

# Machine Learning – Products

# Most Famous Applications

recommendations



chatbots



autonomous driving

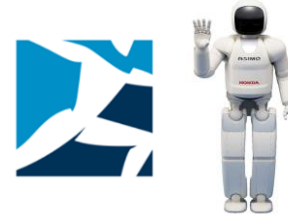


MOTOR Ai

translation



robotics



assistants (speech recognition)



OCR

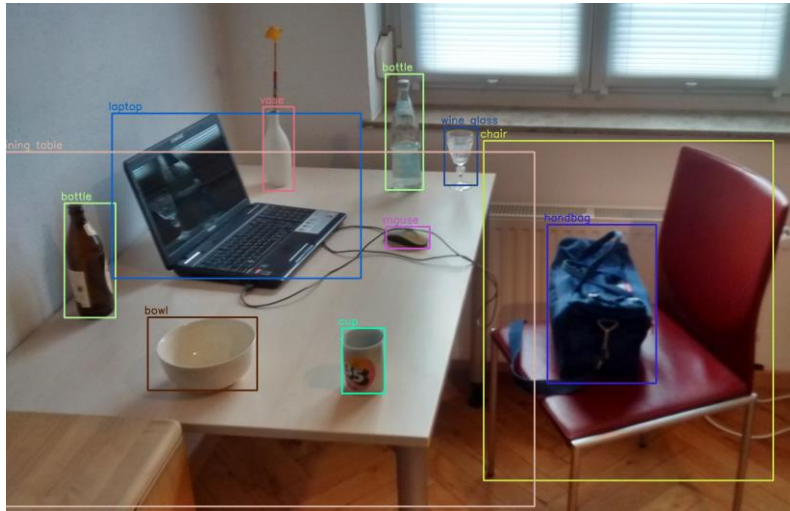


and many more ...

# When to Use ML (= Learning from Data)

## automation

too complex for rules

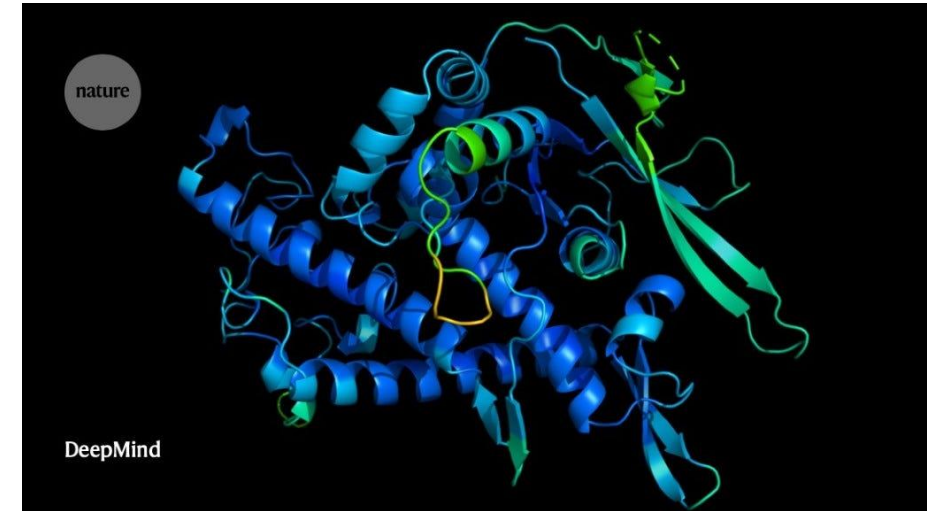


from wikipedia

examples: object recognition, all applications from previous slide

## complexity / uncertainty

too complex for humans



examples: protein structure predictions (AlphaFold), demand forecasting

more scientific use cases: medicine (imaging, diagnosis, drug design), particle physics (analysis of collider experiments), material science (material properties and design of new materials), ...

# Taxonomy of ML Models

# Supervised Learning

## Target Quantity

- **known in training:** labeled samples or observations from past
- to be **predicted** for unknown cases (e.g., future values)

## Features

- input information that is
- correlated to target quantity
  - known at prediction time



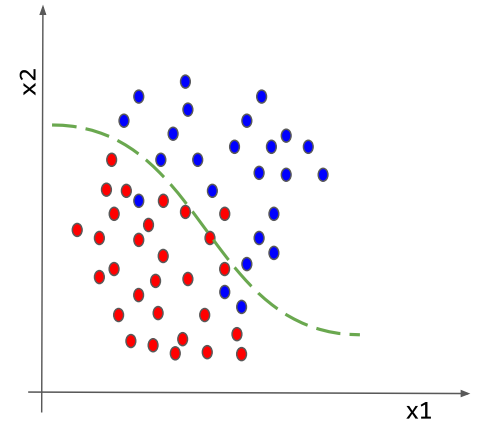
## Example: Spam Filtering

*Classify emails as spam or no spam*

use accordingly **labeled**  
**emails as training set**

use information like  
**occurrence of specific**  
**words or email length**  
as **features**

**features  $x_1$  and  $x_2$**   
**spam, no spam**

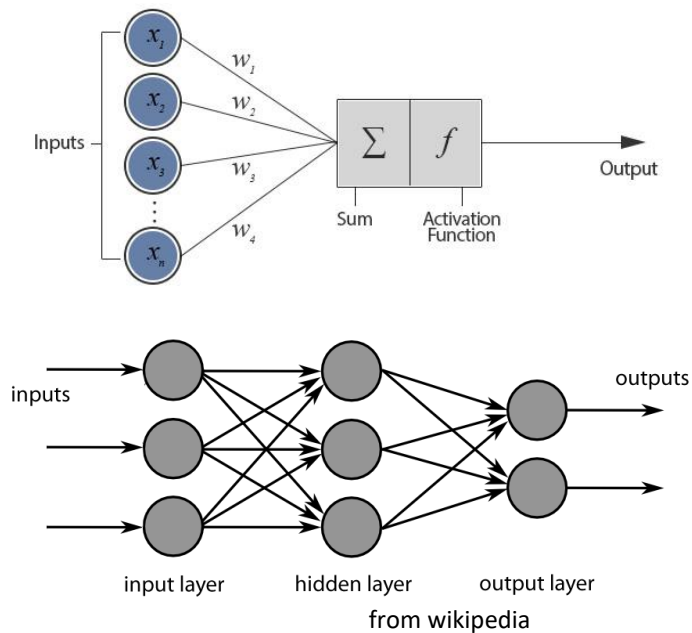


usually rather narrow tasks

# Algorithmic Families

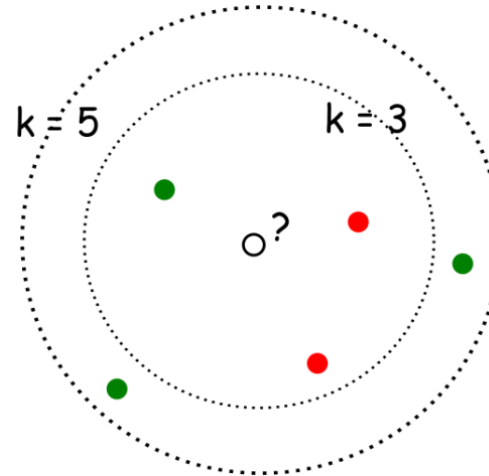
## linear (parametric) models

**neural networks:** non-linear just by means of activation functions



**deep learning:** many hidden layers  
computer vision: CNN  
NLP: transformer

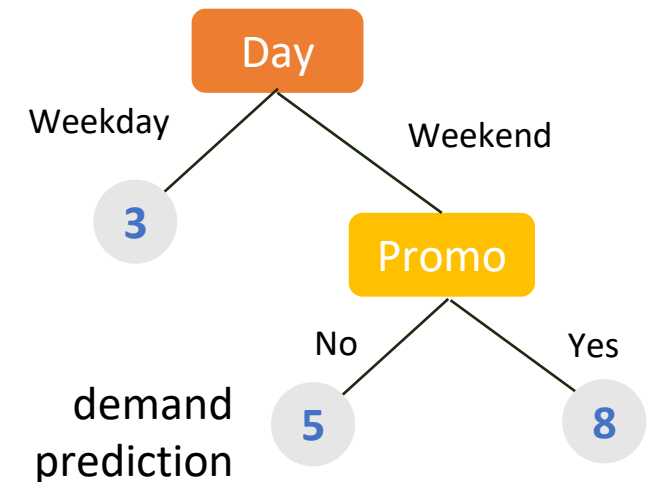
## nearest neighbors (local methods, instance-based learning) – non-parametric models



with  $k=3$ , ●  
with  $k=5$ , ●

**kernel/support-vector machines:** linear model (maximum-margin hyperplane) with kernel trick

## decision trees: rule learning



**often used in ensemble methods**

- bagging: random forests
- boosting: gradient boosting

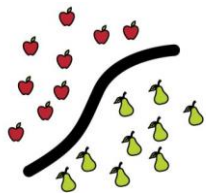
mainly used for structured data

# MACHINE LEARNING

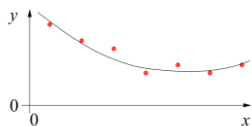
training target available  
(labeled or past data)

## SUPERVISED

### CLASSIFICATION



### REGRESSION

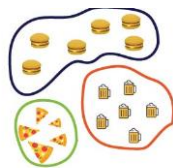


learning by teacher  
(high-dimensional curve fitting)

data not labeled  
in any way

## UNSUPERVISED

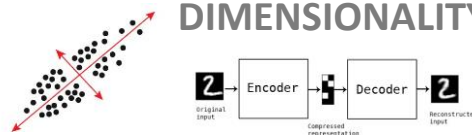
### CLUSTERING



### ASSOCIATION



### DIMENSIONALITY REDUCTION



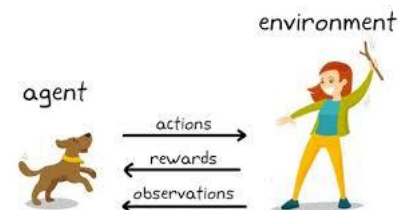
learning by observation  
(pattern recognition)

no supervision, but goal-based  
interaction with environment

## REINFORCEMENT LEARNING

### LEARN STATE OR ACTION VALUES

### LEARN POLICY DIRECTLY



learning by trial-and-error  
(sequential decision making)

unsupervised and reinforcement learning can  
both be cast as supervised-learning setup

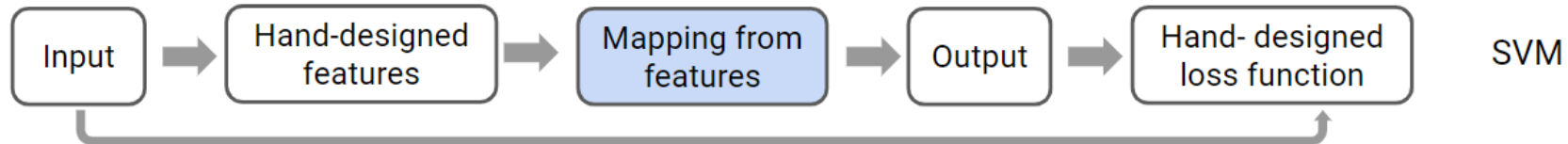
# Ladder of Generalization

## Rule-based systems

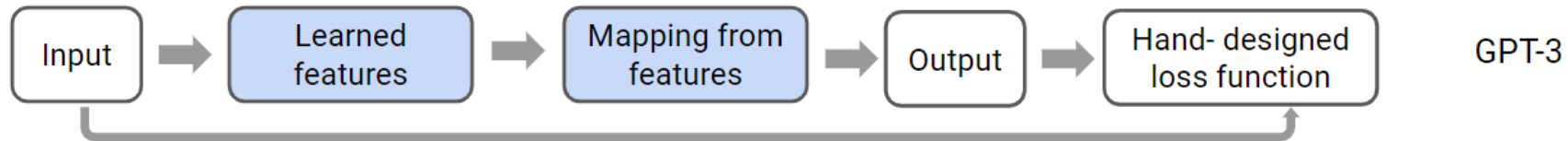


Learnable part of the system

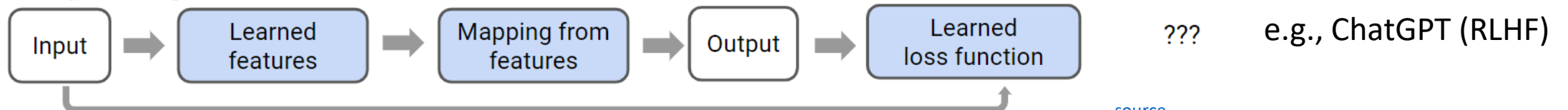
## Classical machine learning



## Deep learning: (self-)supervised learning



## Deep learning: other RL formulations



[source](#)



# Discriminative vs Generative Models

discriminative models:

prediction/estimation of labels (classification)  
or numerical values (regression)

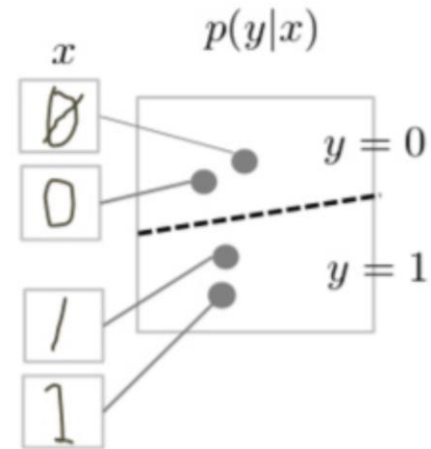
examples for discriminative tasks:

- object recognition
- demand forecasting

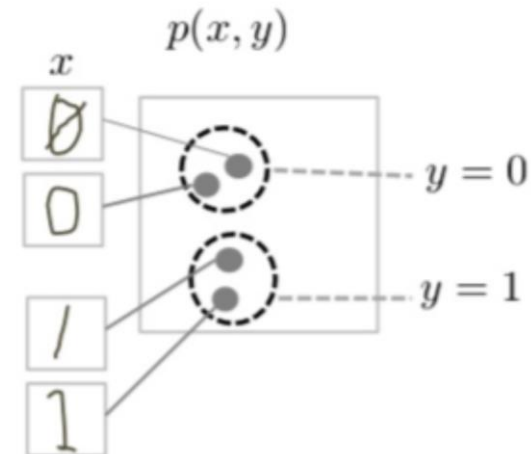
generative models:

generation of new data according to data  
distribution seen in training

discriminative model



generative model

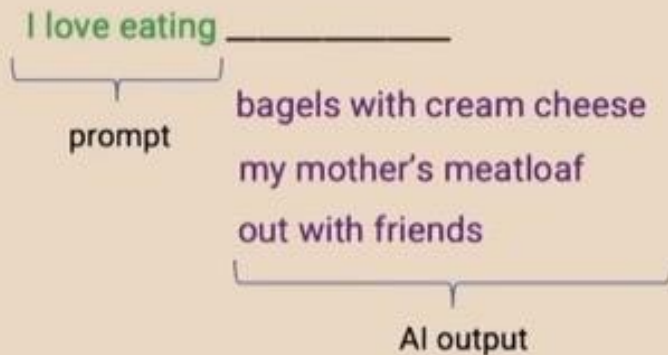


[source](#)

# Text Generation

## This decade: Generative AI

### Text generation process



### How it works

Generative AI is built by using supervised learning ( $A \rightarrow B$ ) to repeatedly predict the next word.

*My favorite food is a bagel with cream cheese and lox.*

Input (A)	Output (B)
My favorite food is a	bagel
My favorite food is a bagel	with
My favorite food is a bagel with	cream

When we train a very large AI system on a lot of data (hundreds of billions of words) we get a Large Language Model like ChatGPT.

Stanford

Andrew Ng



# Large Language Models (LLM)

special class of generative models

new paradigm for ML workflow:

**prompt engineering** (low/no code)

instead of classic training (fit) and inference (predict) steps

→ multi-task (and multi-modal) models

instead of narrow use cases of discriminative models

# Under The Hood: Foundation Models

LLMs: transformer models with hundreds of billions of parameters

self-supervised (pre-)training on vast data sets

→ huge foundation models (impossible to train yourself)

usage options:

- typical: **in-context** learning via prompt (potentially giving few examples for task at hand and add retrieval augmentation or tool usage)
- for special case and high-quality requirements: **fine-tuning** on specific tasks and data sets (example: chatbot like ChatGPT via reinforcement learning from human feedback)

# Generative AI

Depending on the application, there are currently two dominant approaches for generative AI:

- text generation: LLMs
- image synthesis: diffusion models

# Image Synthesis

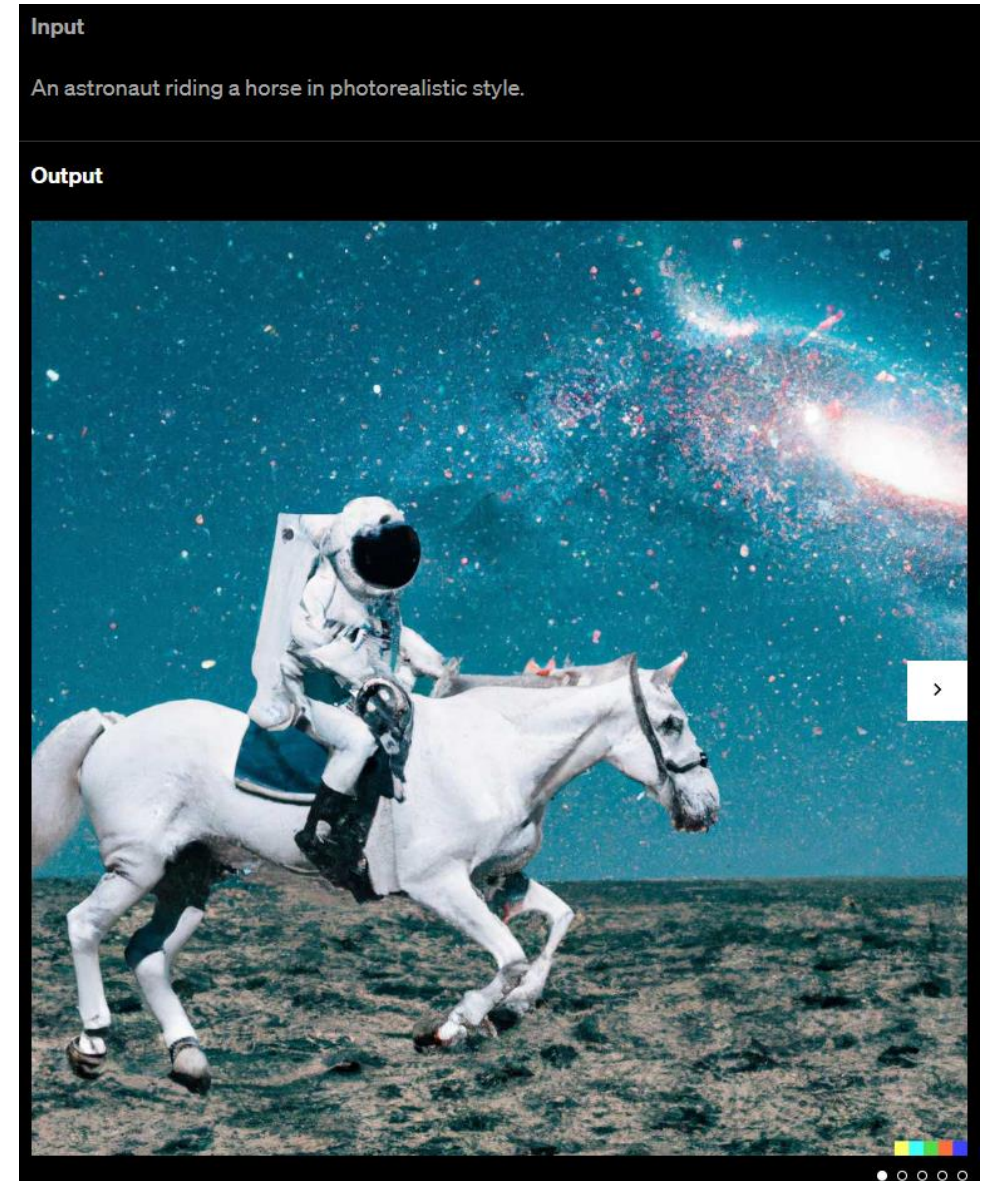
idea: generate new images as variations of training data

condition generation on text prompts:  
text-to-image

trade-off between diversity and fidelity

SOTA: (guided) diffusion models

example: DALL-E 2



# Application

## **discriminative models**

- effective for performing numerical and optimization tasks (predictions)
- continue to account for majority of AI value in wide range of industries (e.g., supply chain)  
→ LLM agents might use prediction or optimization models as tools

## **generative models**

- not suitable for classical use cases like numerical and optimization tasks
- but complimentary: drive value across entire organizations by revolutionizing internal knowledge management systems

# Application Areas



# Examples for Predictive Models: Supply Chain

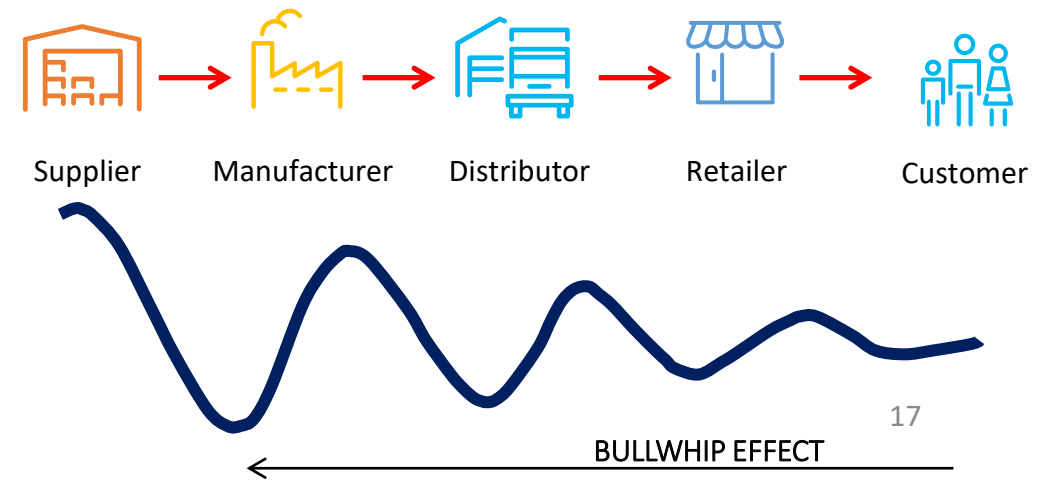
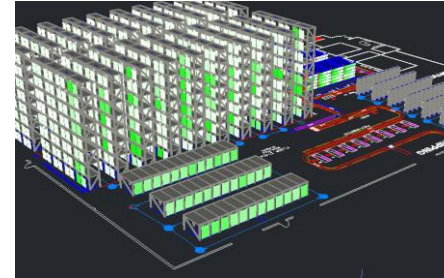
support of operations research

some examples:

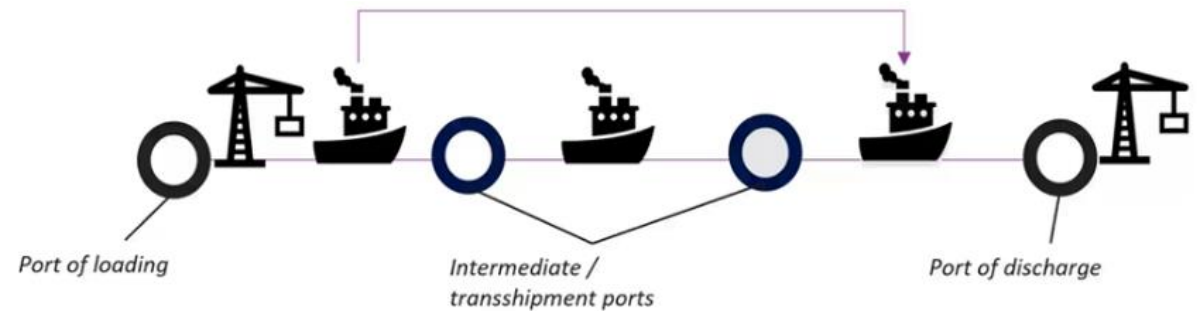
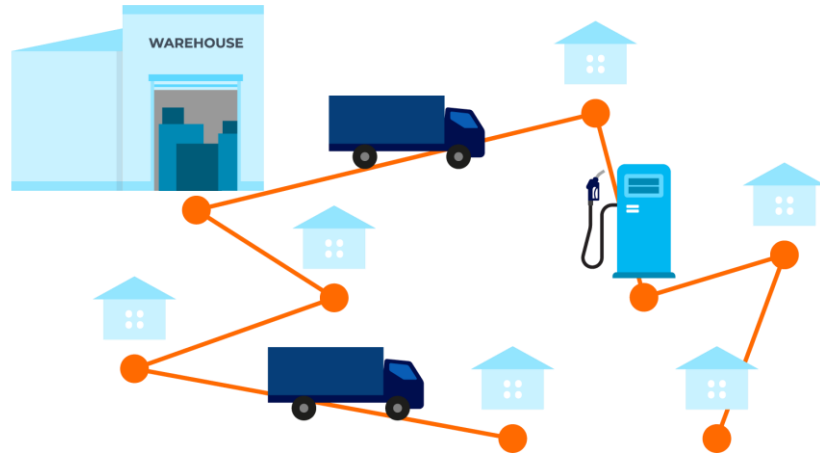
- warehouse operations
- transportation (logistics & mobility)
- retail

(Blue Yonder: demand forecasting, replenishment, pricing, targeting)

slotting:  
e.g., using item affinity  
and order forecasting



# Connected Logistics

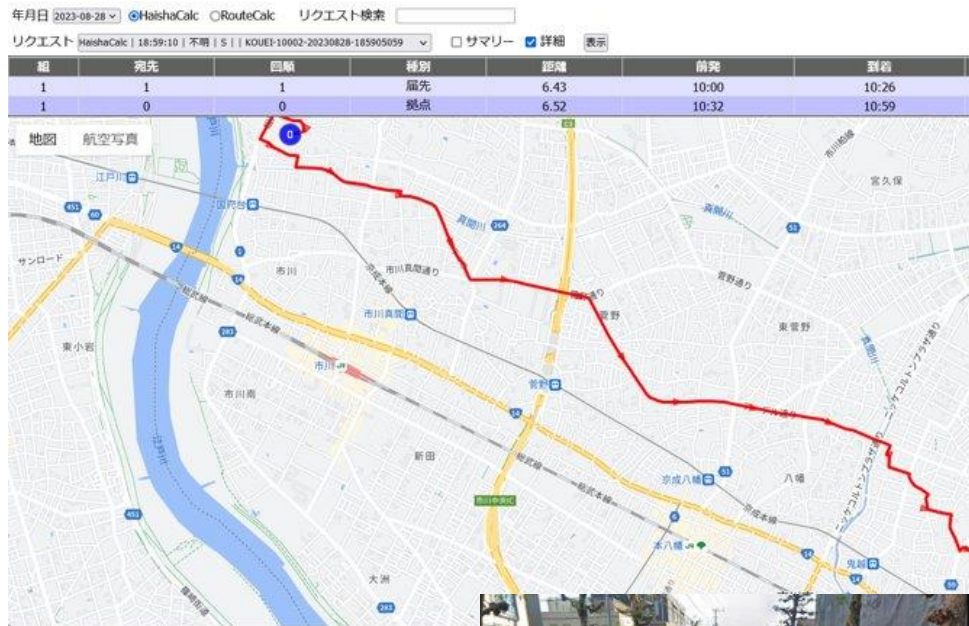


examples for ML solutions:

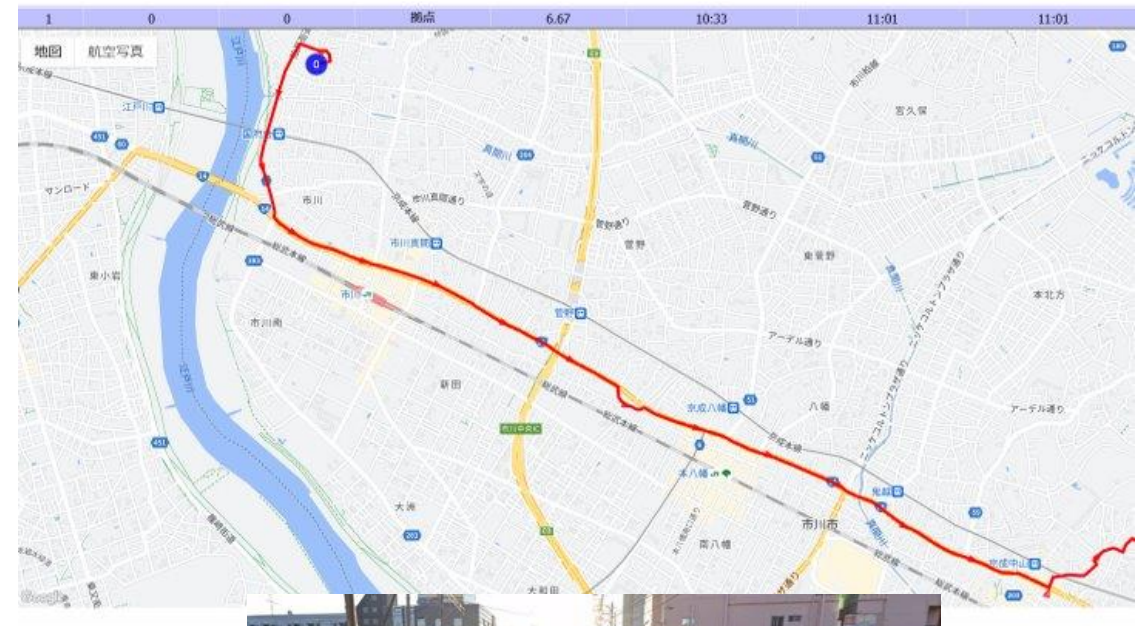
- arrival time predictions (aka ETA), for example in Google Maps via Graph Neural Network (GNN)
- reinforcement learning (combined with other methods like GNN) for better generalization in combinatorial optimization problems (e.g., TSP, VRP, ...)
- disruption predictions (also root cause analysis)

# Routing: Learning from Skilled Drivers

shortest route:



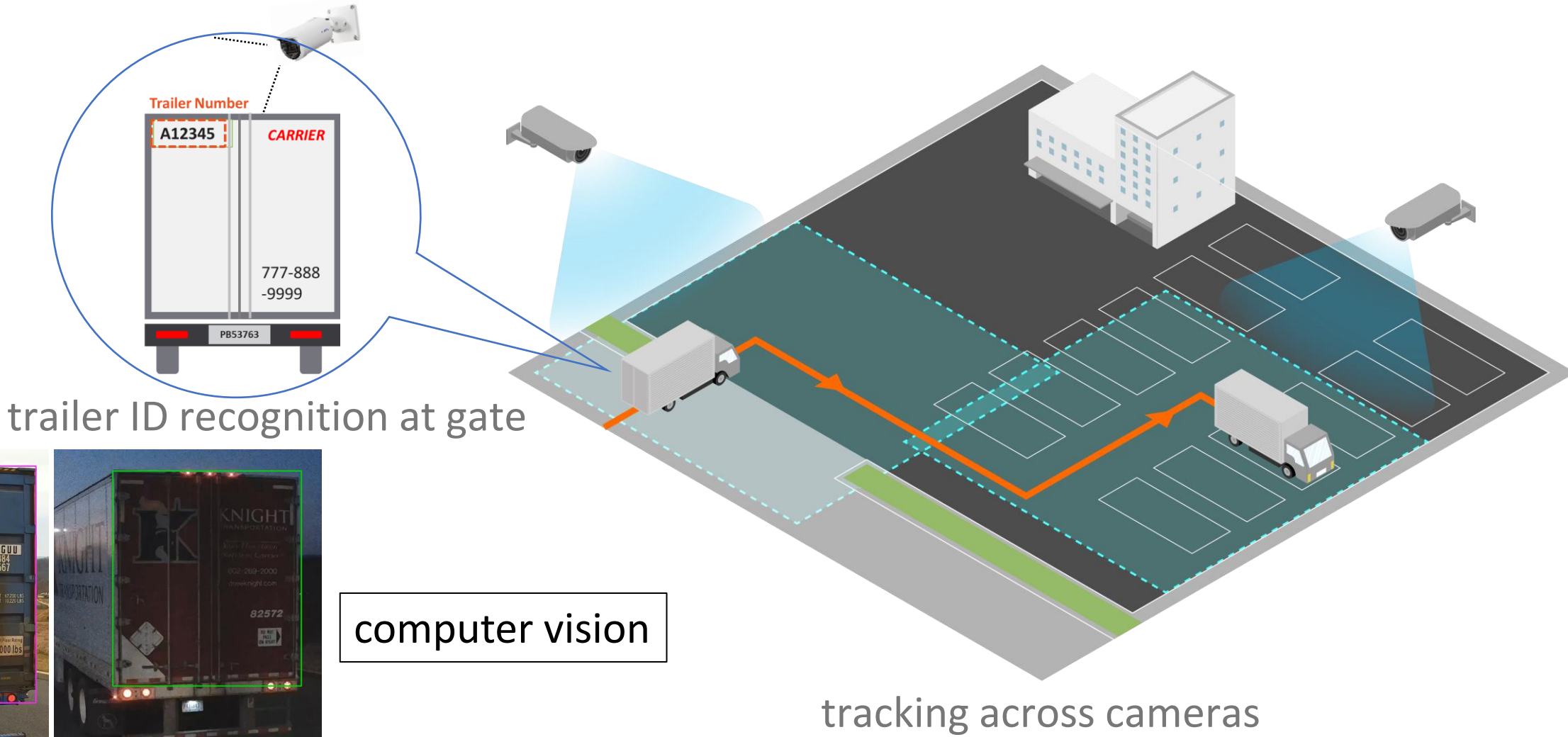
better route:



supervised learning



# Tracking for Yard Management

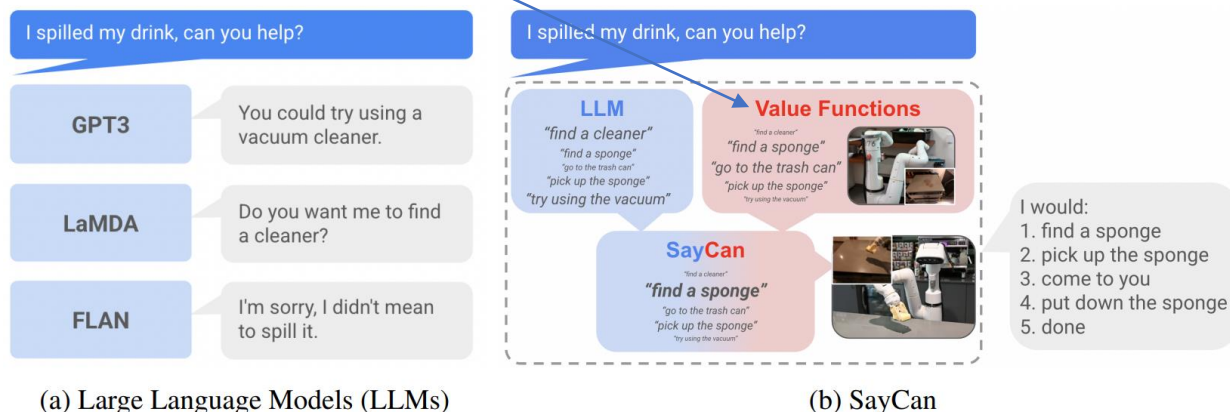


# Robotic Control

different ways:

- reactive control (no ML)
- model predictive control
- imitation learning
- reinforcement learning
- using LLMs (and vision)

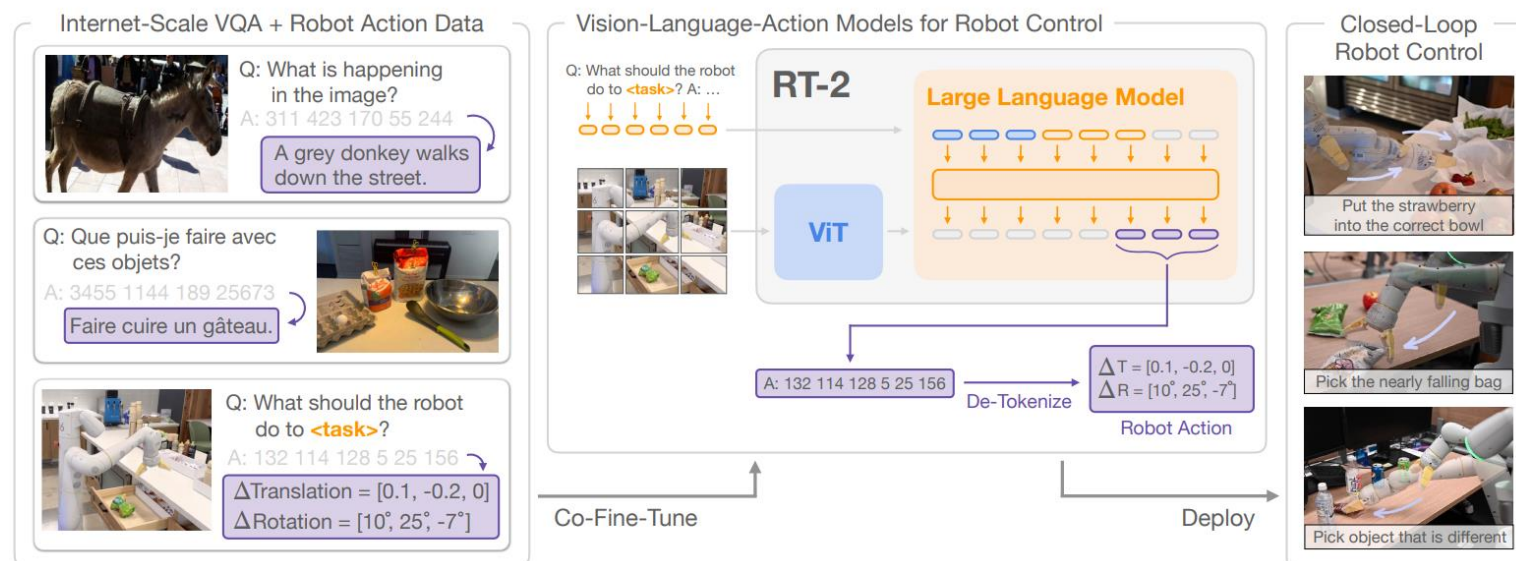
grounding with pre-trained skills ([SayCan](#)):



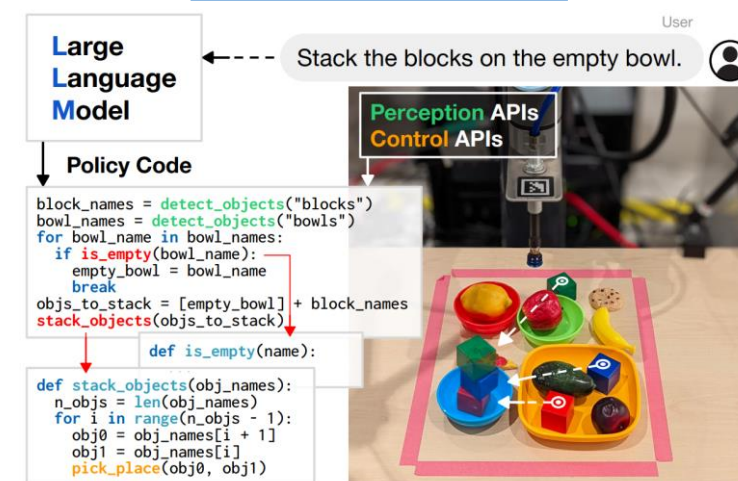
(a) Large Language Models (LLMs)

(b) SayCan

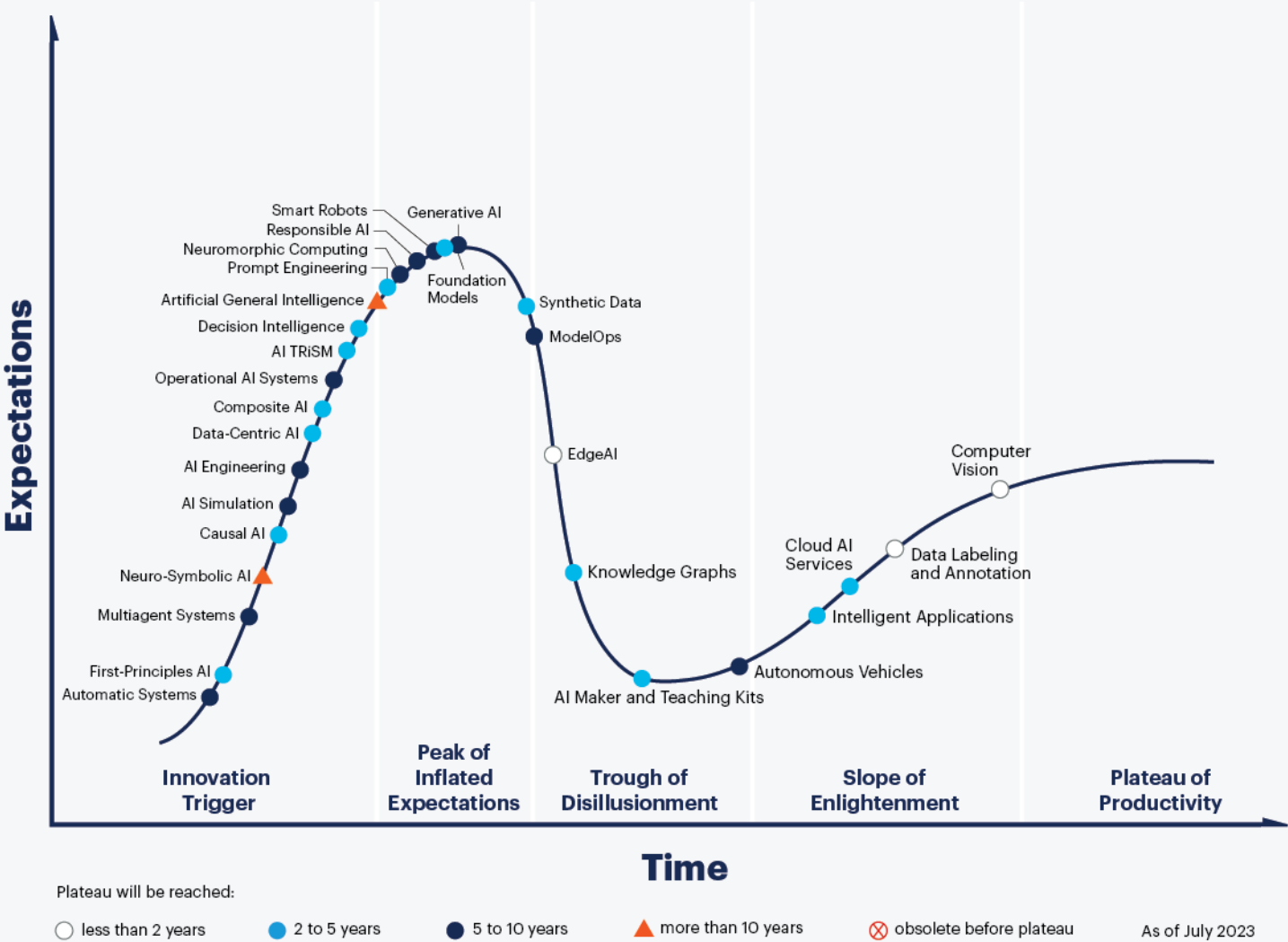
generalization with pre-trained vision-language models ([RT-2](#)):



Code as Policies:



# Hype Cycle for Artificial Intelligence, 2023



Generative AI at peak of inflated expectations

still: there is now something to play with

# Biggest Business Impacts of Generative AI

## customer operations

interactions with customers

## marketing & sales

generation of creative content

## software engineering

coding assistant

## product R&D

generative design (e.g., [for chips](#))

Figure 1. Magic Quadrant for Enterprise Conversational AI Platforms



# Coding Assistant

LLM: text-to-code

prominent example: GitHub Copilot

```
1 import tweepy, os # secrets in environment variables
2
3 def fetch_tweets_from_user(user_name):
4     # authentication
5     auth = tweepy.OAuthHandler(os.environ['TWITTER_KEY'], os.environ['TWITTER_SECRET'])
6     auth.set_access_token(os.environ['TWITTER_TOKEN'], os.environ['TWITTER_TOKEN_SECRET'])
7     api = tweepy.API(auth)
8
9     # fetch tweets
10    tweets = api.user_timeline(screen_name=user, count=200, include_rts=False)
11    return tweets
```

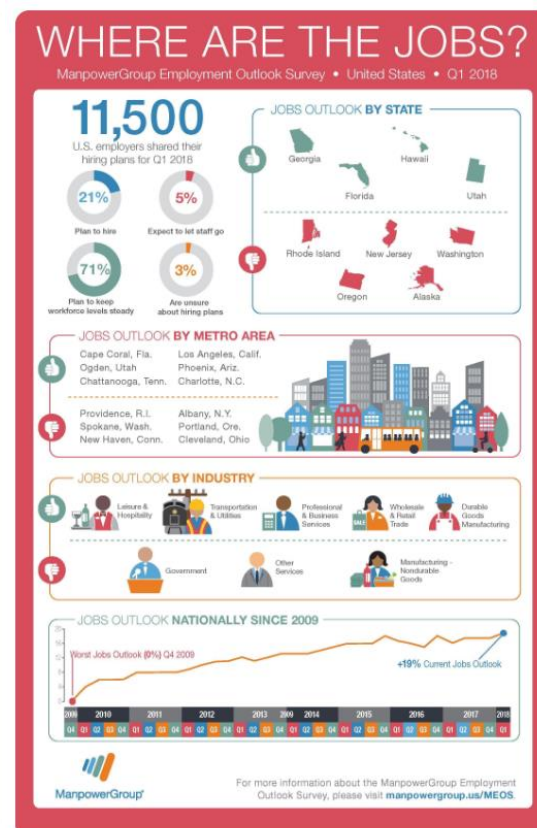
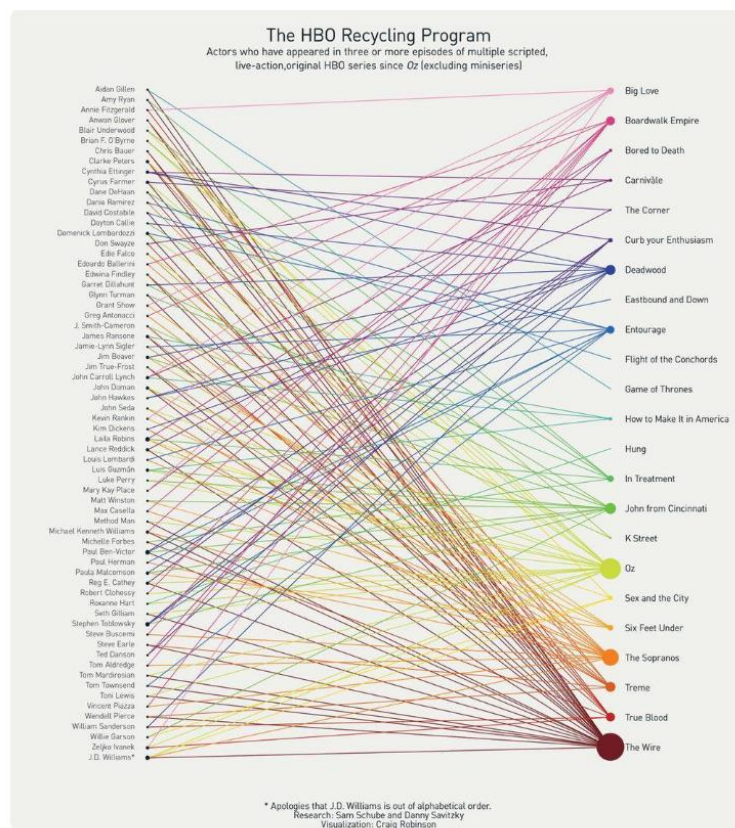


# Copilot for Knowledge Workers

multi-modal models (text and images)

enable image understanding

and execution of high-level  
user requests



Question: "Which is the metro in California that has a good job Outlook?"  
Fuyu's answer: "Los Angeles"

James Veel | Salesforce

adept2.lightning.force.com/lightning/r/Lead/00Q4x0000...

"Adept Client" started debugging this browser

Search...

Sales Task Call was created.

Lead James Veel

Lead Owner Augustus Ode

Lead Status New

Name James Veel

Company Costco

Title buyer

Phone

Email

Rating

Activity Chatter

Log a Call Email More

Recap your call... Add

Filters: All time • All activities • All types

Refresh Expand All View All

Upcoming & Overdue

No activities to show.

Get started by sending an email, scheduling a task, and more.

September • 2022 This Month

Call You logged a call

Call You logged a call

Question: "Aidan Gillen acted in how many series?"  
Fuyu's answer: "2"

[source](#)

[source](#)

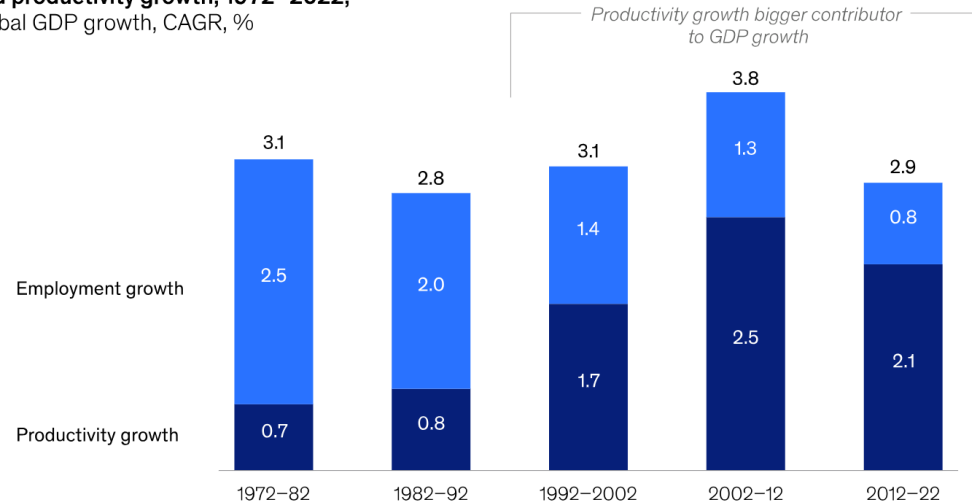
# AI Potential

*(generative) AI could free up 70% of employees' time*

*half of today's work activities could be automated between 2030 and 2060*

Productivity growth, the main engine of GDP growth over the past 30 years, slowed down in the past decade.

Real GDP growth contribution of employment and productivity growth, 1972–2022, global GDP growth, CAGR, %



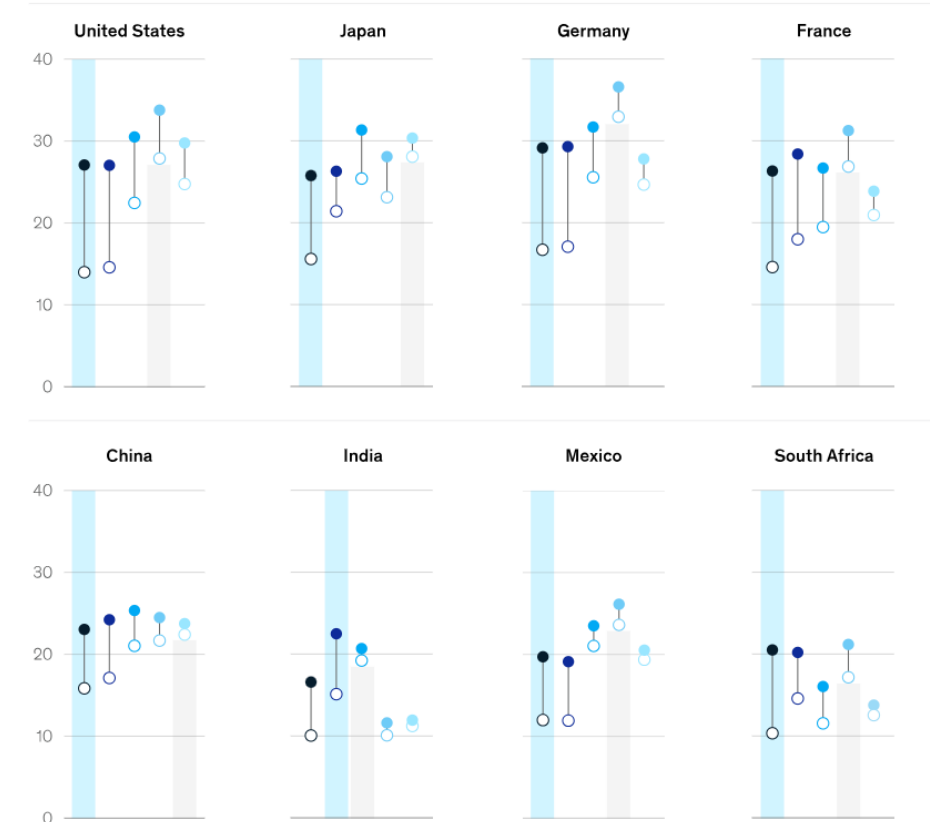
Source: Conference Board Total Economy database; McKinsey Global Institute analysis

*automate white-collar more than blue-collar jobs*

Generative AI could have the biggest impact on activities in high-wage jobs; previously, automation's impact was highest in lower-middle-income quintiles.

Automation adoption per wage quintile, % in 2030, midpoint scenario

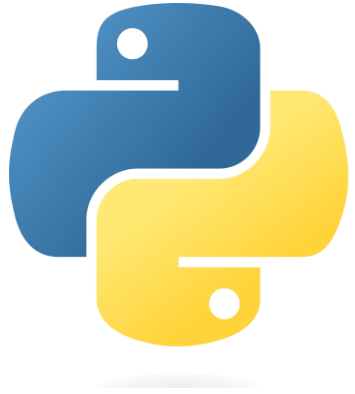
Wage quintiles: Higher earners (81–100, 61–80, 41–60, 21–40, 0–20), Lower earners. Legend: Without generative AI (white circle), With generative AI (black circle), Largest increase in automation adoption from generative AI (light blue bar), Largest automation adoption without generative AI (grey bar).



<sup>1</sup>Previous assessment of work automation before the rise of generative AI. Source: McKinsey Global Institute analysis

# Engineering & Tech Stack

# Scientific Python Stack



# Deep Learning Frameworks



need for lots of memory and compute ...

# IaaS, PaaS, SaaS



Google Cloud

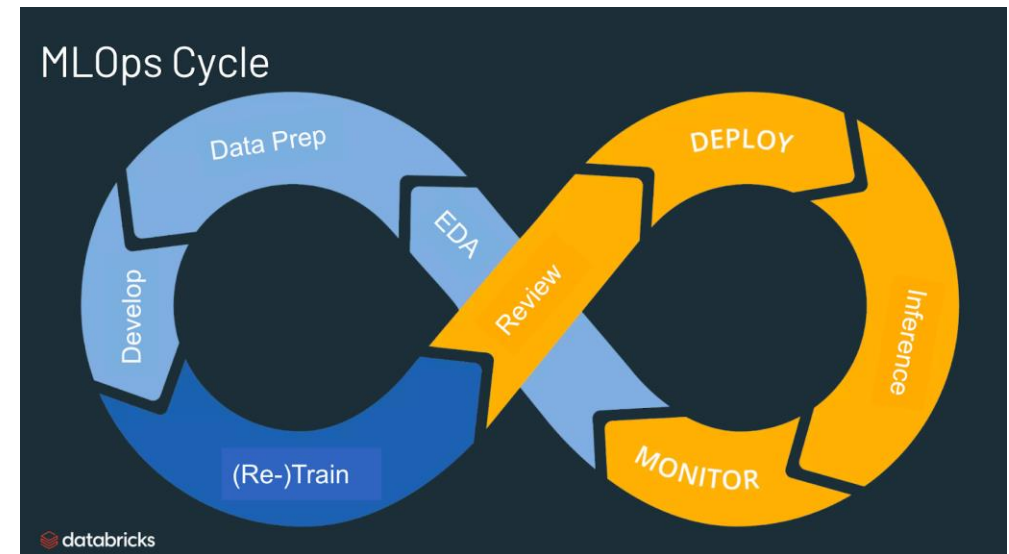
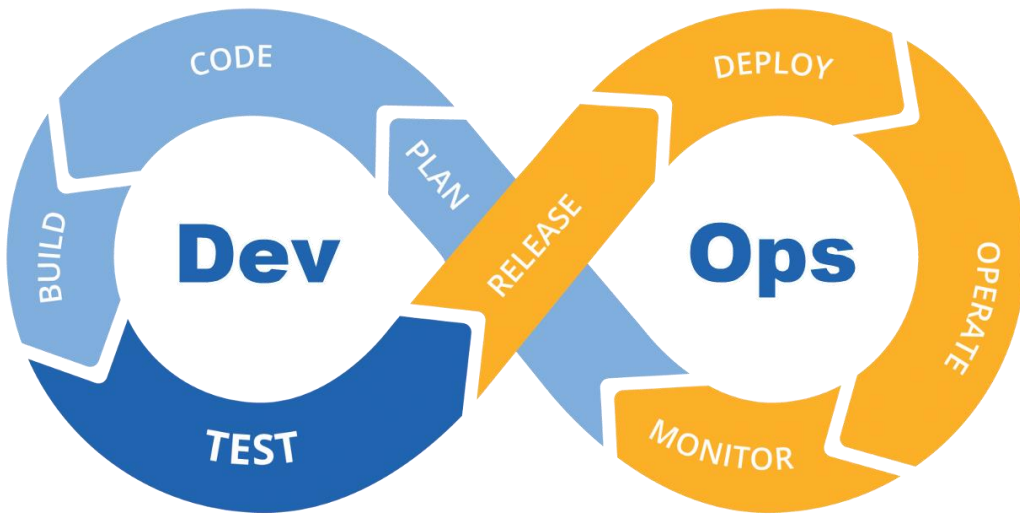


kubernetes



example for managed service:  
Azure Kubernetes Service (AKS)  
for container orchestration

# ML in Production



examples for managed ML services: Azure ML, SageMaker

# Data Management

not only compute but also cloud-based data storage

data lake: raw data

data warehouse: integrated data

unification of data lake and warehouse: ETL → ELT

NoSQL example: graph database



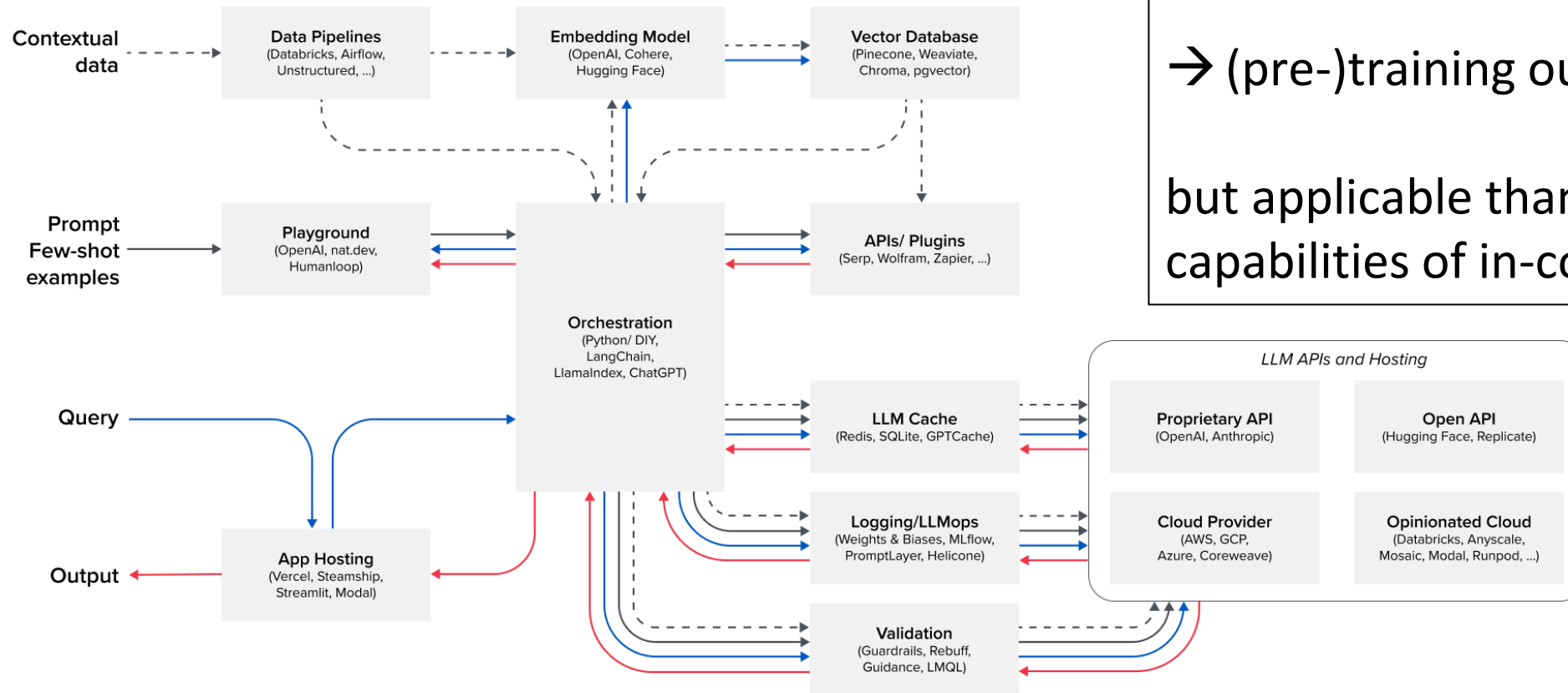


# Emerging LLM App Stack

era of large-scale models:  
AI currently on its engineering peak

→ (pre-)training out of reach for most

but applicable thanks to multi-task capabilities of in-context learning



## LEGEND

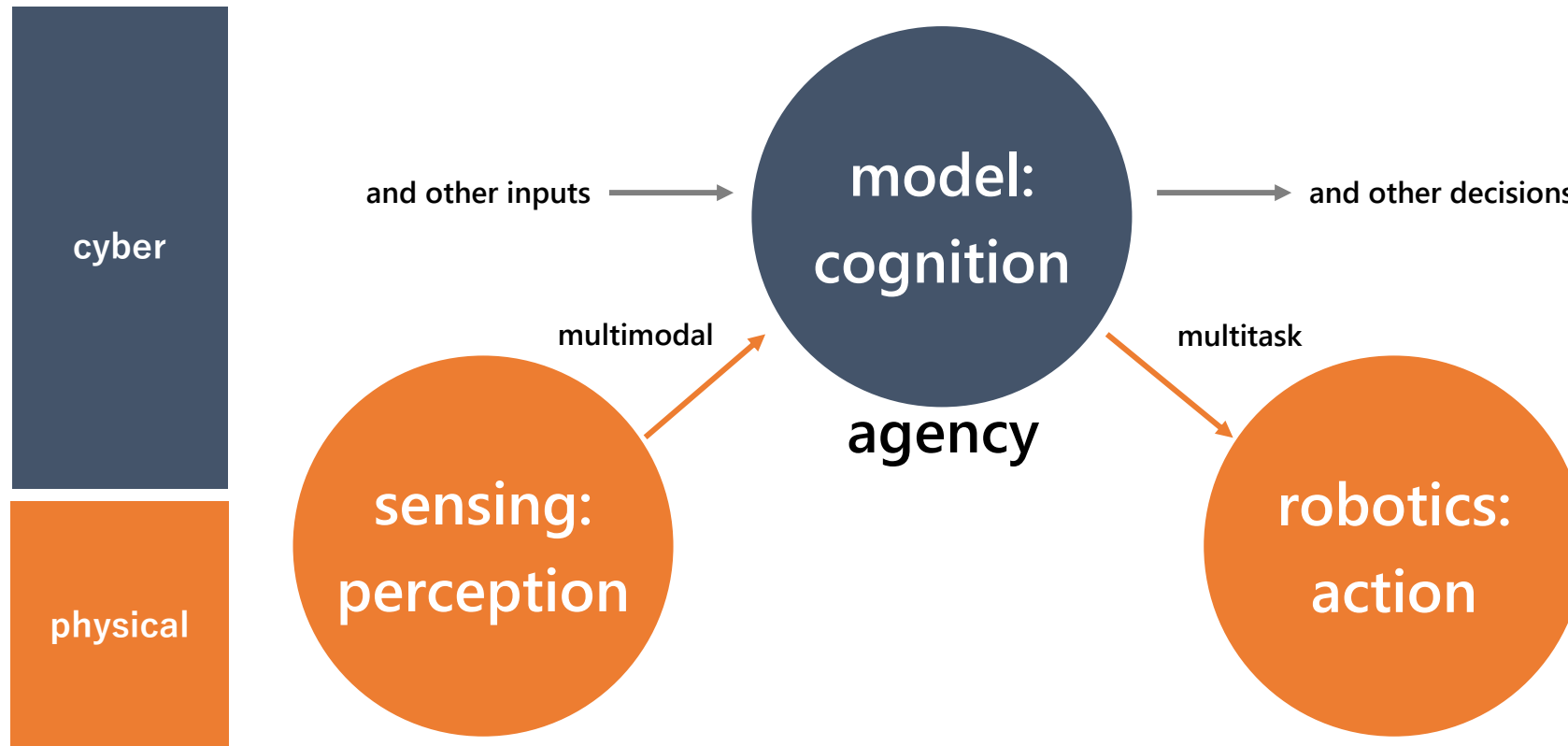
Gray boxes show key components of the stack, with leading tools/systems listed

Arrows show the flow of data through the stack

- - - - - Contextual data provided by app developers to condition LLM outputs
- > Prompts and few-shot examples that are sent to the LLM
- > Queries submitted by users
- > Output returned to users

# What The Future Holds: Autonomous Agents

learning from data



# Facts and Fiction

## **no more AI winters**

data is everywhere, plenty of useful ML products

## **“classical” ML still needed**

most value from predictive applications

## **Generative AI important step**

“mirror” to see the world in different light

## **(probably) still Chinese room**

no reliable indication for real understanding

## **AGI is not here (yet)**

but performance of multi-task and multi-modal models still improving with scale

## **AI no (foreseeable) existential risk**

more realistic and near-term risk: small number of companies control AI technology