

Gen AI & Classic ML

when to use what

July 2024

Felix Wick

Classic ML vs Deep Learning

text generation

```
done = False
total_reward = 0
while not done:
    state = torch.tensor(state, dtype=torch.float32)
    logits = policy_network(state)
    action = torch.argmax(logits).item()
    next_state, reward, done, _ = env.step(action)
    total_reward += reward
    state = next_state
    total_rewards.append(total_reward)

# Print average reward
print("Average Reward:", np.mean(total_rewards))
```

This code uses PyTorch to implement the policy gradient method (REINFORCE algorithm) to solve the CartPole problem. The policy network is defined as a simple feedforward neural network, and the training loop updates the policy network parameters to maximize the expected reward. Finally, it evaluates the learned policy by running it for 100 episodes and prints the average reward. Adjust hyperparameters and network architecture as needed for better performance.

Can you extend this to an actor-critic method?

Certainly! Here's an extension of the previous code using the actor-critic method to solve the CartPole problem:

Message ChatGPT

ChatGPT can make mistakes. Check important info.

[ChatGPT](#)

BERT family



dmlc
XGBoost

tabular data

image synthesis



Prompt: Epic anime artwork of a wizard atop a mountain at night casting a cosmic spell into the dark sky that says "Stable Diffusion 3" made out of colorful energy

[Stable Diffusion 3 — Stability AI](#)

Generative vs Predictive Models

text-to-video



Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She... +

[Sora | OpenAI](#)

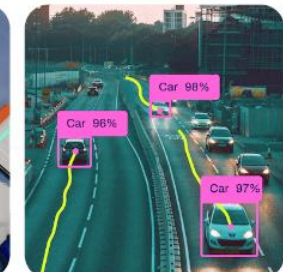
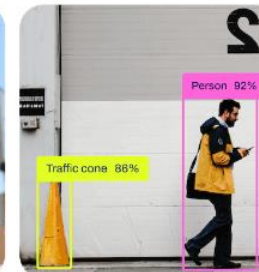
computer vision

Classify

Detect

Segment

Track



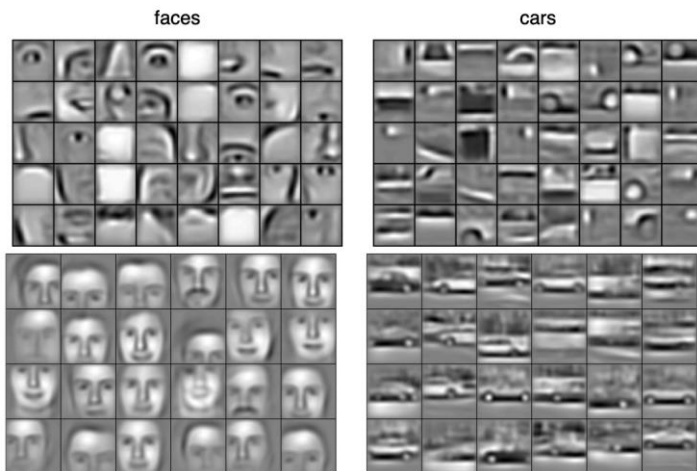
[YOLO](#)

Ladder of Generalization

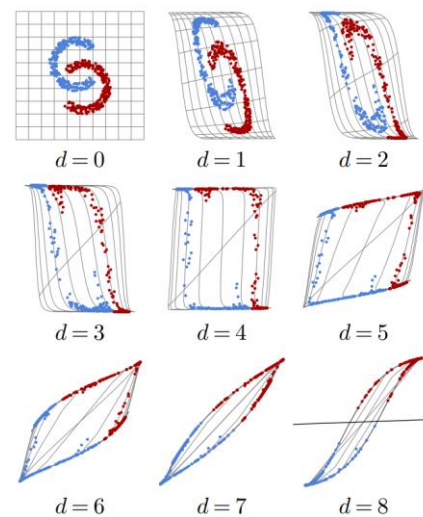
classic ML: feature engineering

deep learning: feature learning

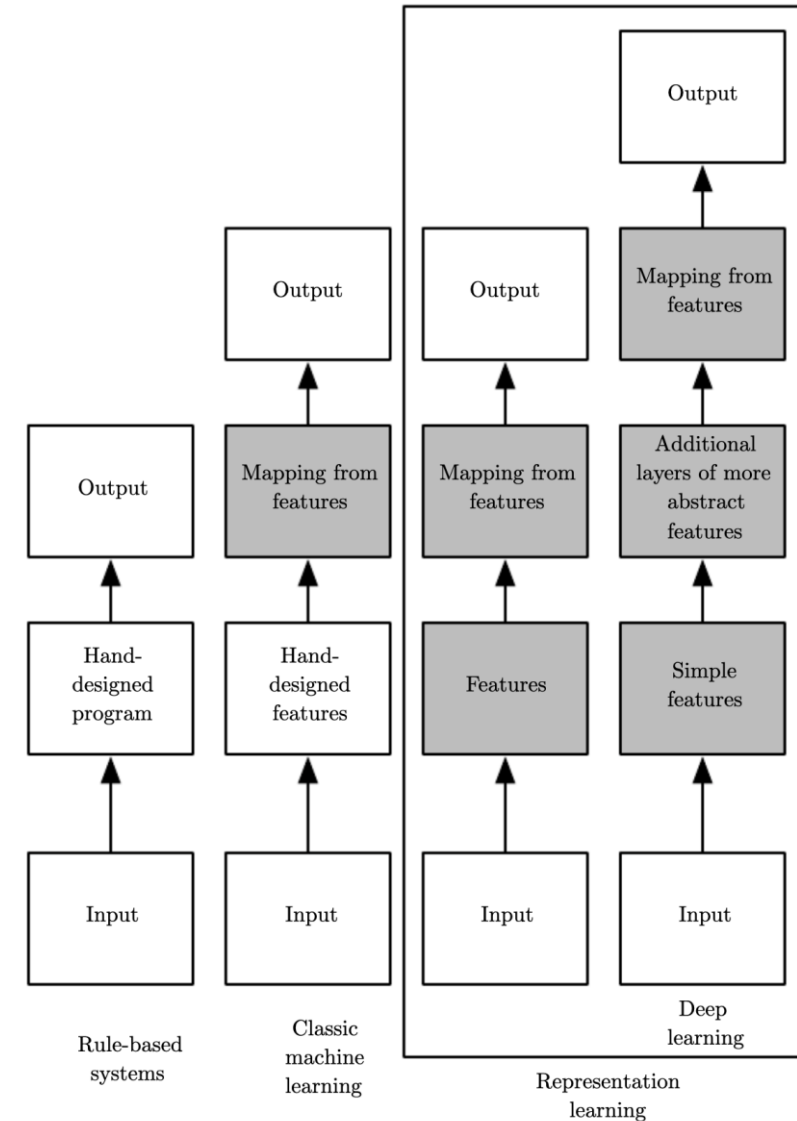
(hierarchy of concepts learned from raw data in deep graph with many layers)



[source](#)



[source](#)



[source](#)

Structured/Tabular vs Unstructured Data

unstructured data: homogenous

→ deep learning rules

→ allows transfer learning (foundation models in CV and NLP)



ImageNet

The Lord of the Rings

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

(Redirected from [Lord of the rings](#))

This article is about the book. For other uses, see The Lord of the Rings (disambiguation).
 "War of the Ring" redirects here. For other uses, see War of the Ring (disambiguation).

The Lord of the Rings is an [epic](#)^[1] [high fantasy novel](#)^[4] by the English author and scholar J. R. R. Tolkien. Set in [Middle-earth](#), the story began as a sequel to Tolkien's 1937 children's book *The Hobbit*, but eventually developed into a much larger work. Written in stages between 1937 and 1949, *The Lord of the Rings* is one of the [best-selling books ever written](#), with over 150 million copies sold.^[2]

The title refers to the story's main antagonist,²⁸ Sauron, the Dark Lord who in an earlier age created the One Ring to rule the other Rings of Power given to Men, Dwarves, and Elves, in his campaign to conquer all of Middle-earth. From homely beginnings in the Shire, a hobbit land reminiscent of the English countryside, the story ranges across Middle-earth, following the quest to destroy the One Ring, seen mainly through the eyes of the hobbits Frodo, Sam, Merry, and Pippin. Aiding Frodo are the Wizard Gandalf, the Men Aragorn and Boromir, the Elf Legolas, and the Dwarf Gimli, who unite in order to rally the Free Peoples of Middle-earth against Sauron's armies and give Frodo a chance to destroy the One Ring in the fire of Mount Doom.

Although often mistakenly called a trilogy, the work was intended by Tolkien to be one volume in a two-volume set along with *The Silmarillion*.^[37] For economic reasons, *The Lord of the Rings* was first published over the course of a year from 29 July 1954 to 20 October 1955 in three volumes rather than one^[38] under the titles *The Fellowship of the Ring*, *The Two Towers*, and *The Return of the King*. The *Silmarillion* appeared only after the author's death. The work is divided internally into six books, two per volume, with several appendices of background material.^[3] These three volumes were later published as a boxed set, and even finally as a single volume, following the author's original intent.

structured data: heterogenous

→ feature engineering needed

→ deep learning loses its advantage over shallow methods

→ e.g., gradient boosting still used a lot

[illegible]

Transfer Learning

idea:

- generic pre-training of foundation models on huge data sets
 - subsequent fine-tuning for specific tasks on small(er) data sets
- (usually done with deep learning methods, using its compositional nature)

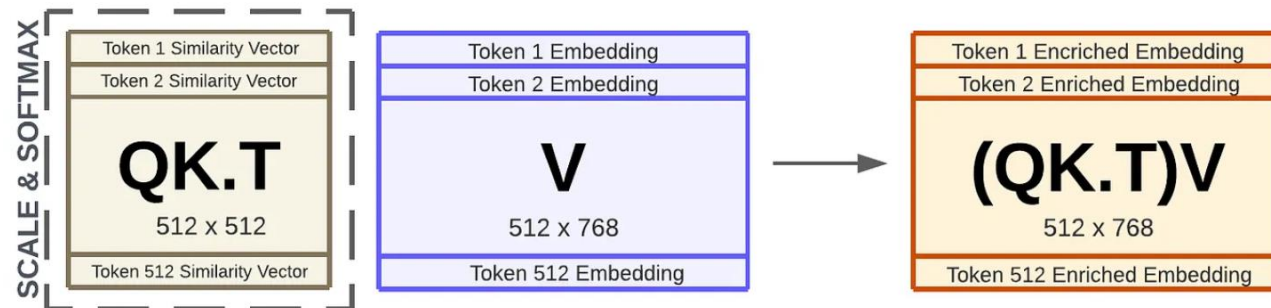
very successful for:

- computer vision (e.g., object classification)
- language models (e.g., BERT, GPT)

not (yet) for tabular data (due to its notorious heterogeneity)

Language Models: Contextual Semantics

- self-supervised learning: e.g., next/masked-word prediction
- tokenization: split text into chunks (e.g., words)
- semantics by means of vector embeddings: e.g., via bag-of-words (or end-to-end in transformer)
- positional encoding & embeddings: order of sequence
- contextual embeddings: (self-)attention (weighted averages: influence from other tokens)

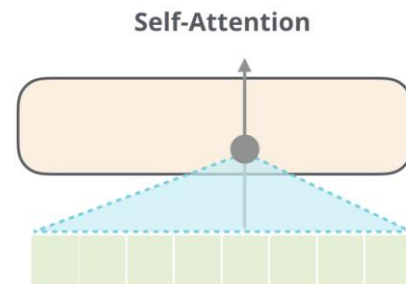


[source](#)

Encoder vs Decoder LLMs

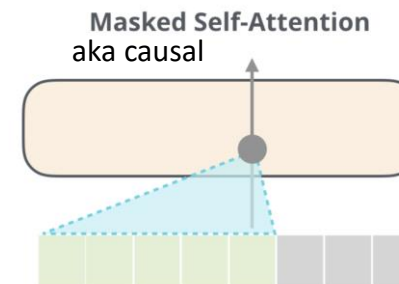
encoder-only LLMs

- prime example: BERT
- self-supervised pre-training: masked-word prediction
- fine-tuning on downstream tasks (e.g., sequence classification)
- can't generate text
- can't be prompted



decoder-only LLMs

- prime example: GPT
- self-supervised pre-training: next-word prediction
- instruction tuning (e.g., RL from human feedback)
- generate text: chat bots
- prompt engineering



Generative vs Predictive/Discriminative Models

discriminative models:

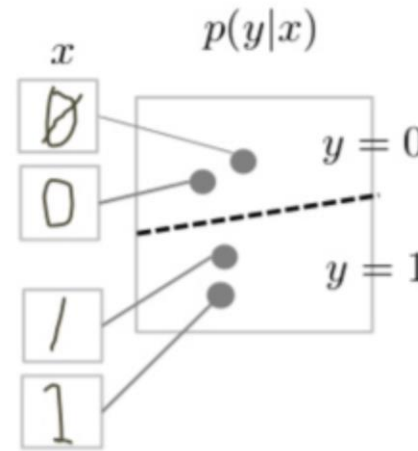
predict conditional probability $P(Y|\mathbf{X})$

generative models:

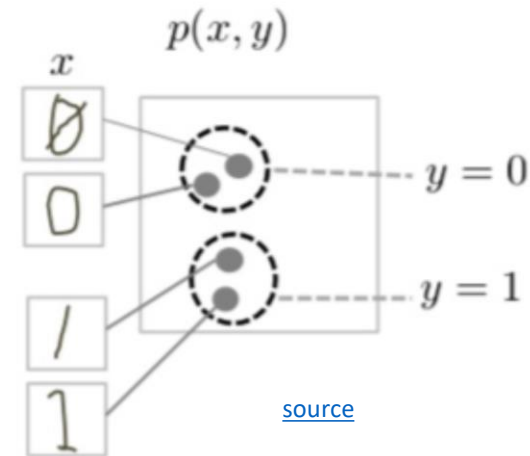
predict joint probability $P(Y, \mathbf{X})$

(or just $P(\mathbf{X}) \rightarrow$ unsupervised learning)

discriminative model



generative model



task of generative models more difficult: need to model full data distribution rather than merely find patterns in inputs to distinguish outputs

generative models allow to generate new data samples (text, images, video, proteins, ...)

predictive models usually better for predictive tasks, business problems often specific/predictive

Deep Learning for Generative AI

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

- text generation: decoder LLMs
- image synthesis: diffusion models

note the difference between image synthesis and multimodal understanding in LLMs
(images as additional input sequences to transformer, tokenized by splitting into patches)

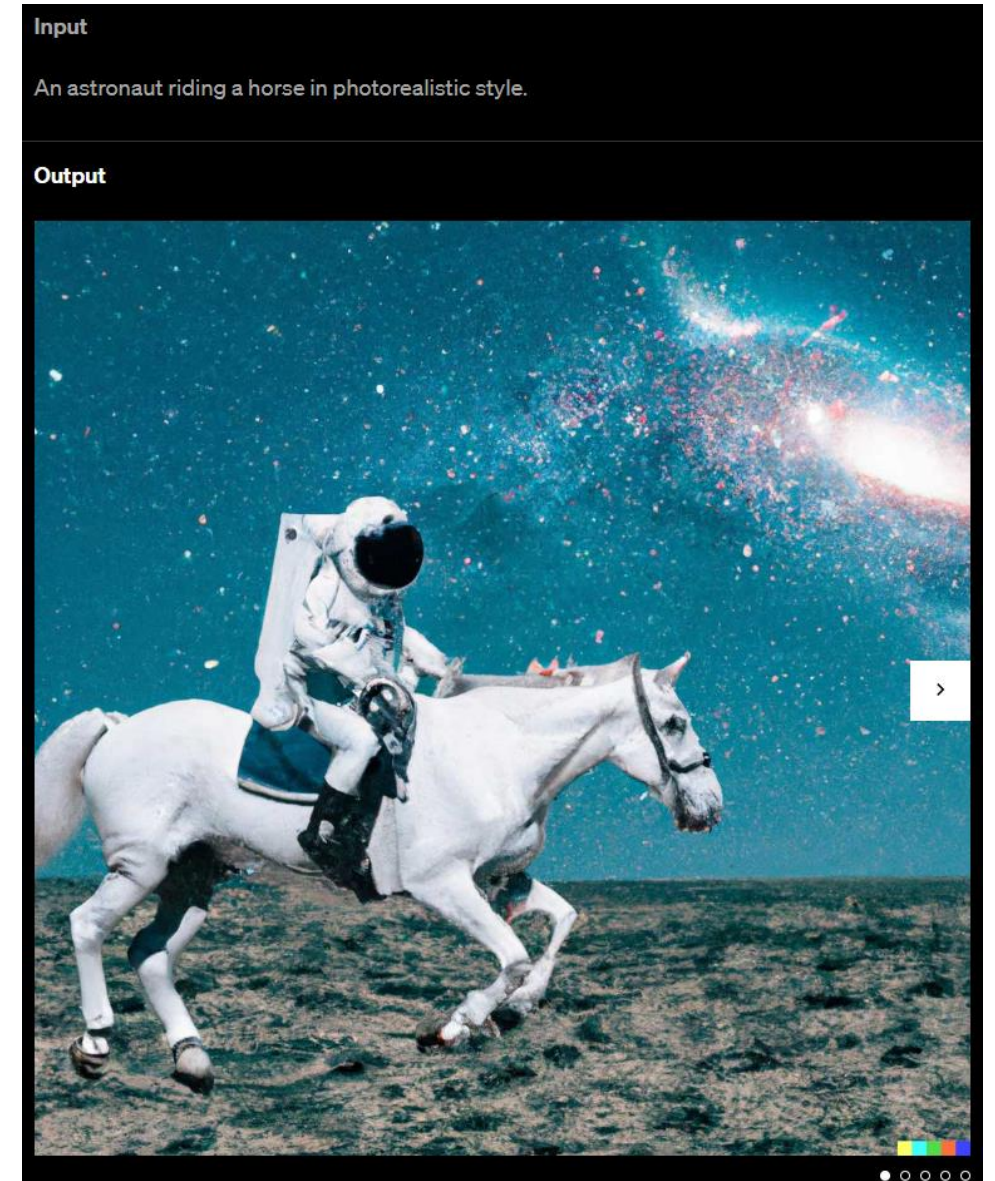
Image Synthesis

idea: generate new images as variations of training data

usually conditioned on text (prompt) by transformers

compared to text generation, additional mechanism needed (e.g., diffusion) to create more complex structures

example: DALL-E 2



Text Generation

in-context learning as alternative to fine-tuning (new paradigm):

feed information into LLM via input prompt (decoder LLMs)

→ attention to context

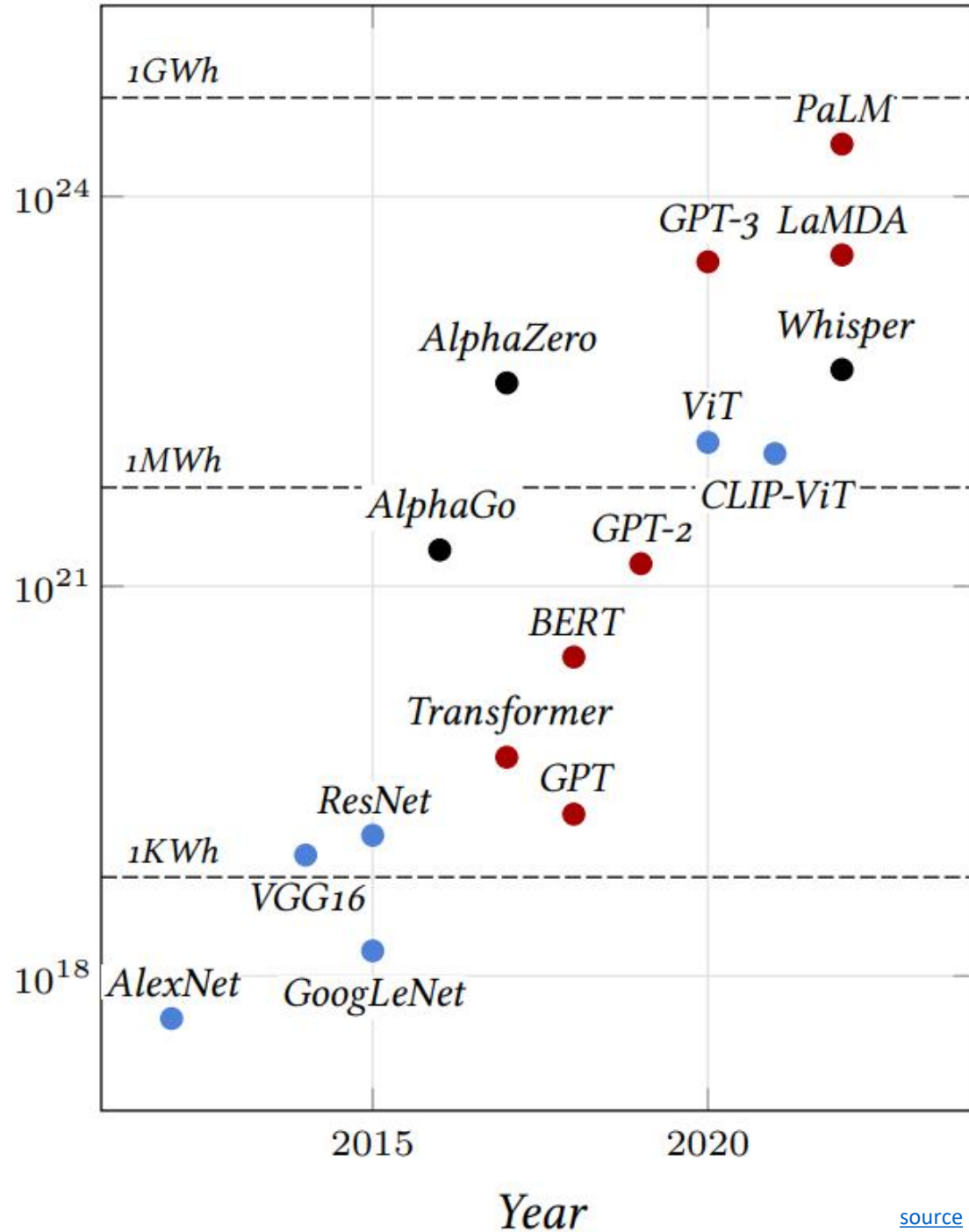
typical prompt:

instructions, context (potentially retrieved), query, output indicator

enables multi-task capabilities (all of ML before was only narrow tasks)

→ assistants

Training cost (FLOP)



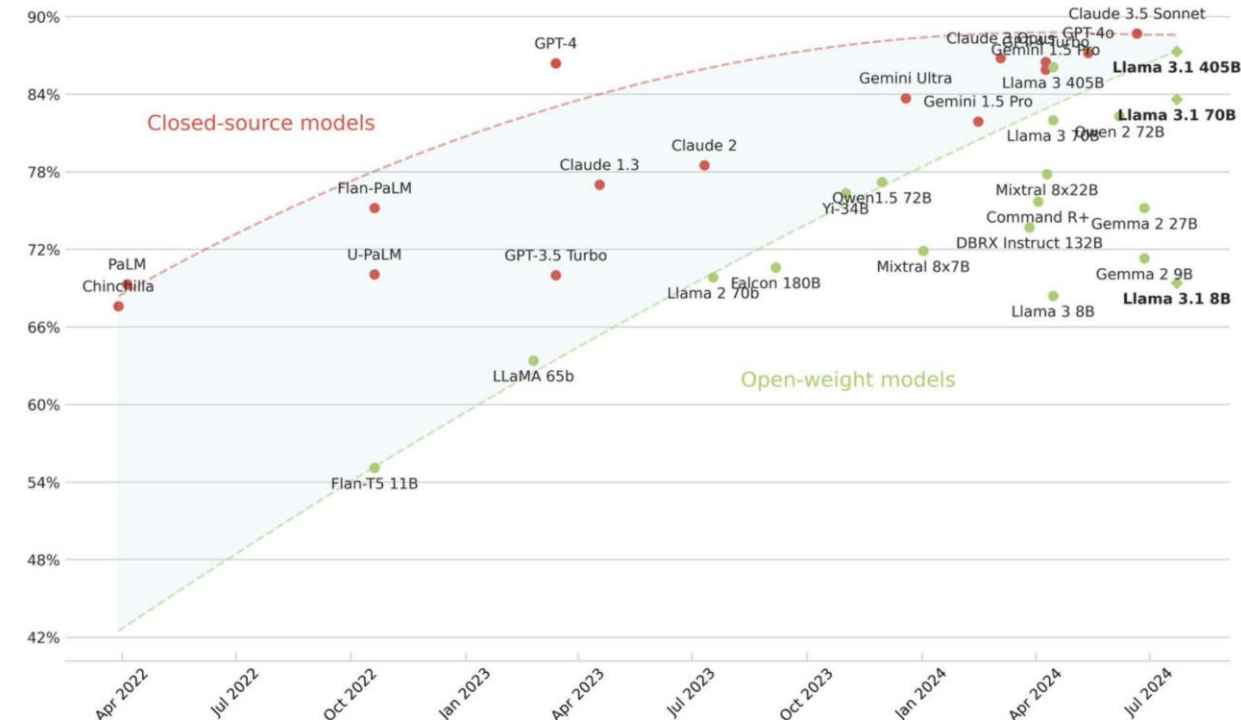
[source](#)

(open-source) SOTA (July 2024):
[Llama 3](#)

Closed-source vs. open-weight models

Llama 3.1 405B closes the gap with closed-source models for the first time in history.

MMLU (5-shot)



@maximelabonne

a few surveys: [1](#), [2](#), [3](#)

Some LLM Numbers

example Llama 3 405B:

- vocabulary size (tokens): 128K
 - embedding/model dimensions: 16,384
 - parameters: 405B
 - training tokens: 15.6T
 - context length/window (tokens): 128K
 - training hardware: 16K GPUs (H100)
- ← factor less than 40
→ a lot of memorizing

LLM Agents

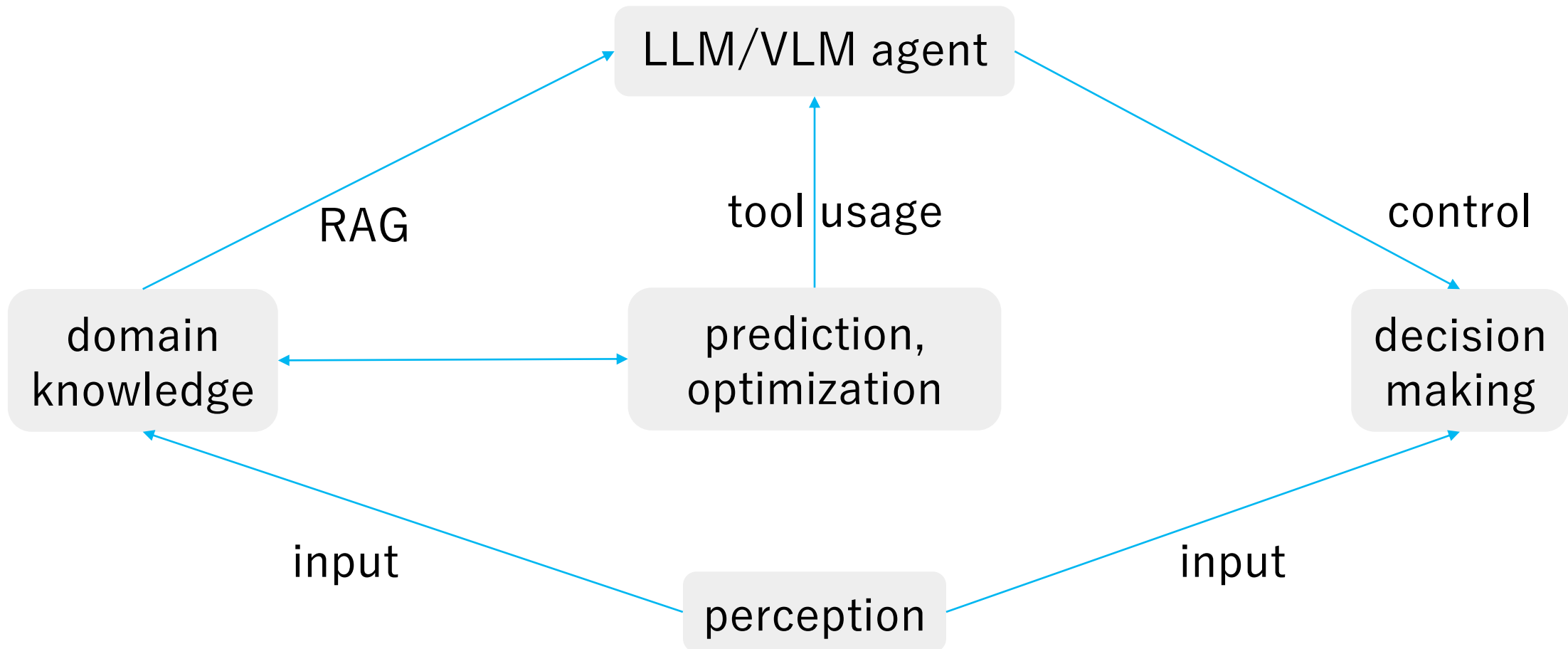
current AI good at learning statistical patterns and making predictions

but no real “understanding”, and limited reasoning and planning capabilities

desired agent capabilities:

- planning (LLM: decomposition of complex issue in multiple simple steps)
- tool use (LLM: use predictive models for numerical/optimization tasks)
- reflection
- collaboration with other agents

Goal: Autonomous End-to-End Workflow



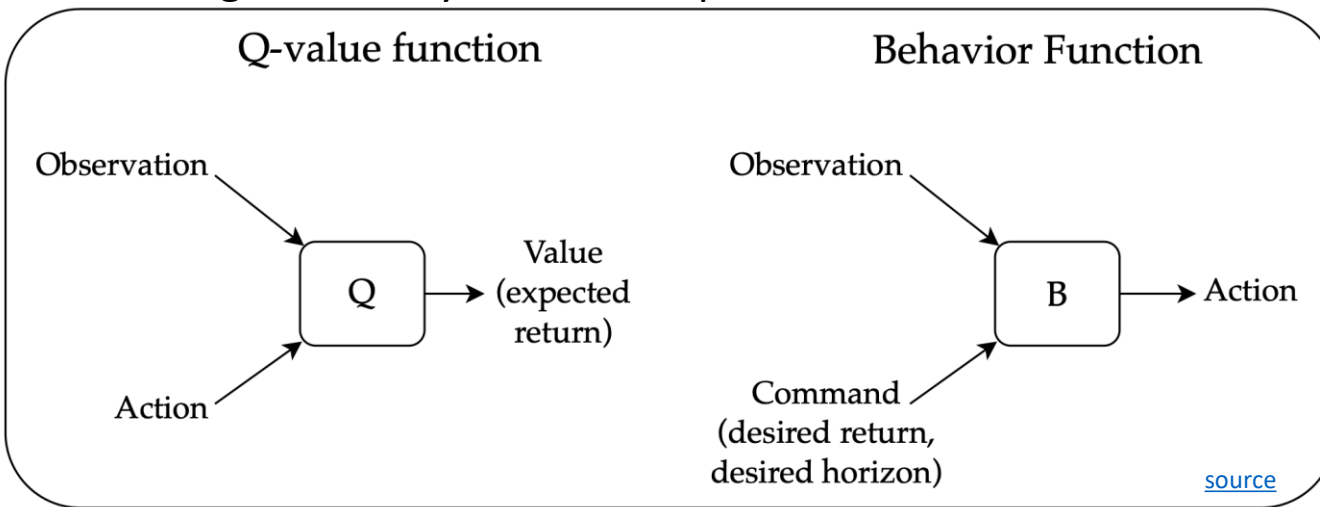
Sequential Decision Making

typically, domain of reinforcement learning

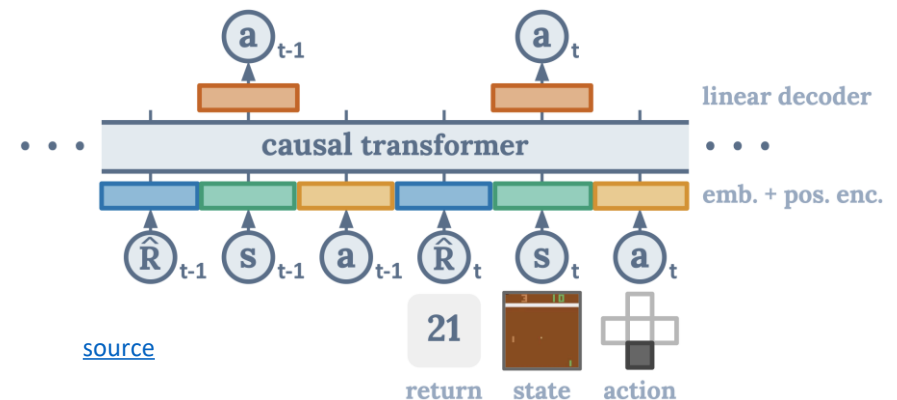
sequence modeling as alternative:

- generative: transformer decoder to autoregressively model trajectories
- credit assignment directly via self-attention: state-return associations
- desired return tokens as prompt for action generation

overcoming the deadly triad of deep RL:

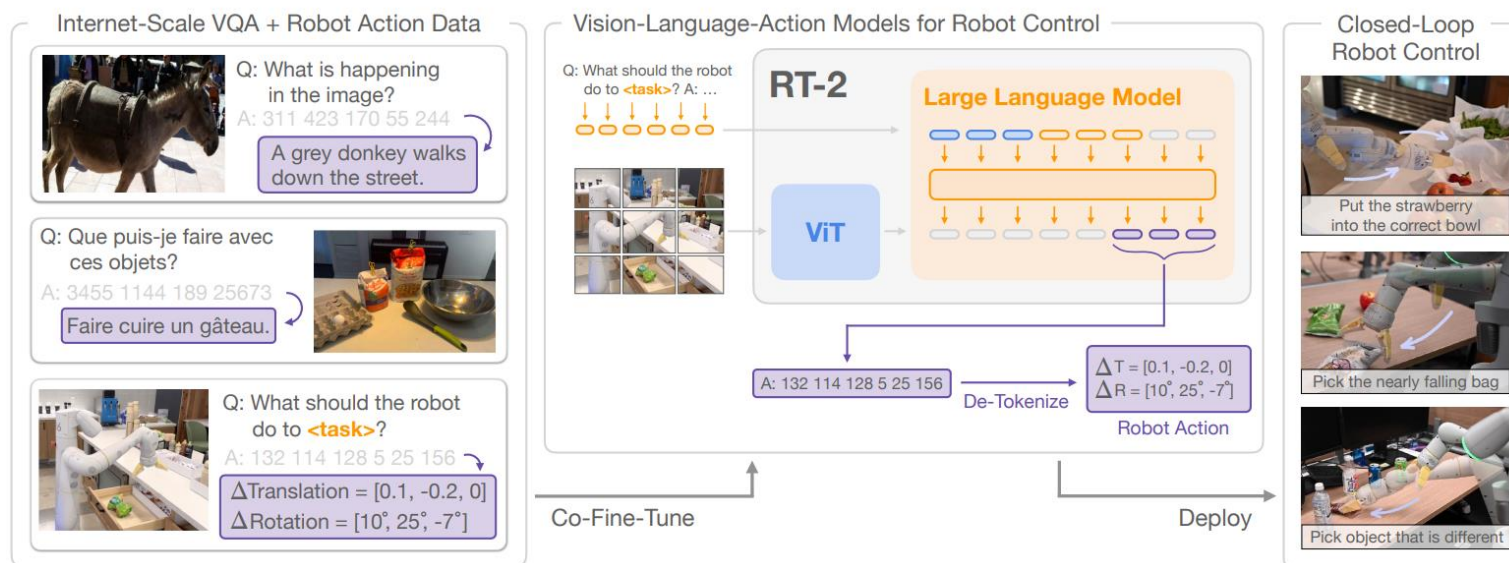


Decision Transformer:



Robotic Control generated by LLMs/VLMs

RT-2:



Code as Policies:

SayCan (grounding with pre-trained skills):

