗓️ Friday, 12/5, 12:00–1:00 PM

📍 Join via Zoom: https://bit.ly/jhuaita25

Attendance is optional. Materials are on Canvas

Review: LLM API Programming

From ChatGPT User to Agent Builder

Why API Instead of Just ChatGPT?

ChatGPT UI

The ChatGPT web interface is optimized for **one-off, interactive conversations**: great for drafting, exploring ideas, and quick Q&A. It's your personal assistant for immediate needs.

While excellent for individual productivity, UI doesn't scale to organizational needs or integrate with your existing business systems.

LLM API

The **API** is optimized for **systems**: you can automate workflows, connect to databases, integrate with CRM/ERP, schedule jobs, and build products that run 24/7. Through the API, you can get larger context windows and direct file inputs, which enable tasks like full-report analysis, contract review, and multi-document synthesis that may exceed practical limits of the UI.

APIs also give you clear **versioning and cost control**: each call is logged and billable, which is essential for budgeting in a business context.

1

ChatGPT UI

Human has a conversation

2

Your App + API

Automated, repeatable systems

3

Customers & Employees

Scalable business value

Six Steps in Building AI Agents

01

Secure Your API Key

Authentication and environment setup

02

Install SDK & Make Text Calls

Your first "Hello, world" with LLMs

03

Analyze Images & Files

Multi-modal business applications

04

Extend with Tools

Web search, databases, custom APIs

05

Stream Responses

Real-time user experiences

06

Build Agents

Orchestrate complex workflows

Step 1 – Secure Your API Key

The API key is like a password for your application and must **never** be shared or hard-coded into public repositories. Think of it as the keys to your organization's AI infrastructure—losing control of this key could mean unauthorized usage and unexpected costs. The key should be stored as an environment variable, which the SDK will automatically read, keeping it separate from your codebase and version control systems.

On macOS and Linux systems, you can export the key directly in your terminal session. Similar can be done for Windows systems and for Colab notebooks.

```
export OPENAI_API_KEY="your_api_key_here"
```

Security Best Practice

Never commit API keys to Git repositories. Use environment variables, secret management services (like AWS Secrets Manager or Azure Key Vault), or configuration files that are explicitly excluded from version control via .gitignore.

Cost Management

Each API key can have usage limits and monitoring enabled through the OpenAI dashboard. Set up alerts to prevent unexpected charges and track usage patterns across different projects or teams.

Step 2 – Install SDK and Call the Model

To use the OpenAI API in Python, you can use the official OpenAI SDK for Python. Get started by installing the SDK using pip:

```
pip install openai
```

Once the SDK is installed, you create a client and send your first text request. This is the "Hello, world" of LLM APIs: one function call that turns a natural-language input into a natural-language output.

```
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Write a one-sentence bedtime story about a unicorn."
)

print(response.output_text)
```

1

Configure Client
Initialize with your API key from environment

2

Specify Model
Choose the right model for your use case

3

Provide Input
Send your natural language prompt

4

Read Response
Extract output_text from the result

https://bit.ly/llmapi25

Step 3 – Using Image Inputs

API can analyze images, combining text and visual information in a single request. This multi-modal capability opens up entirely new categories of business applications.

```
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input=[
        {
            "role": "user",
            "content": [
                {
                    "type": "input_text",
                    "text": "What teams are playing in this image?",
                },
                {
                    "type": "input_image",
                    "image_url": "https://upload.wikimedia.org/wikipedia/commons/3/3b/LeBron_James_Layup_%28Cleveland_vs_Brooklyn_2018%29.jpg"
                }
            ]
        }
    ]
)

print(response.output_text)
```

Step 4 – Extending the Model with Tools

The tools concept is where the API becomes truly powerful for business applications. Tools give the model "buttons it can press" to fetch data or trigger actions.

```
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    tools=[{"type": "web_search"}],
    input="What was a positive news story from today?"
)

print(response.output_text)
```

Web Search

Access current information, news, and real-time data beyond the model's training cutoff

Database Queries

Retrieve customer records, inventory data, or business metrics from your systems

Custom APIs

Connect to internal tools, CRM platforms, or third-party services for seamless workflows

File Search

Search through uploaded documents, knowledge bases, or document repositories

https://bit.ly/llmapi25

Step 5 – Streaming Responses

Streaming lets you show partial results as they are generated, dramatically improving user experience. This is critical for chatbots, dashboards, and interactive tools where users expect fast, incremental feedback rather than waiting for a complete response. Think about the difference between watching text appear word-by-word versus staring at a loading spinner for 30 seconds; streaming transforms the perceived responsiveness of your application.

```
from openai import OpenAI
client = OpenAI()

stream = client.responses.create(
    model="gpt-5",
    input=[
        {
            "role": "user",
            "content": "Say 'double bubble bath' ten times fast.",
        },
    ],
    stream=True,
)

for event in stream:
    print(event)
```

Without Streaming: User waits 20-30 seconds staring at a loading indicator, then sees the complete response all at once. Higher perceived latency and potential for user abandonment.

With Streaming: User sees text appearing within 1-2 seconds and can start reading immediately. Lower perceived latency and better engagement, even if total time is similar.

Step 6 – Building Agents That Take Action

Agents are higher-level constructs that can decide what to do next: which tools to call, which sub-agent to hand off to, and when to stop.
Agents are workflow engines powered by language models that can route work, draft responses, and interact with software on behalf of users. This represents a fundamental shift from simple request-response patterns to autonomous systems that can handle complex, multi-step business processes.

```
from agents import Agent, Runner
import asyncio

spanish_agent = Agent(
    name="Spanish agent",
    instructions="You only speak Spanish.",
)

english_agent = Agent(
    name="English agent",
    instructions="You only speak English",
)

triage_agent = Agent(
    name="Triage agent",
    instructions="Handoff to the appropriate agent based on the language of the request.",
    handoffs=[spanish_agent, english_agent],
)


async def main():
    result = await Runner.run(triage_agent, input="Hola, ¿cómo estás?")
    print(result.final_output)
```

The triage agent routes the request to the right language agent, demonstrating multi-agent orchestration. This pattern scales to far more complex scenarios: routing customer inquiries to specialized support agents, coordinating between research and writing agents for report generation, or managing approval chains that involve multiple stakeholders.

Taken Together:

https://bit.ly/llmapi25

3rd TA Tutorial: LLM API Programming and AI Agents

https://bit.ly/AIAgent25

Reinforcement Learning

Three Types of Learning in AI

Learning

Supervised

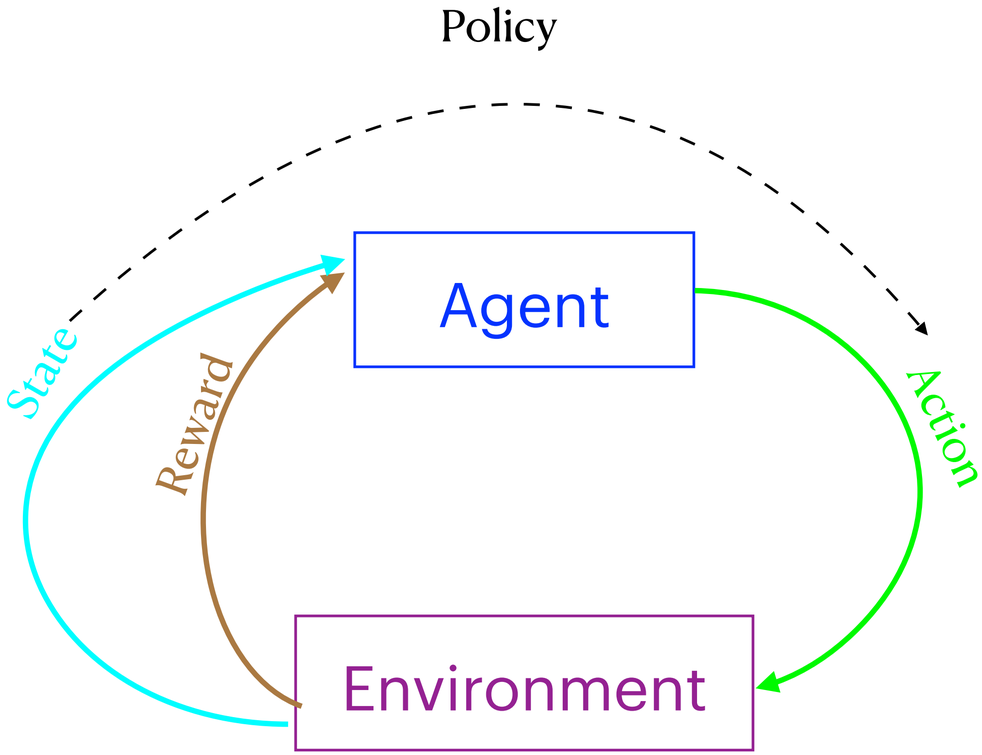
Unsupervised

Reinforcement Learning

Reinforcement Learning

- A reinforcement learning model **learns by doing**
  - By contrast, a supervised or unsupervised learning model learns from data

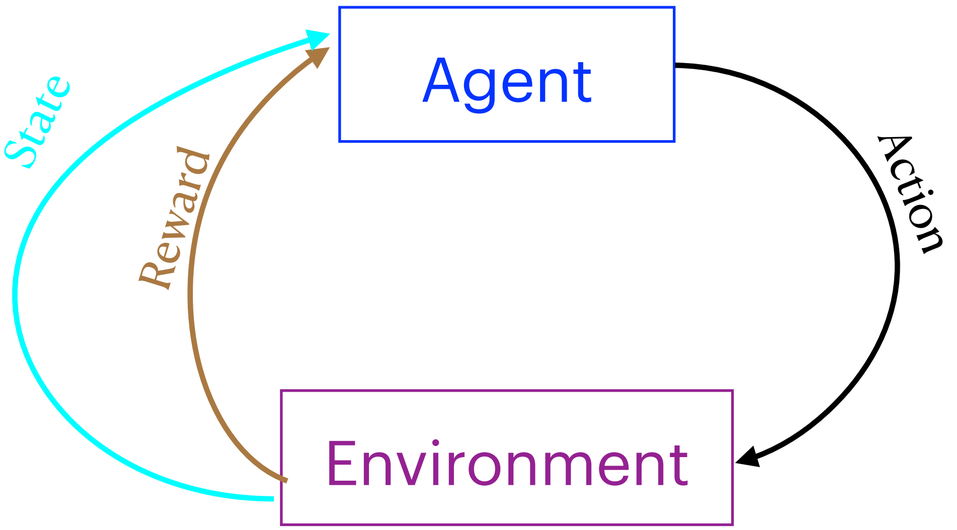
- A reinforcement learning model is an **agent** who takes a sequence of **actions** within some **environment** and receives direct feedback (**reward** and **state**) on the actions it takes



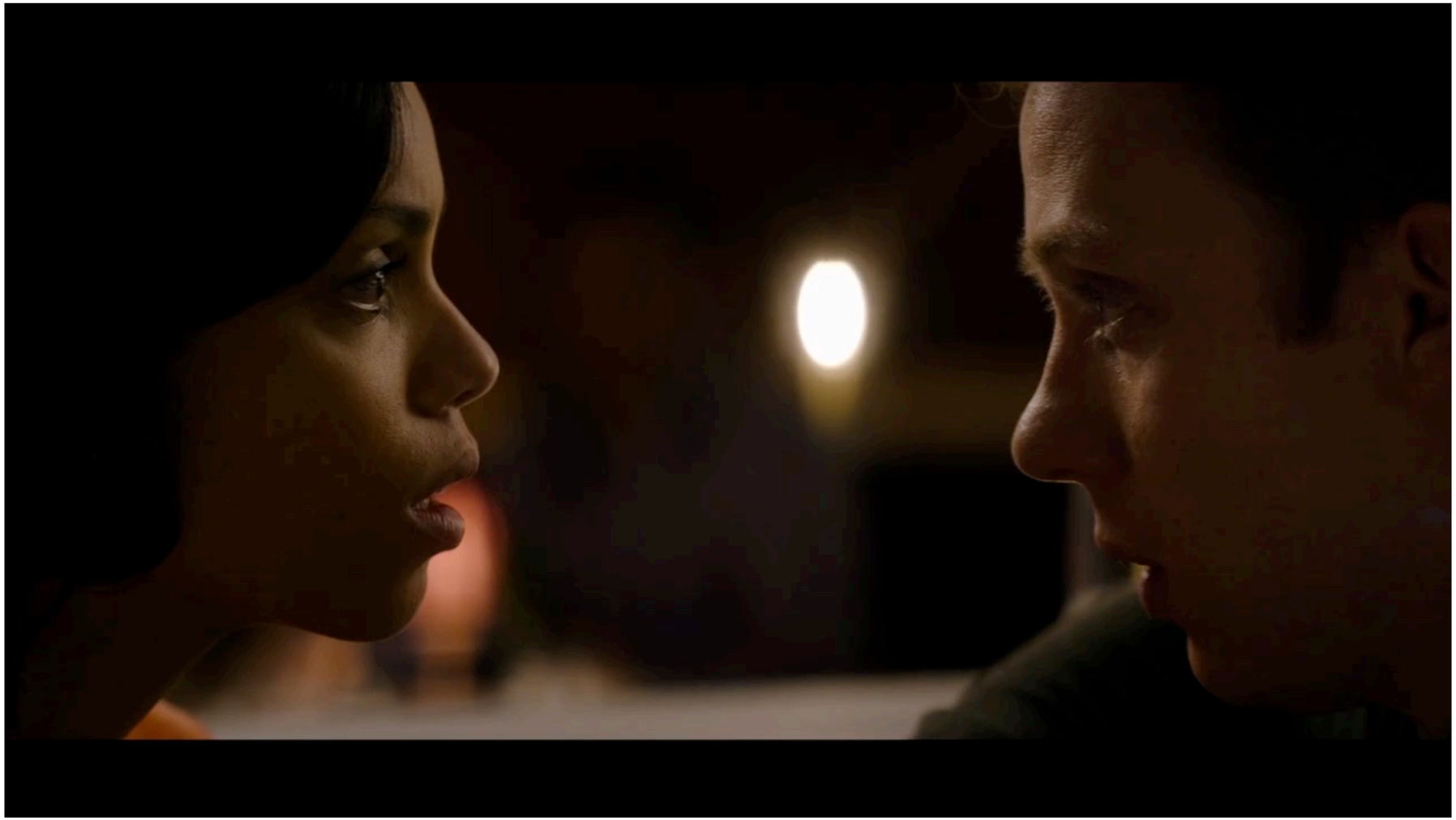
Elements of Reinforcement Learning

The Case of Roomba

- **Actions**: Moving direction and speed
- **Environment** (the floor) returns information back to the agent
  - **State**: Where the Roomba is
  - **Reward**: The agent's score (floor cleaning without incidents)
- **Policy**: The algorithm that the software agent uses to determine its actions (i.e., the policy tells the agent what to do in each state)



Reinforcement Learning



Black Mirror, Season 4, Episode 4, “Hang the DJ”

Q-Learning

Q = Quality

- We define the quality of an action by how rewarding it is, which consists of
  - instant reward
  - indication of future reward
- Quality = instant reward + indication of future reward
  $Q(s, a) = r + \gamma \cdot \max Q(s', a')$
  - $Q(s, a)$ is the quality value we are calculating for given state s and action a
  - r is the instance reward we achieve by performing the action a in the current state s
  - $\max Q(s', a')$ is the maximum quality of the resulting state s'
  - γ is the discount factor, i.e., how much importance to give to future rewards versus immediate rewards. A high value prioritizes future rewards; a low value prioritizes short-term outcomes

Q-Learning Algorithm

Four Steps

- **Step 1.** For each episode, record the entire sequence of moves and the outcome
- **Step 2.** Assign a reward to the last move the machine makes as its quality value
- **Step 3.** For each previous move, update its quality value using the Bellman equation: $Q(s,a) = Q(s,a) + \alpha \cdot (r + \gamma \cdot \max Q(s',a') - Q(s,a))$
  - α is the learning rate to ensure the algorithm learns the optimal policy gradually
  - If α is too big, the algorithm may learn wrong lessons
- **Step 4.** Using the learned model, the agent chooses an action a to maximize $Q(s,a)$

Exploration vs. Exploitation

- Now, let's rethink **Step 4,** what could possibly go wrong?
  - Using the value tables $Q(s, a)$, the agent chooses an action a to maximize $Q(s, a)$

- Problem: The learned model $Q(s, a)$ is not the same as the true environment, so acting upon the learned model may lead to suboptimal results in the long run

- The agent must make a tradeoff between **exploitation** to maximize its rewards based on the current model and **exploration** to maximize its long-run being

- To ensure the agent explores sufficiently, we often add an **exploration rate** ϵ, which is the probability that the agent takes a random action

Three Key Parameters

- α: learning rate, which determines to what extent newly acquired information overrides old information

- ϵ: exploration rate, which determines the probability of taking a random action, rather than action that gives a maximum value of Q

- γ: discount rate, which determines the importance of future rewards vs. instant rewards

Let’s explore a simulator to make sense of Q-learning:

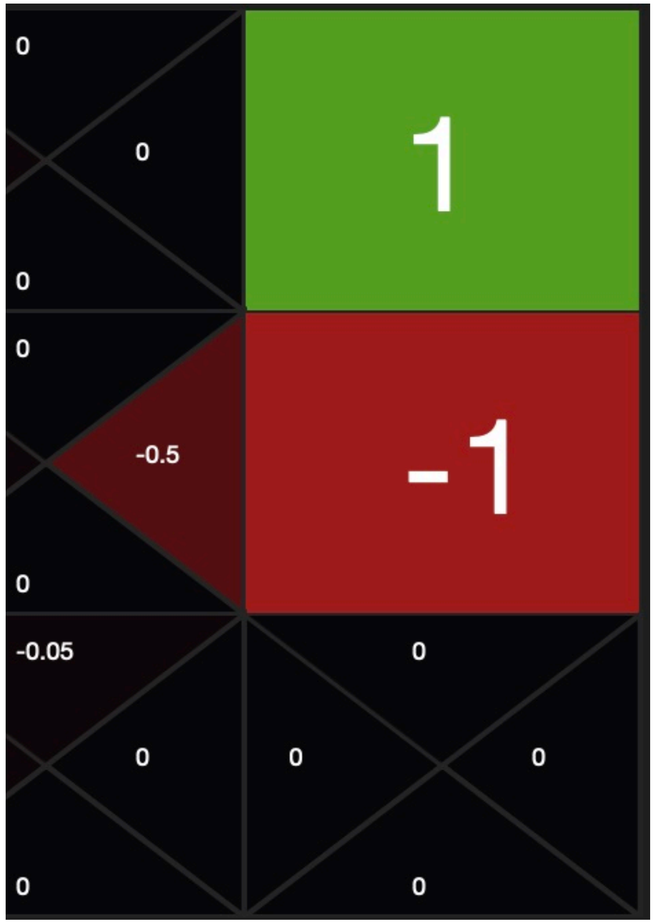
https://bit.ly/qsimulator

Q-Learning Simulator

- In this 4×3 maze, Q-learning agent learns by trial and error from interactions with the environment
- Agent starts the episode in the bottom left corner
- The action that is optimal for each state is the action that has the highest long-term reward
- Episode terminates when the agent reaches $+1$ or -1 state; in all other states, the agent receives an instant reward of -0.1
- Essence of the problem: Figuring out a "policy" to get to the green destination with the fewest moves

How Bellman Equation Works

- Recall that "Episode terminates when the agent reaches $+1$ or -1 state, in all other states agent, will receive an instant reward -0.1."
- Epoch 0 terminates when the agent hits the -1 state
- Under $\alpha = 0.5, \gamma = 0.9, \epsilon = 0.3$, the Q value of the last move is $0 + \alpha(-1) = -0.5$
- Note that once the agent is in the -1 state, it can no longer move.



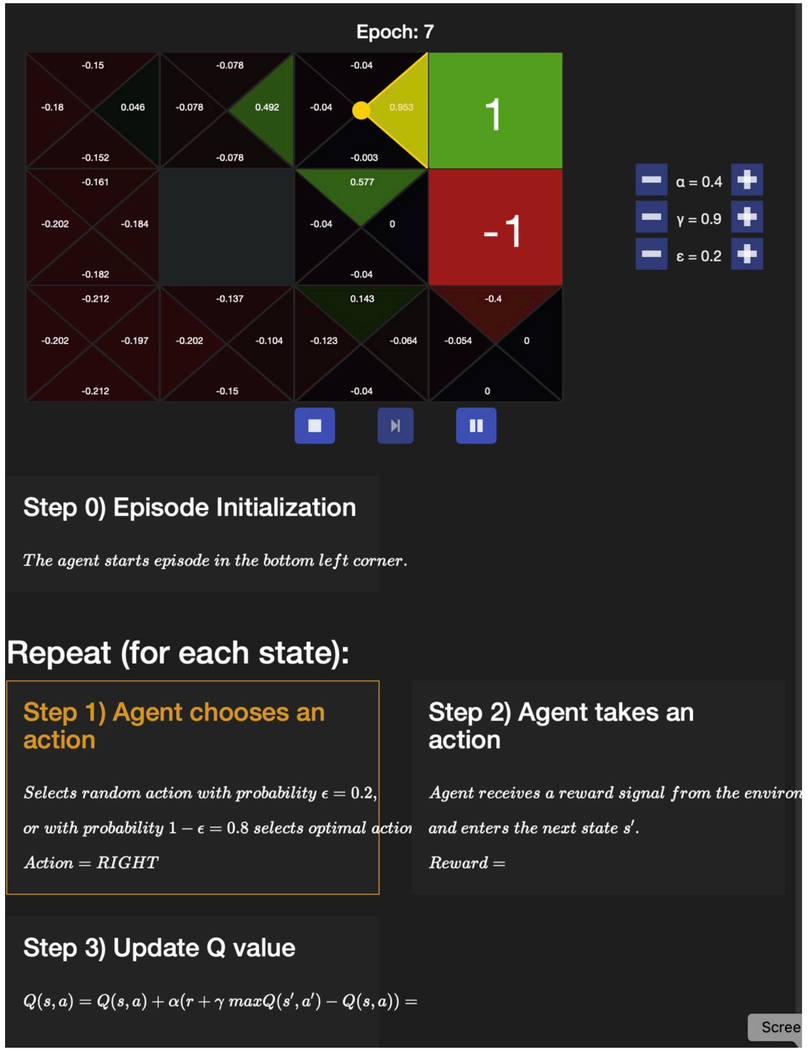
Experiment the Following Settings (10 minutes)

1. Default parameters: $\alpha = 0.1, \gamma = 0.9, \epsilon = 0.3$

2. A high learning rate: $\alpha = 0.9, \gamma = 0.9, \epsilon = 0.3$
   - What's the effect of the learning rate?

3. A low discount rate: $\alpha = 0.9, \gamma = 0.1, \epsilon = 0.3$
   - What's the effect of the discount rate?

4. A high exploration parameter: $\alpha = 0.9, \gamma = 0.9, \epsilon = 0.9$
   - What's the effect of the exploration parameter?



Explore at home: Identify a set of parameters that learns the optimal policy in an efficient manner

https://bit.ly/qsimulator

Example: Reinforcement Learning for Dynamic Pricing

https://bit.ly/RLPricing25

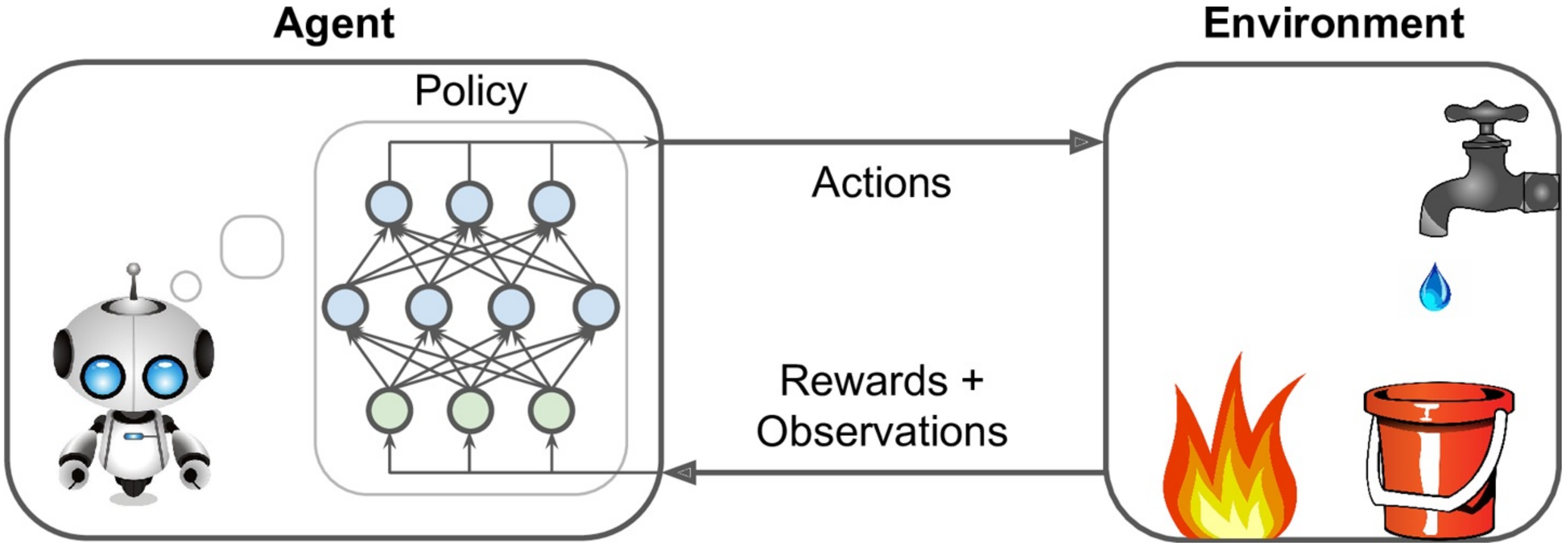
- The Challenge
  - Demand for latte changes with weather: cold & rainy → high demand, warm & sunny → lower demand.
  - We must choose a latte price each day that maximizes daily revenue, without knowing demand upfront.

- Our RL Approach
  - Let an AI "barista" learn by trial and error instead of using a fixed pricing rule.
  - For each weather–price pair, keep a Q-value = expected daily revenue (Q-table).
  - Use ε-greedy: sometimes explore random prices, otherwise exploit the best-known price.

Let's try it out and see how it works:

https://bit.ly/RLPricing25

Deep Reinforcement Learning

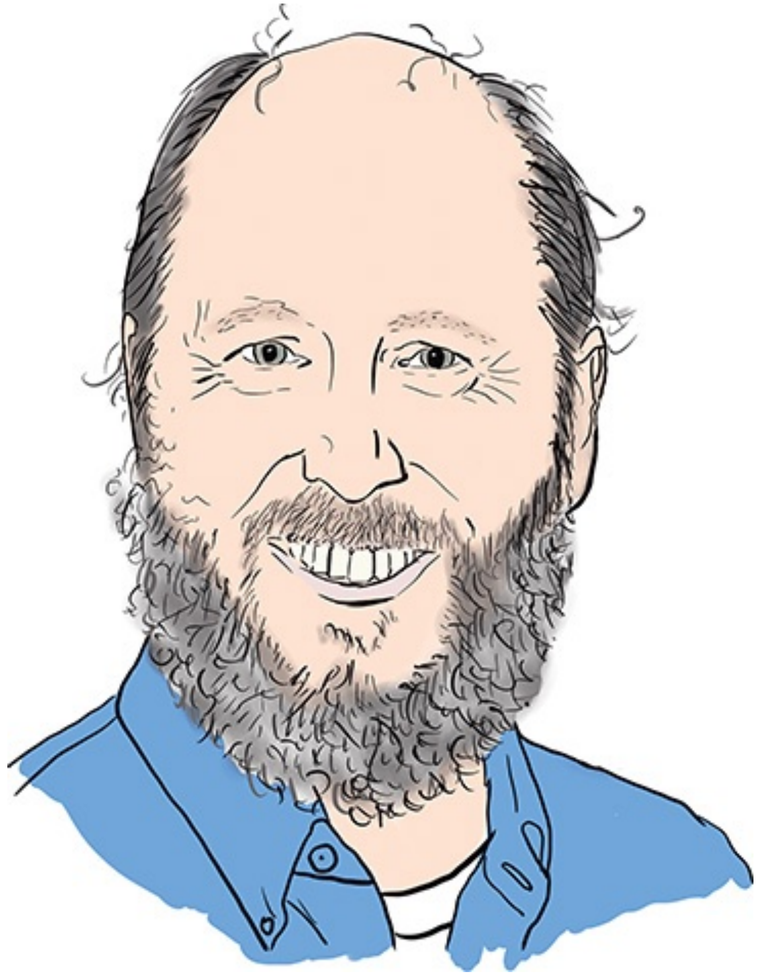


In deep reinforcement learning, the policy is generated by an artificial neural network (to be learned), instead of through maximizing a given function

Geron (2019)

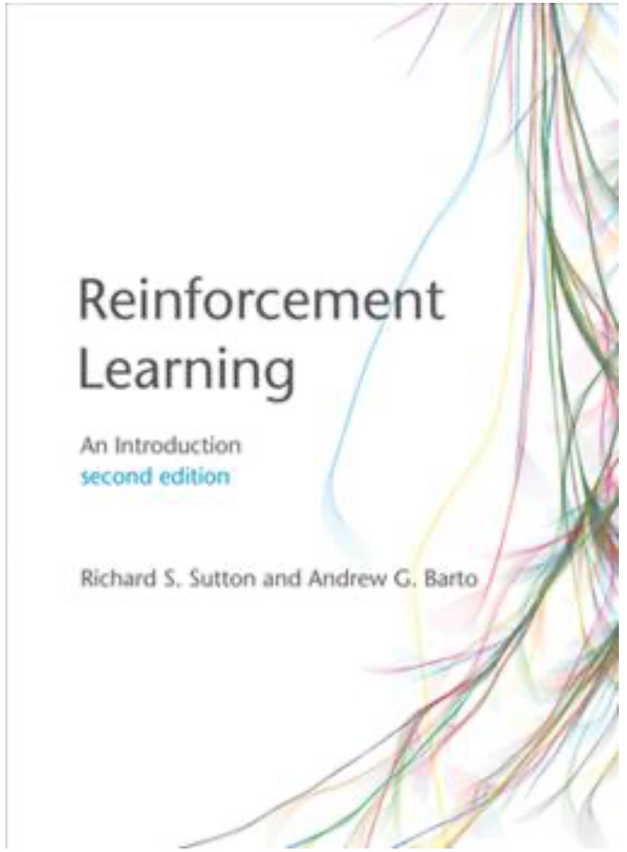
Deep Reinforcement Learning

https://bit.ly/rlairsim

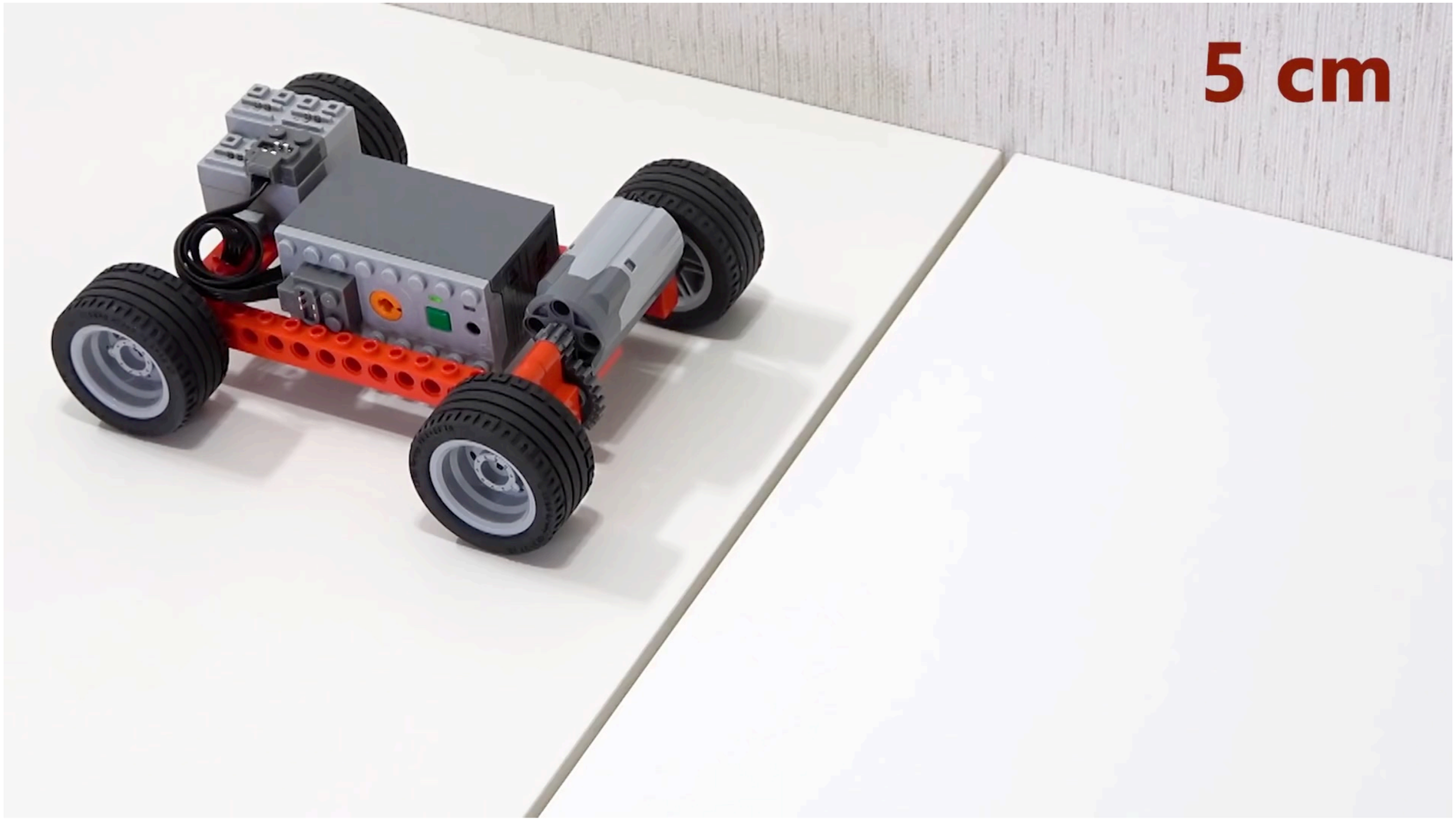


Richard Sutton
“Father of Reinforcement Learning”

University of Alberta; Deep Mind



Developing an AI System Is Like...



https://bit.ly/makelegocar

Human-AI Interaction

How Will AI Change Work (Knickrehm 2019)

Five Schools of Thought

- **Dystopians:** Man and machine will wage a Darwinian struggle that machine will win. Expect massive unemployment, falling wages, and wrenching economic dislocation
- **Utopians:** Intelligent machines will take on even more work, but the result will be unprecedented wealth, not economic decline. AI and computing power will advance to achieve the “singularity”
- **Technology optimists:** AI will result in a leap in productivity and create economic growth and improvements in living standards
- **Productivity skeptics:** AI will result in little productivity increase and more income inequality
- **Optimistic realists:** AI will advance productivity in certain companies and sectors. It requires better human-AI interaction to realize the potential of AI

“It's Easier to Invent the Future Than to Predict It”

- Ultimately, business leaders influence the future of work, so it is decisions rather than predictions that matter more

- Actions that can shape the future
  - Use technology to augment human skills and reinvent operating models
  - Refine jobs and rethink organizational design
  - Unleash human talents that machines cannot match: creativity, empathy, communications, adaptability, and problem solving

The Case of Stitch Fix

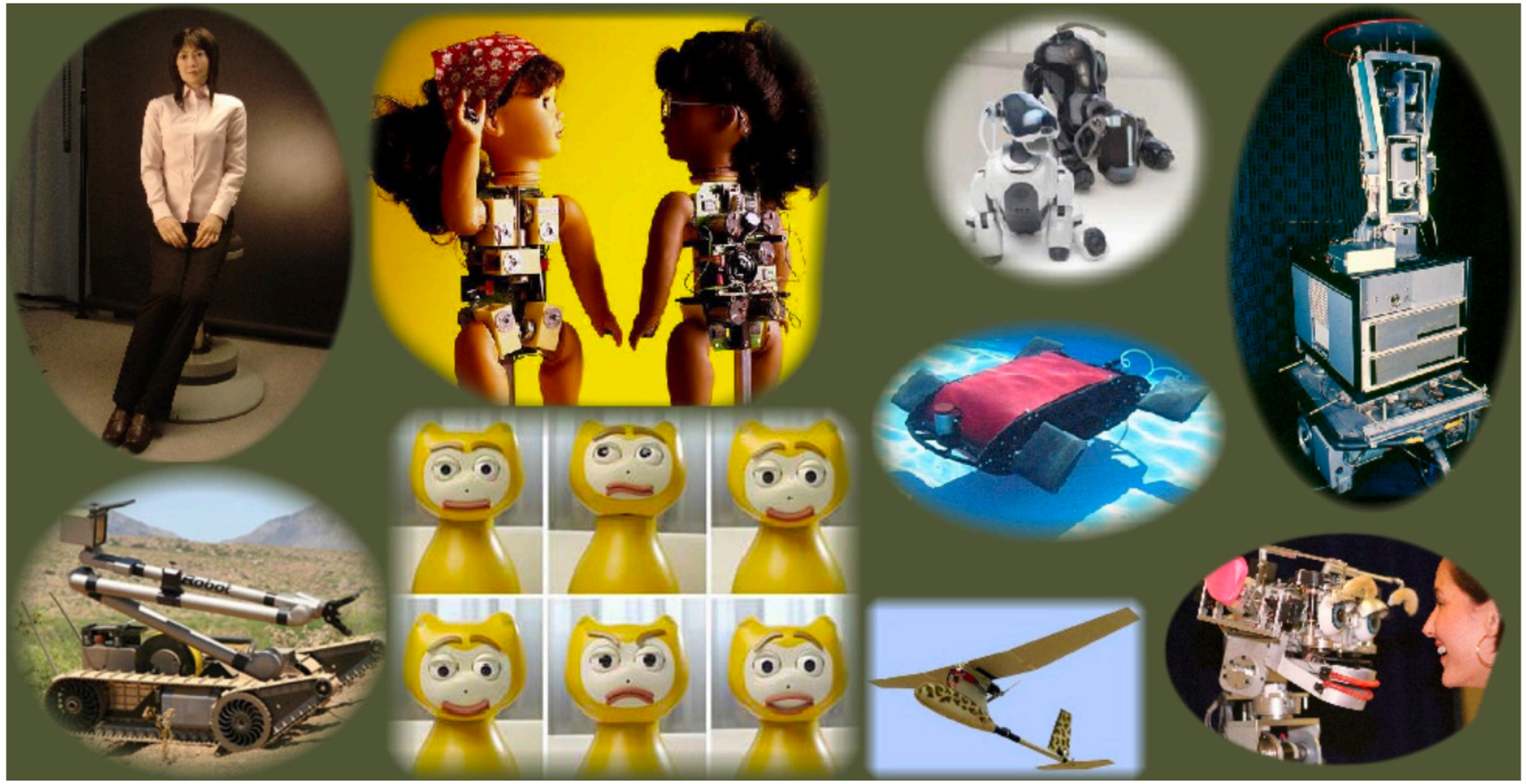
100+ Data Scientists and 5,000+ Human Stylists

More details about human-AI interaction at Stitch Fix: https://bit.ly/sfhai

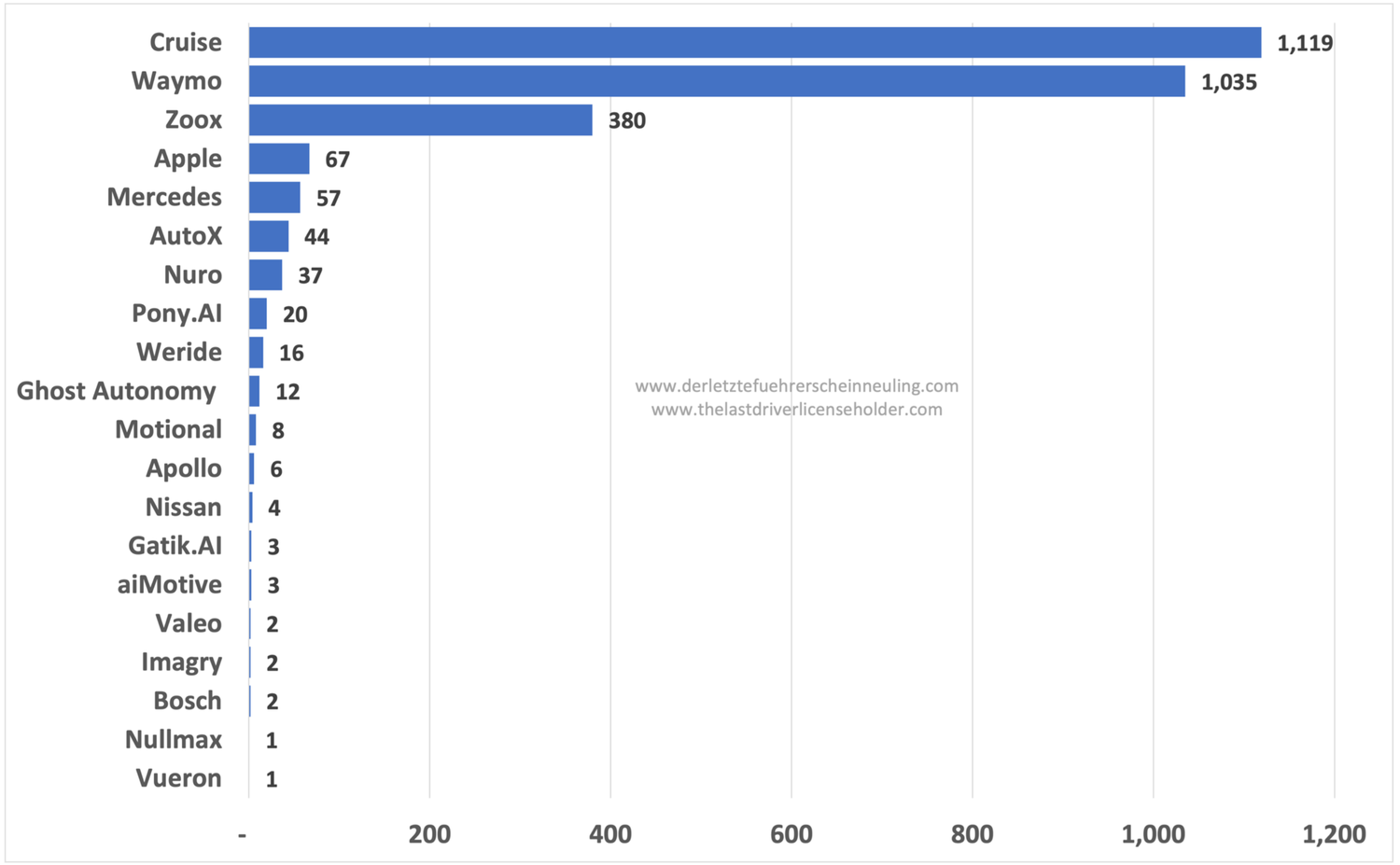
AI Poses Real Risks to Human Beings

- People might lose their jobs to automation
- People might work even harder as AI increases the pace of technological innovations
- People might lose their sense of being unique
- The use of AI may result in a loss of accountability

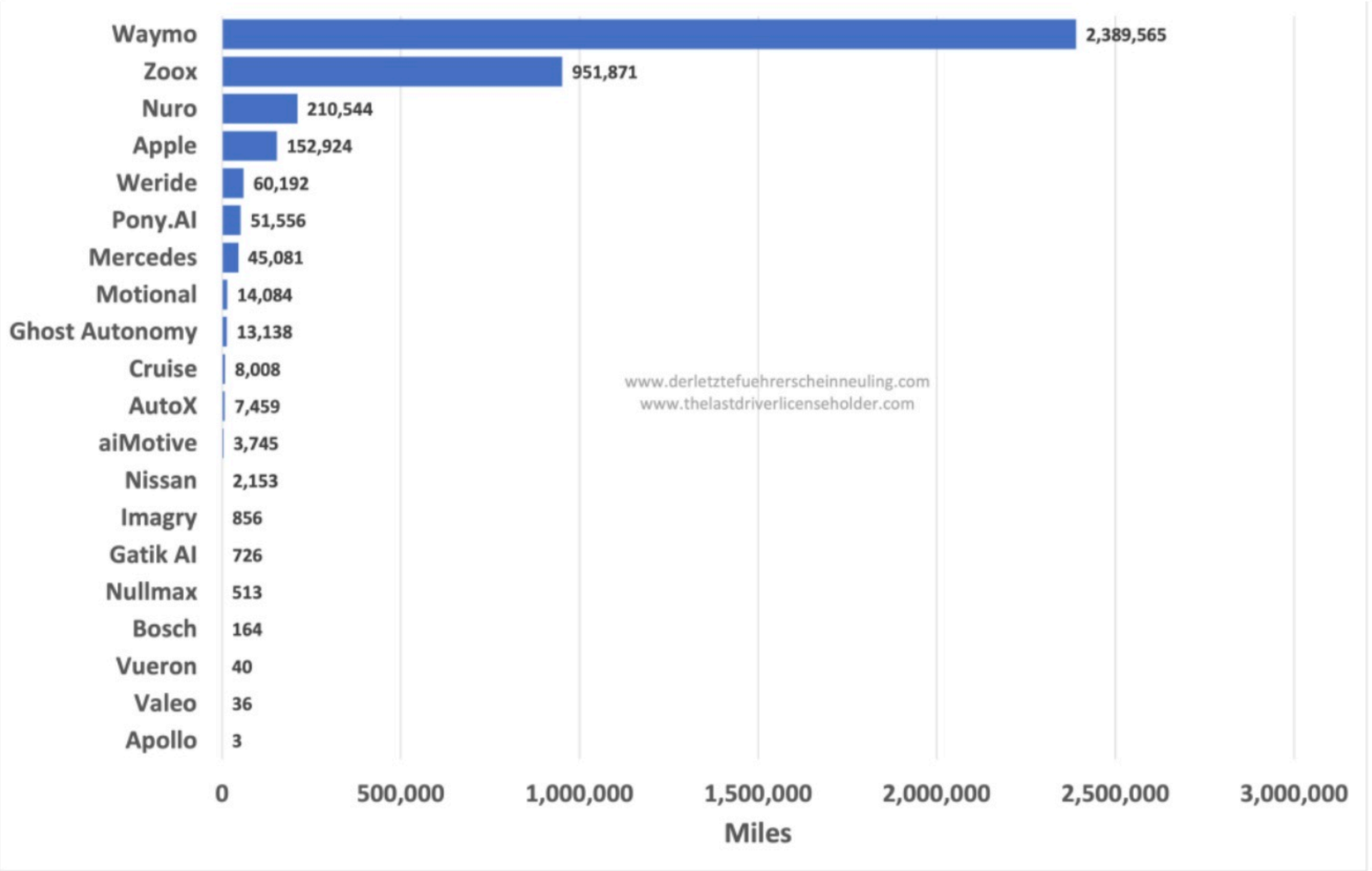
Human-Robot Interaction



Number of autonomous cars in California (Dec. 2023 – Nov. 2024)



Miles Driven by Autonomous Cars in California (Dec. 2023 – Nov. 2024)



REMOTE OPERATORS

The New York Times

TIMES INSIDER

When Self-Driving Cars Don't Actually Drive Themselves

An immersive article shows readers what a New York Times reporter has tracked for nearly a decade: Robot taxis still need human help.

Listen to this article · 5:46 min Learn more

Share full article

The New York Times reporter Cade Metz riding in a Zoox robot taxi in Foster City, Calif. Jason Henry for The New York Times

"Although today's robot taxis do not have drivers behind the steering wheels — some don't even have steering wheels — they still lean on the good sense of people like you and me. All robot taxi companies operate command centers like the one I visited in Foster City…"

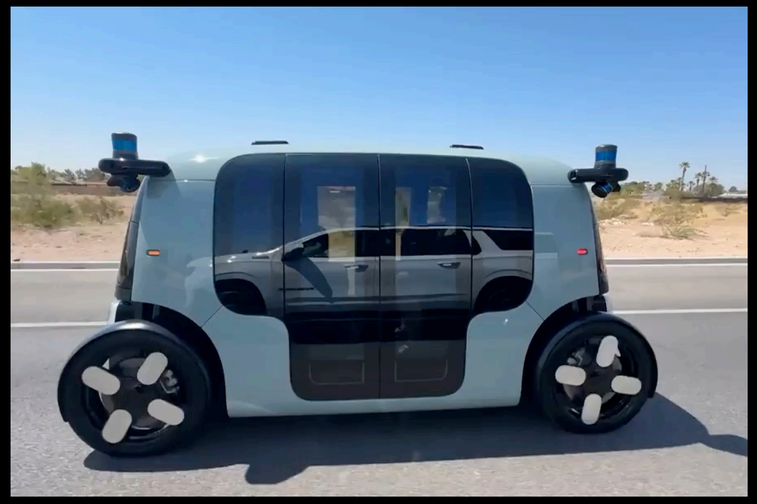
Cade Metz, *New York Times*, 9/11/2024

REMOTE OPERATORS

The New York Times

How Self-Driving Cars Get Help From Humans Hundreds of Miles Away

By Cade Metz, Jason Henry, Ben Laffin, Rebecca Lieberman and Yiwen Lu Sept. 3, 2024



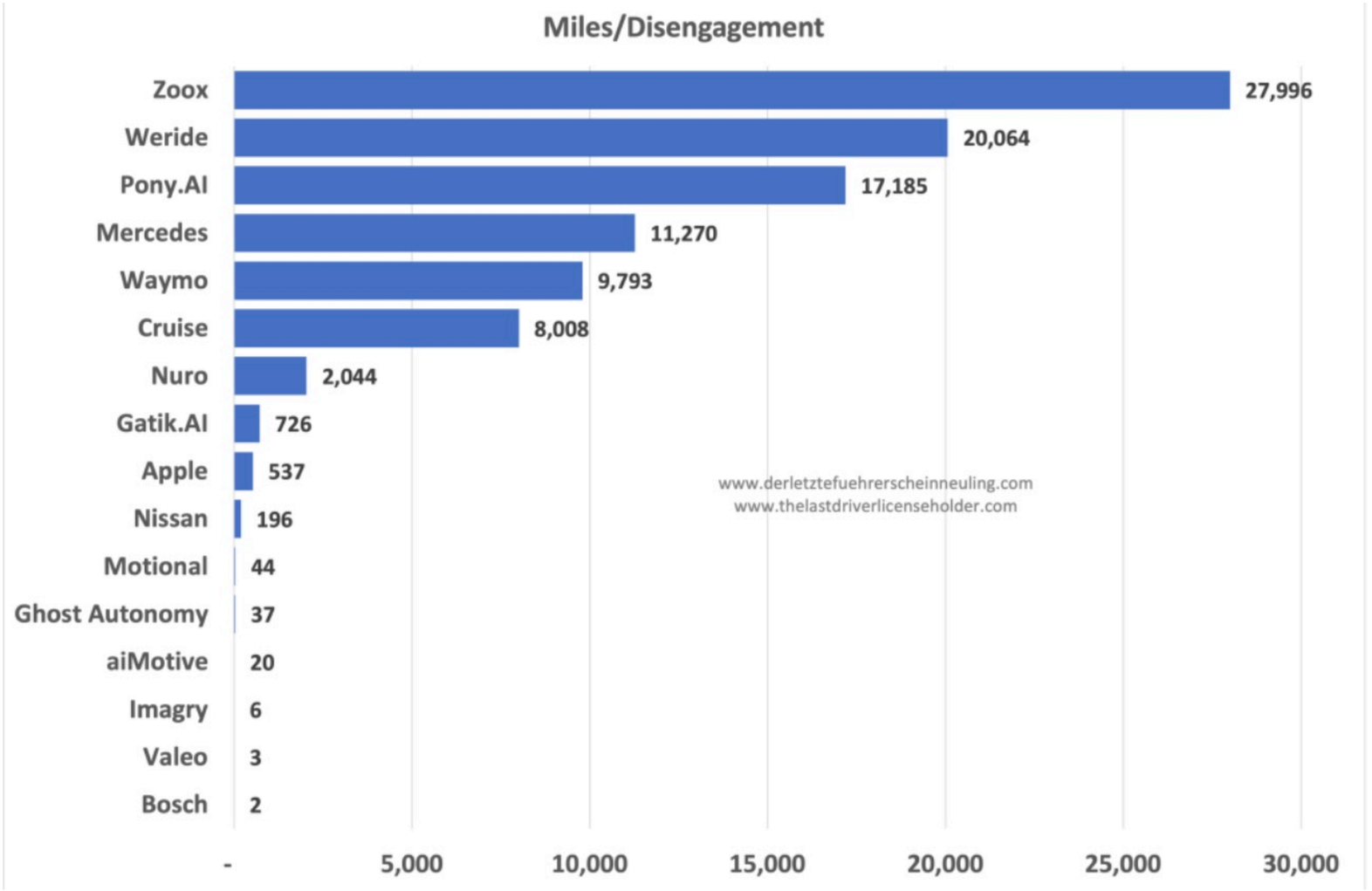
https://www.nytimes.com/interactive/2024/09/03/technology/zoox-self-driving-cars-remote-control.html

Since 2014, California has required all self-driving car testers to file annual disengagement reports

The idea: transparency and accountability

The reality: firms hate it, complaining it dampens innovation

Miles Driven per Disengagement in California (Dec. 2023 – Nov. 2024)



The New York Times

G.M.'s Cruise Moved Fast in the Driverless Race. It Got Ugly.

Cruise has hired a law firm to investigate how it responded to regulators, as its cars sit idle and questions grow about its C.E.O.'s expansion plans.

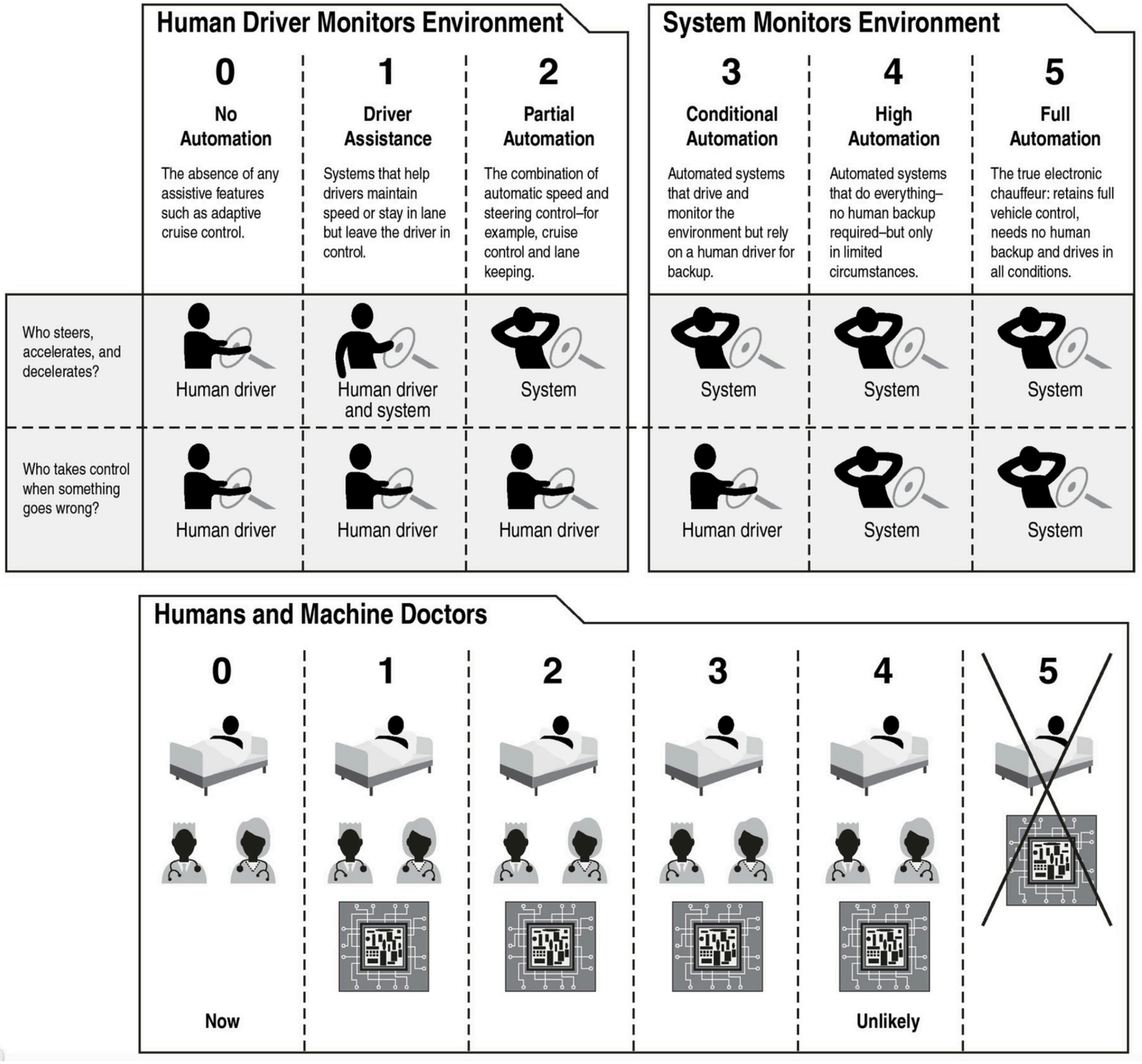
Half of Cruise's 400 cars were in San Francisco when the driverless operations were stopped. Those vehicles were supported by a vast operations staff, with 1.5 workers per vehicle. The workers intervened to assist the company's vehicles every 2.5 to five miles, according to two people familiar with is operations. In other words, they frequently had to do something to remotely control a car after receiving a cellular signal that it was having problems.

To cover its spiraling costs, G.M. will need to inject or raise more funds for the business, said Chris McNally, a financial analyst at Evercore ISI. During a call with analysts in late October, Ms. Barra said G.M. would share its funding plans before the end of the year.

https://bit.ly/nytcruise23

Human-Robot Interaction: Autonomy

- The most important measure of autonomy for human-robot interaction is the amount of time that a robot can be neglected, that is, the **neglect tolerance** of the robot
- Related measure: **interaction time** = the average amount of time it takes for human operators to interact with robot
- A system with a high level of autonomy is one that can be neglected for a long period of time without interaction
- Another measure: **level of autonomy** (LOA)



Source: Eric Topol, *Deep Medicine*

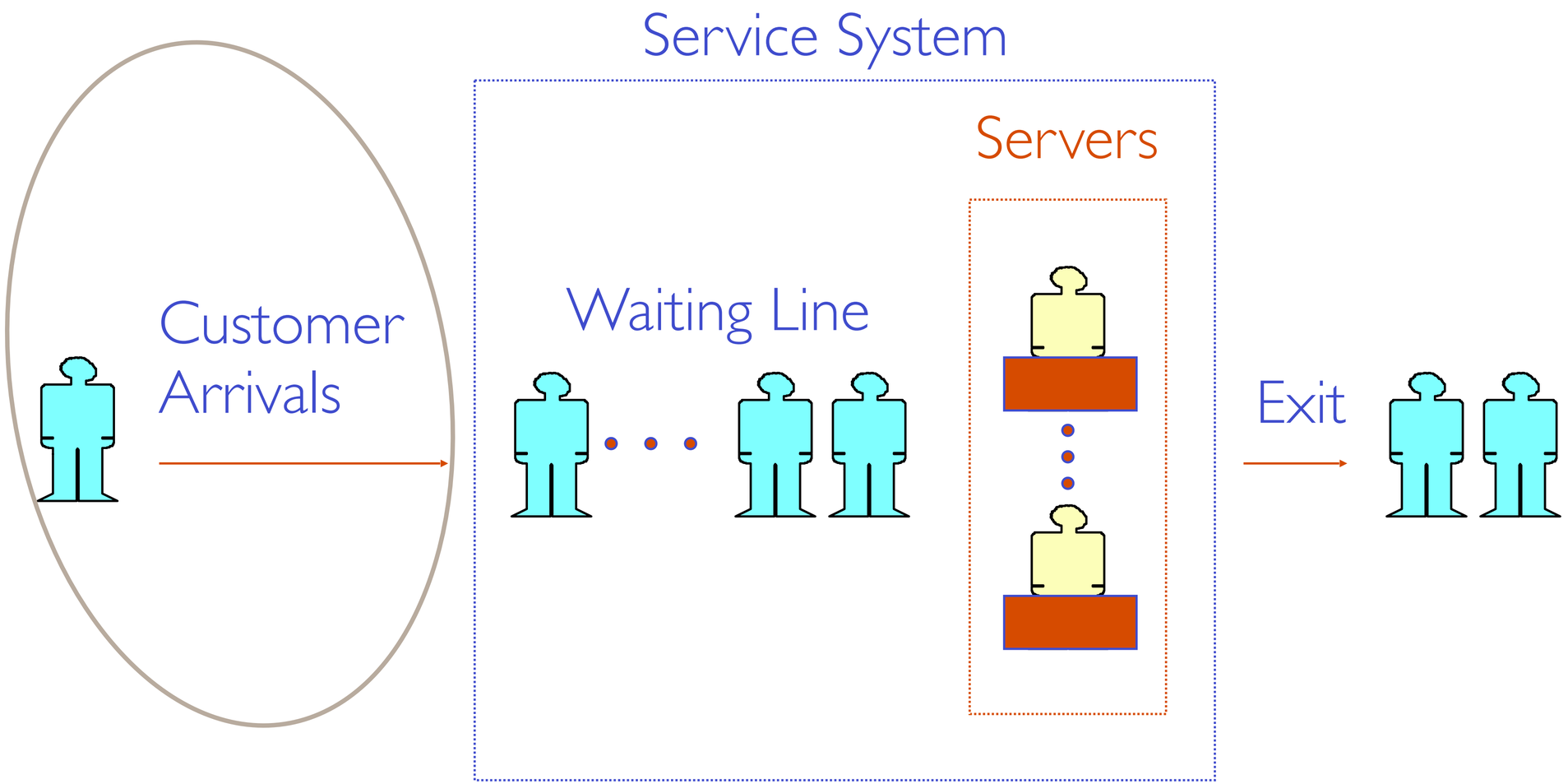
Human-Robot Interaction: Teams

- Another important question in human-robot interaction: How many remote robots a single human can manage?

- **Fan-out**: an upper bound on the number of independent, homogeneous robots that a single person can manage
  - A robot team's fan-out depends on neglect tolerance and interaction time

- A more practical question: How many humans does it take to efficiently manage a fixed number of robots, allowing for the possibility of adaptable autonomy and dynamic handoffs between humans?

Components of a Queueing System



Queuing Theory

- a = average inter-arrival time (**neglect tolerance**)
- p = average processing/service time (**interaction time**)
- m = number of servers (**number of human operators**)

- u = system utilization

  = arrival rate / aggregate service rate

  = (1/a) / m·(1/p)

  = p / m·a

Applying Queuing Theory to Analyze Human-Robot Interaction

- u = system utilization

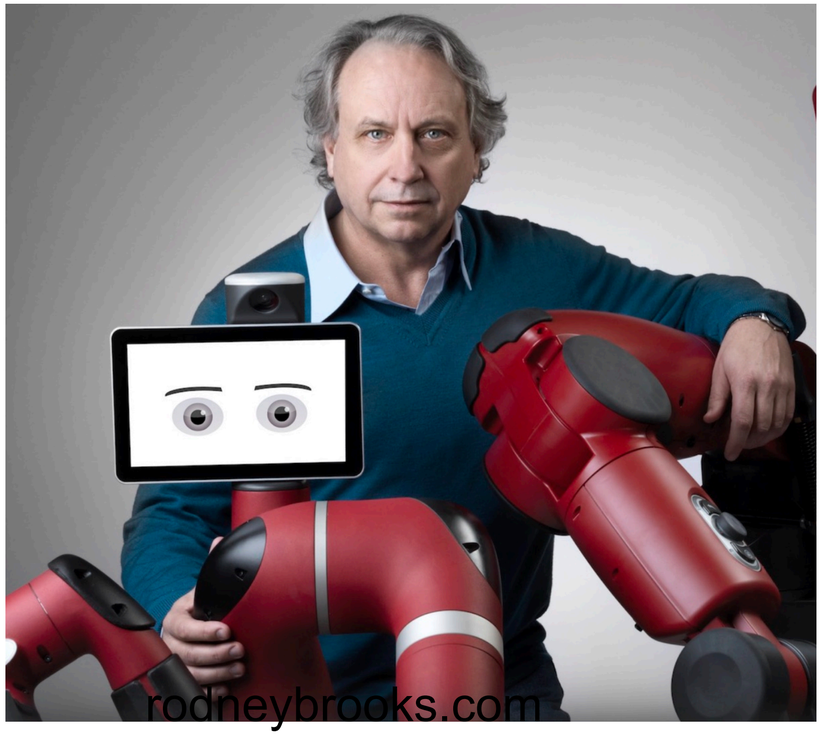
  = arrival rate / aggregate service rate

  = (1/a) / m·(1/p)

  = p / (m·a)

We can use the maximum utilization rate to determine the number of human operators needed to maintain a robot team

RODNEY BROOKS' THREE LAWS OF AI



1. When AI performs a task, human observers immediately estimate its general competence in areas that seem related. Usually that estimate is wildly overinflated

2. Most successful AI deployments have a human in the loop and their intelligence smooths the edges

3. Without carefully boxing in how AI is deployed there is always a long tail of special cases that take decades to discover and fix. Paradoxically all those fixes are AI-complete themselves


We are Doomed...

Three Laws of Robotics (Isaac Asimov 1942)

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm

2. A robot must obey orders given to it by human beings, except where such orders would conflict with the First Law

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law

Three Laws of Robotics (Isaac Asimov 1942)

The success of AI might mean the end of the human race....

First, the AI system's state estimation may be incorrect, causing it to do the wrong thing.

Second, specifying the right utility function for an AI system to maximize is not easy.

Third, the AI system's learning function may cause it to evolve into a system with unintended behavior.

Russell and Norvig (2015)

4th TA Tutorial: Creating Agentic AI*

***Based on Suhas' Real-World Expereince at Amazon**

🗓️ Friday, 12/5, 12:00–1:00 PM

📍 Join via Zoom: https://bit.ly/jhuaita25

Attendance is optional. Materials are on Canvas

Until Next Class

- AI Lab due on December 12th (Friday) noon; see Canvas for detailed instructions

- Refer to the syllabus for Readings for Session 7


Thank You!