Learning Concepts and Doing Experiments with Language, Code, and Probability

Kevin Ellis

Work with:
Wen-Ding Li, Keya Hu,
Top Piriyakulkij, Yichao Liang,
Cassidy Langenfeld, Hao Tang,
Evan Pu, Zenna Tavares

The Learning + Recursion Lab

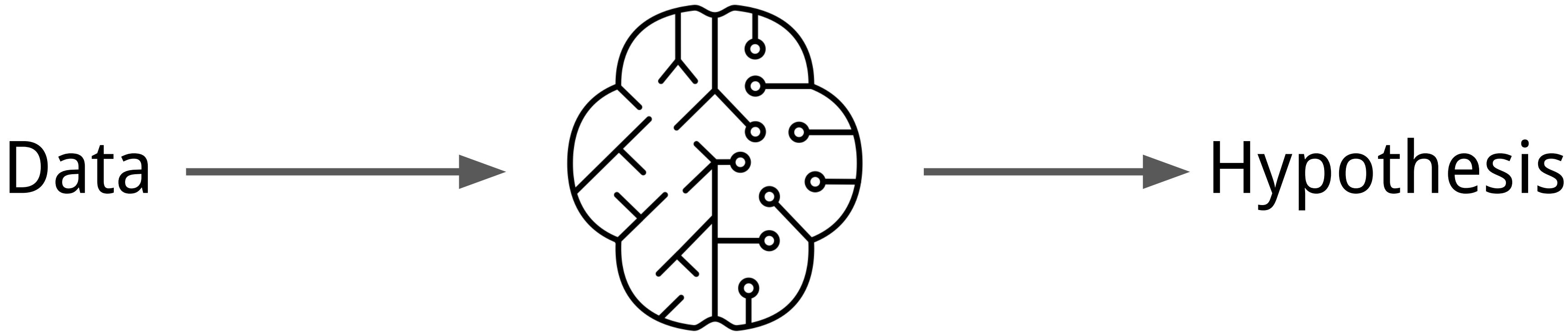
What you'll learn in this talk

1. Computational models of human few-shot learning

2. How to make LLMs better at forming hypotheses and doing experiments

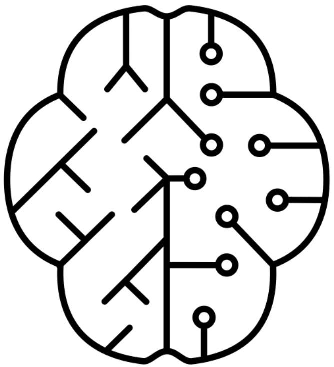
Part 1:

Human Few-Shot Learning



Goal: Generalization to unseen test data , aiming for human-like efficiency and flexibility *

*Few examples; Low-dimensional inputs

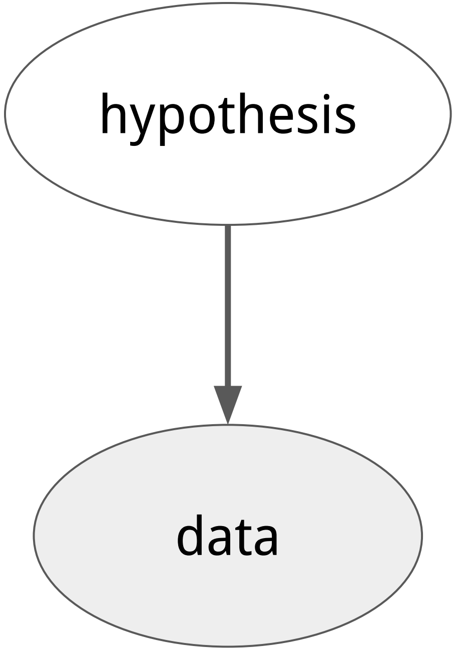
Data →  → Hypothesis

Efficient: few examples needed ⇨ Human-like prior

Flexible: infinite, diverse concepts ⇨ Compositional hypothesis space

Efficient (v2): little time/energy ⇨ Tension with flexibility……

Tractability vs Expressivity



$$\underset{\text{hypothesis}}{\operatorname{argmax}} \quad p(\text{hypothesis}) \times p(\text{data} \mid \text{hypothesis})$$

Tiny Subset of What Humans Learn

Popular Idea: Compositional Hypothesis Space
Recursive + Expressive

“possession of the infinitely many concepts that are expressible in an innate language of thought would be a curse: **the curse of a compositional mind.”**

Spelke [2022]

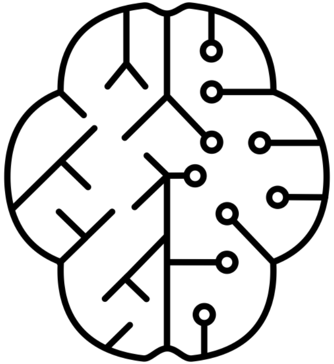
Languages for Composition

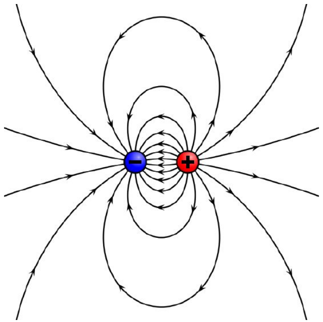
Coulomb's Law

$$\vec{F} \propto \frac{q_1\, q_2}{|\vec{r_1} - \vec{r_2}|^2} \widehat{r_1 - r_2}$$

Composed subparts: vector algebra ops

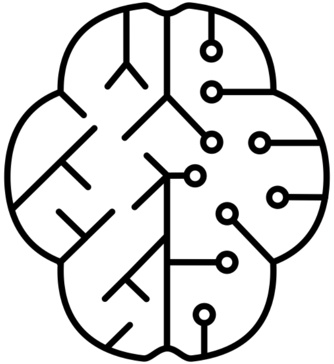
Dissecting the Curse of Compositionality

Data →  → Inference



$$\vec{F} \propto \frac{q_1\, q_2}{|\vec{r_1} - \vec{r_2}|^2}\widehat{r_1 - r_2}$$

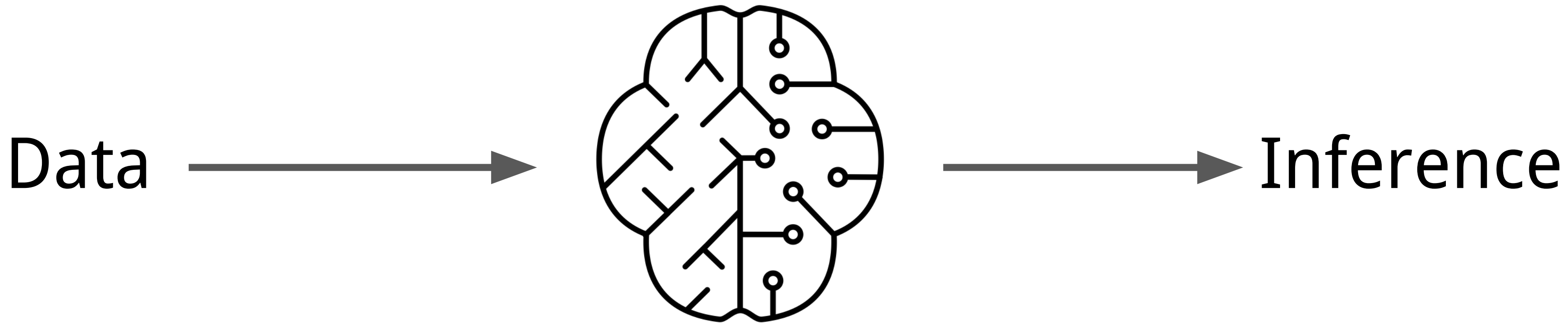
Dissecting the Curse of Compositionality

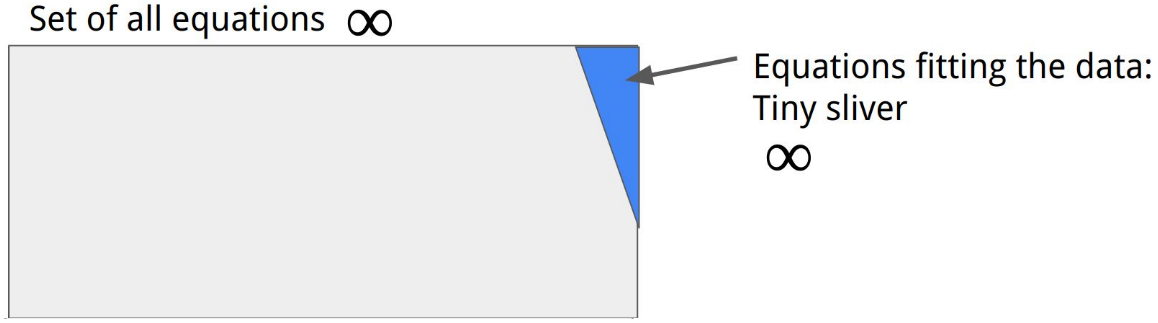
Data →  → Inference

Set of all equations ∞

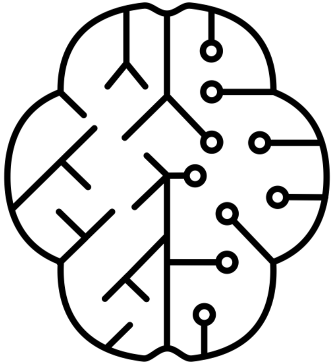
Equations fitting the data:
Tiny sliver
∞

Dissecting the Curse of Compositionality





Dissecting the Curse of Compositionality

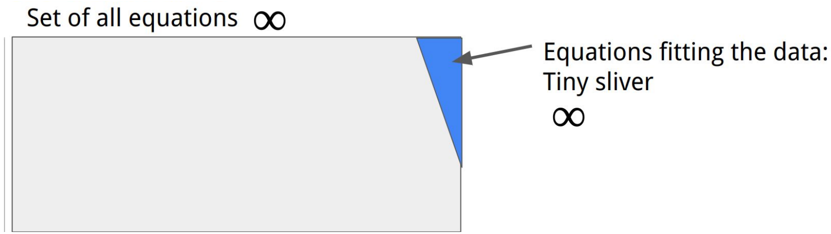
Data →  → Inference

Curse #1:

Most hypotheses don't work

Curse #2:

Infinitely many hypotheses work

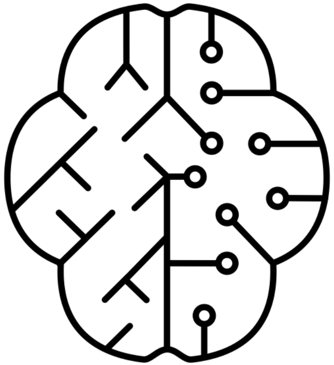


Part 1, Human part of talk:

Taming the Curse of Compositionality, using Bayes and Natural Language

Data

“Axolotl”



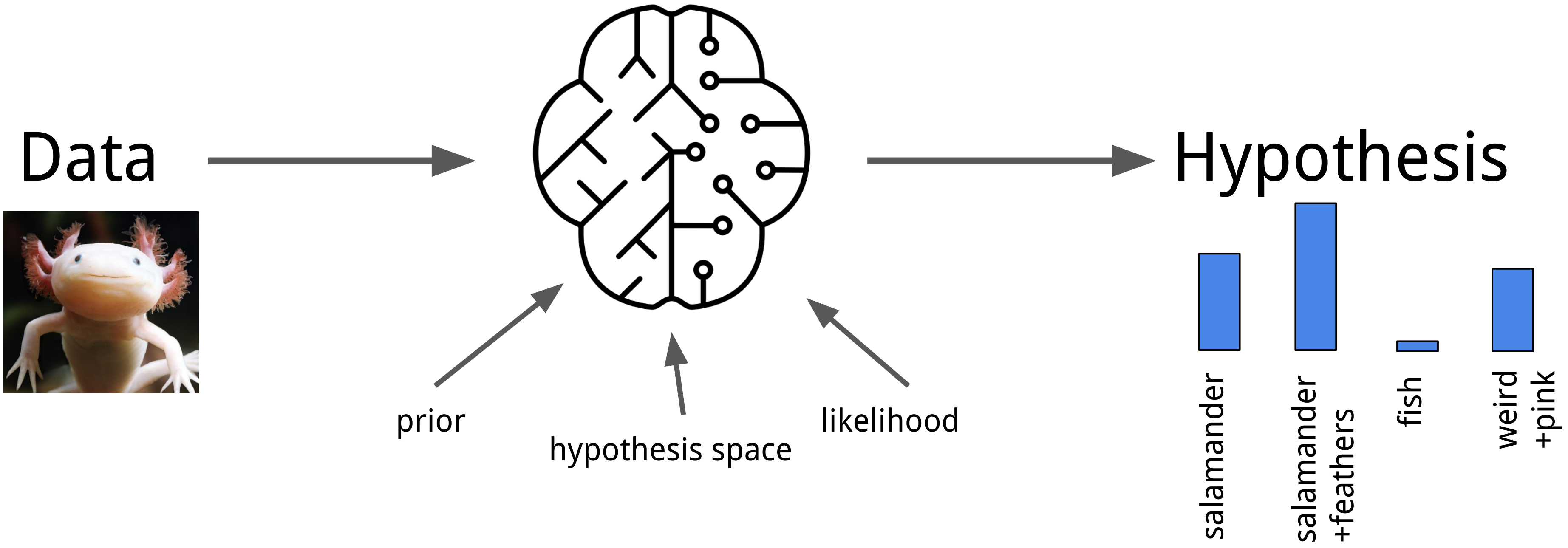
Hypothesis

“Salamander with feathers”

Model

P(hypothesis | data) ∝ P(data | hypothesis) × P(hypothesis)

Reverend Bayes

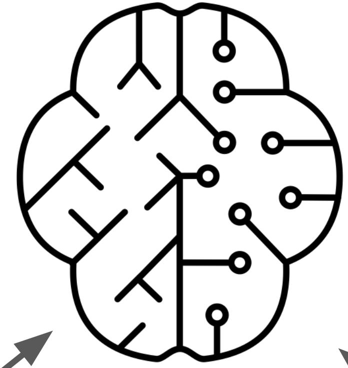


Model

Reverend Bayes

P(hypothesis | data) ∝ P(data | hypothesis) × P(hypothesis)

Data

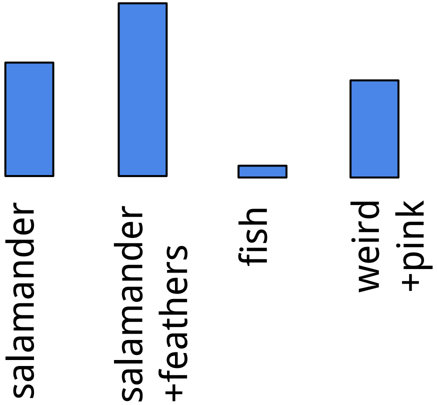


prior

hypothesis space:
natural language

likelihood

Hypothesis



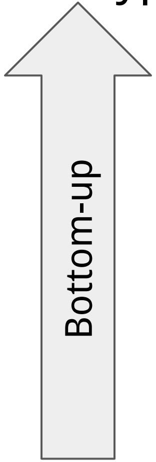
See Latent Language:
Andreas et al. 2018

Tractable Approximate Inference:

Top-down + Bottom-up

Tractable Approximate Inference

Candidate Hypotheses

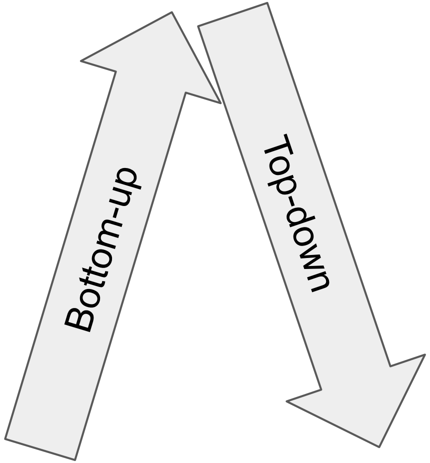


Bottom-up Proposals:
Imprecise, learned, data-driven

Data

Tractable Approximate Inference

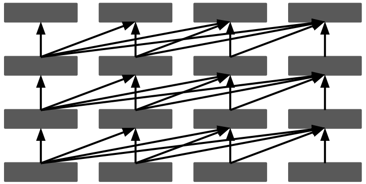
Candidate Hypotheses



Data

Bottom-up Proposals:
Imprecise, learned, data-driven

LLM



Top-down Reasoning:
Probabilistic inference

Reverend Bayes

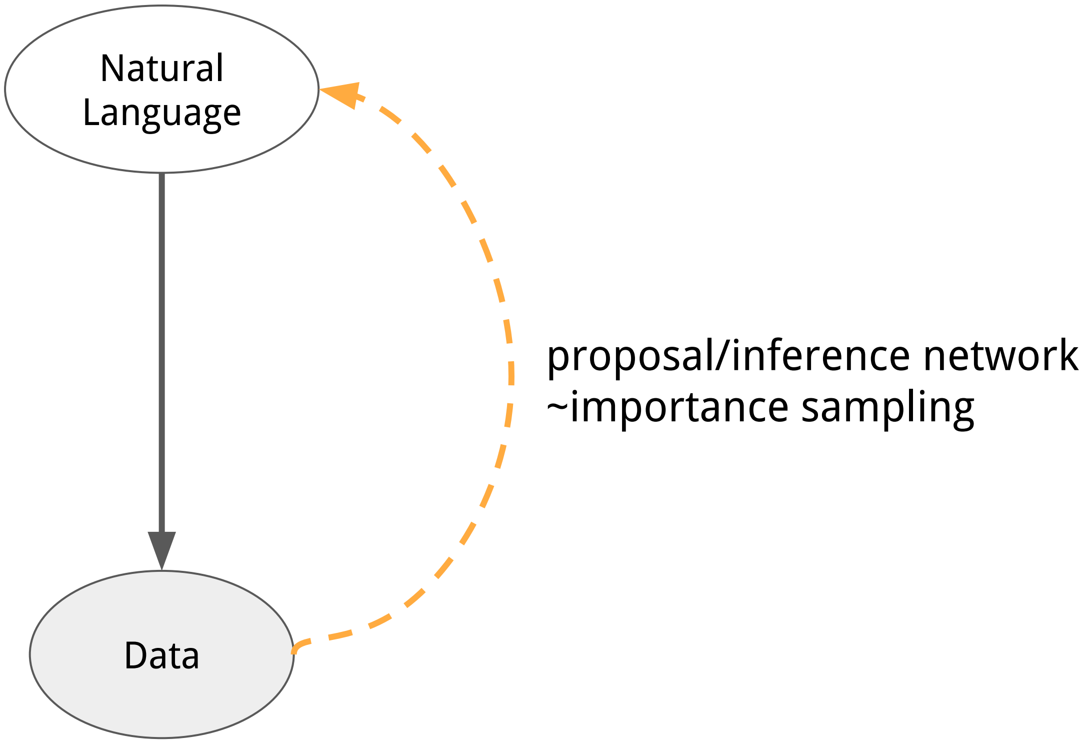
Bayesian Network

Bayesian Network: Learnable Prior

Prior: LLM.
Autoregressive ⇒ Occam's
Tune prior to human data

Bayesian Network:
Data-Driven Proposal Distribution

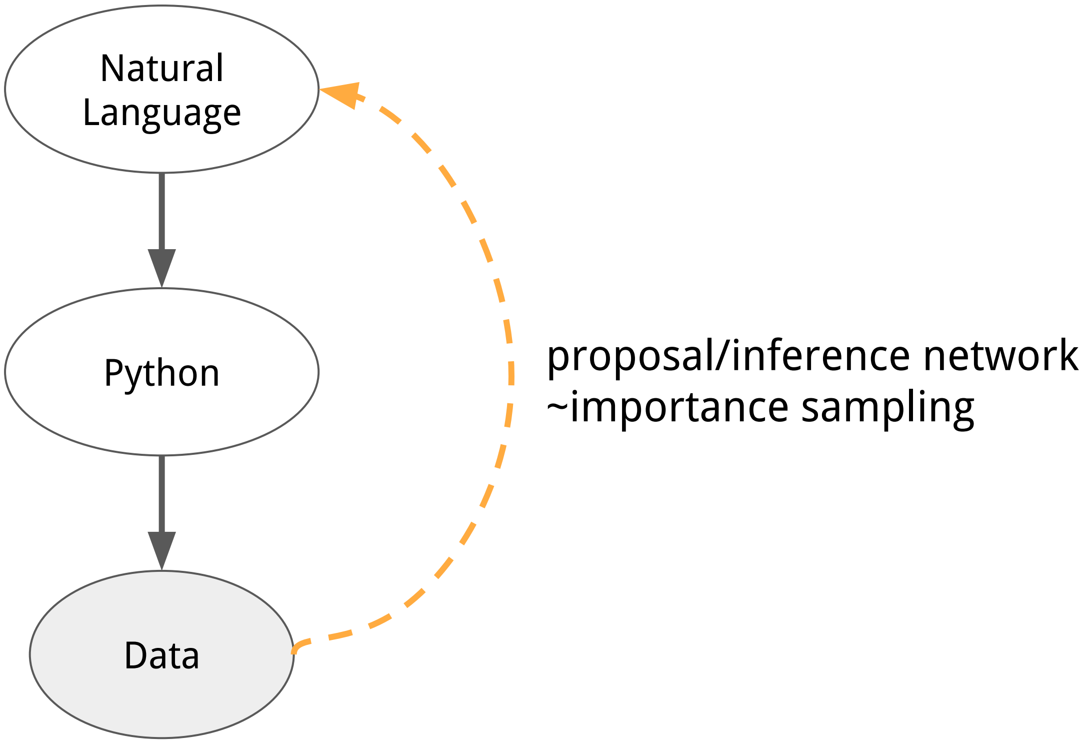
Prior: LLM.
Autoregressive ⇒ Occam's
Tune prior to human data



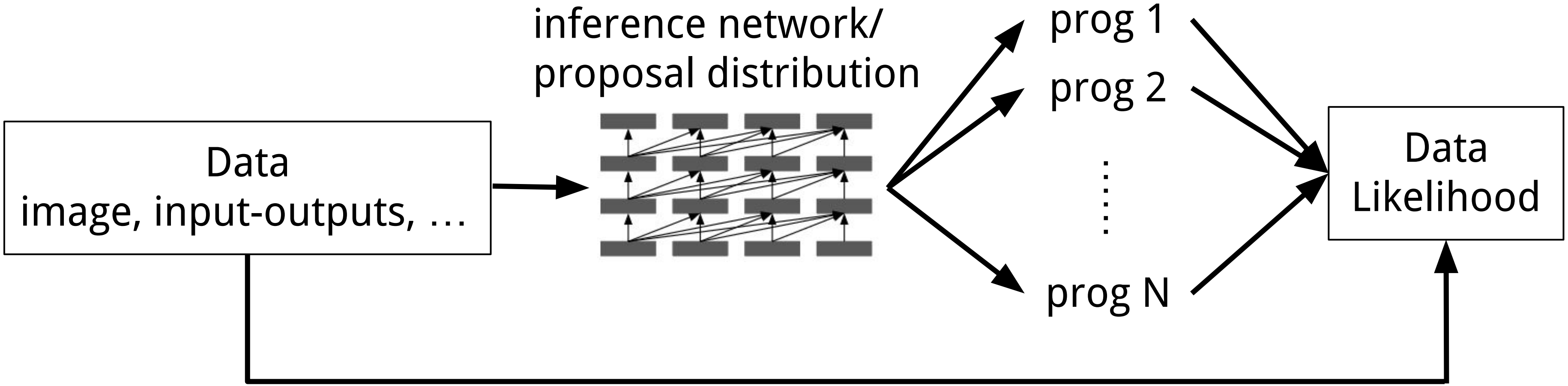
Bayesian Network:
Likelihood via Python Code Generation

Prior: LLM.
Autoregressive ⇒ Occam’s
Tune prior to human data

Likelihood:
Convert NL to Python



Likelihood filters out bad proposals



Model vs Humans: Logical Concepts

Logical Concepts

"Bachelor"

$$(\text{Male} \wedge \neg \text{Married})$$

"Valedictorian"

$$\text{Valedictorian}(x) \iff (\forall y : \text{School}(x) = \text{School}(y) \implies \text{GPA}(x) \geq \text{GPA}(y))$$

Task+Data From Piantadosi et al. 2016: 112 concepts, >1k human participants

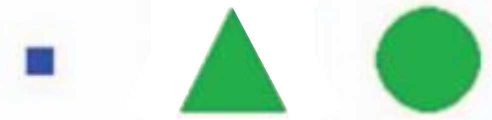
Example 1: No shapes in concept

Task from:
Piantadosi et al. 2016

Example 1: No shapes in concept

Task from:
Piantadosi et al. 2016

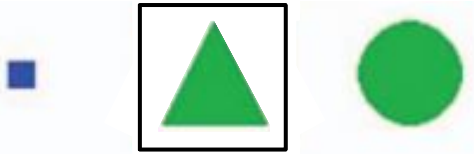
Example 2



Example 1: No shapes in concept

Task from:
Piantadosi et al. 2016

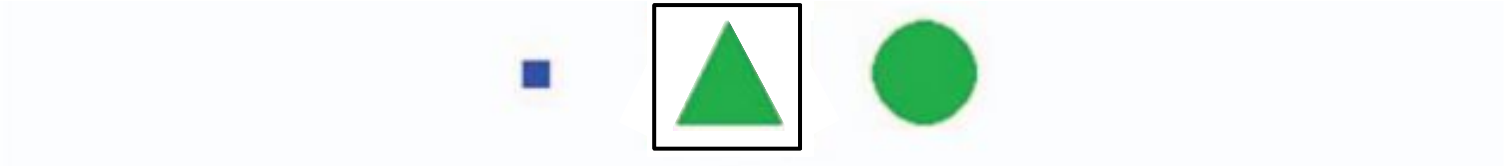
Example 2: Only middle shape in concept



Example 1: No shapes in concept

Task from:
Piantadosi et al. 2016

Example 2: Only middle shape in concept



Example 3: Which shapes are in the concept?

Yes

No

Yes

No

Yes

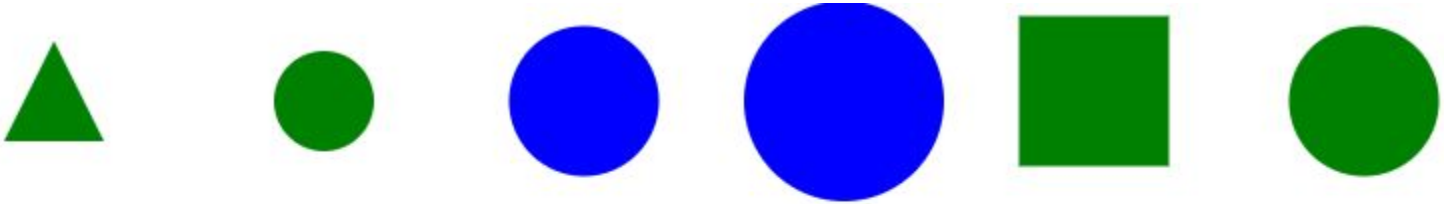
No

Yes

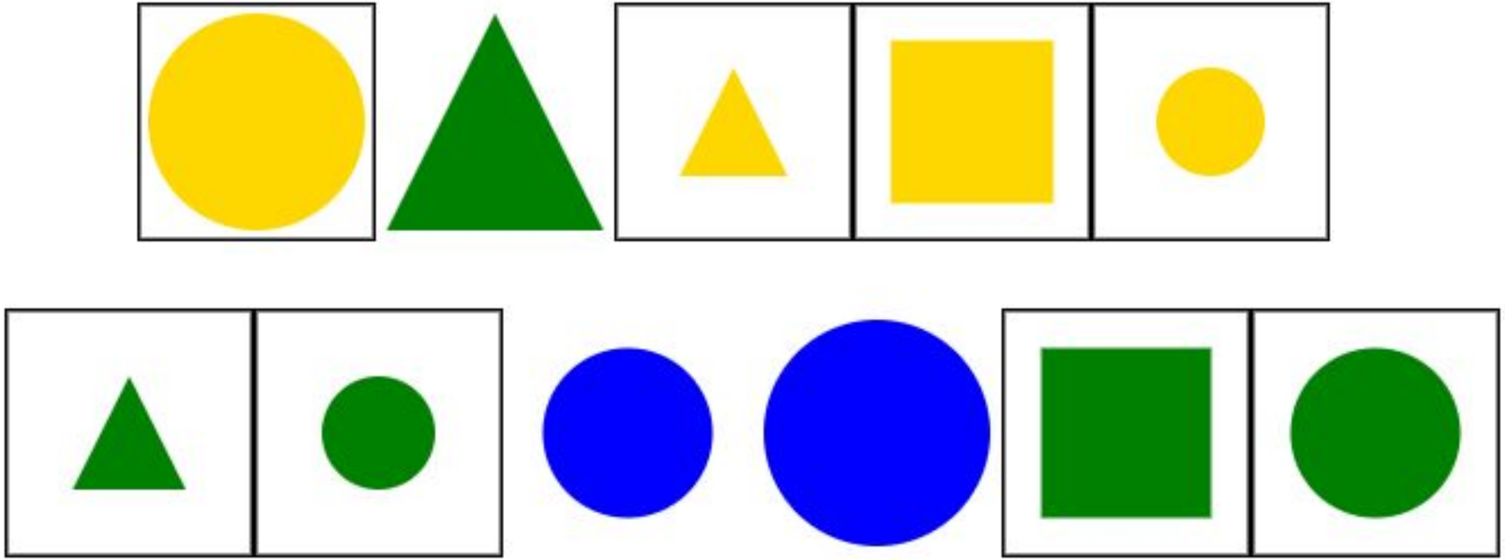
No

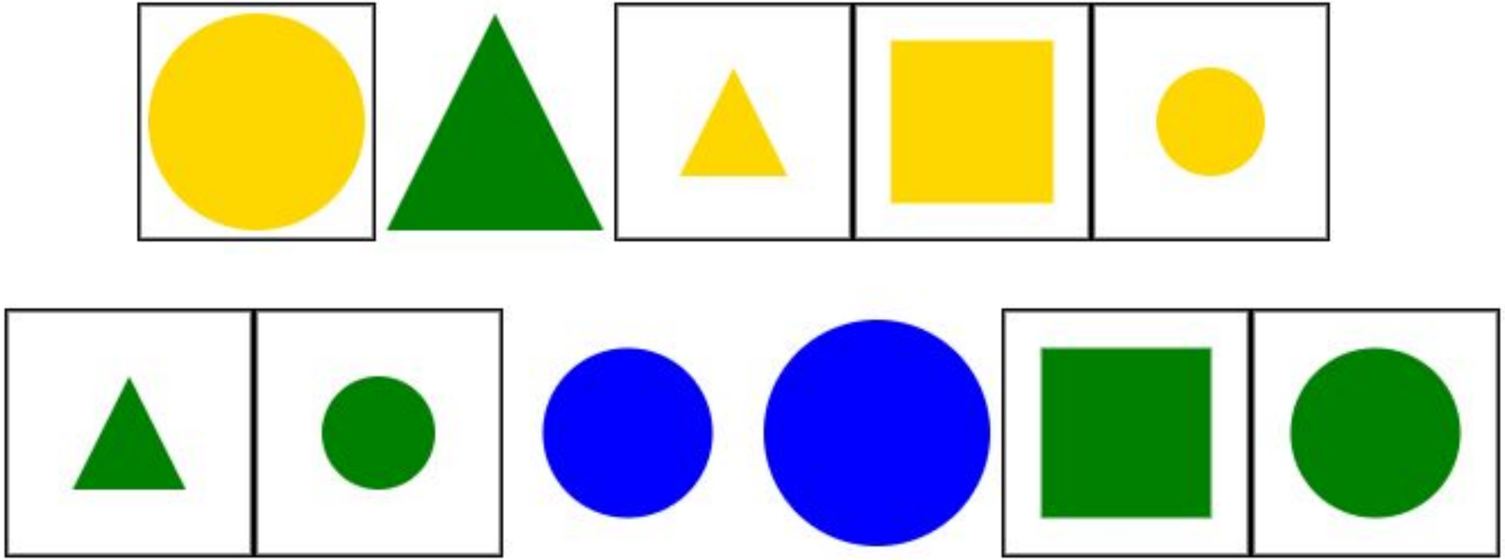
Yes

No

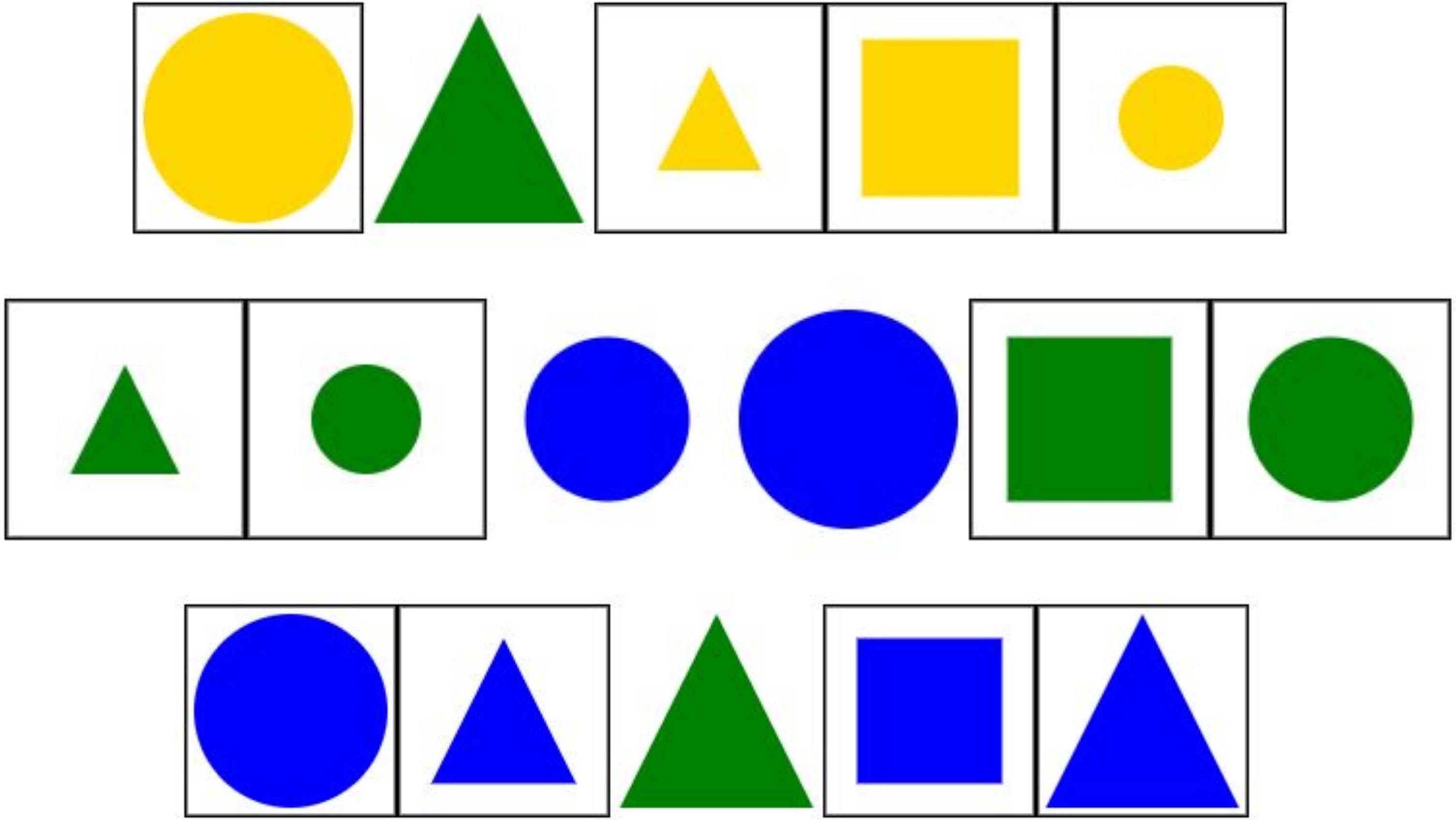


Yes	Yes	Yes	Yes	Yes	Yes
No	No	No	No	No	No



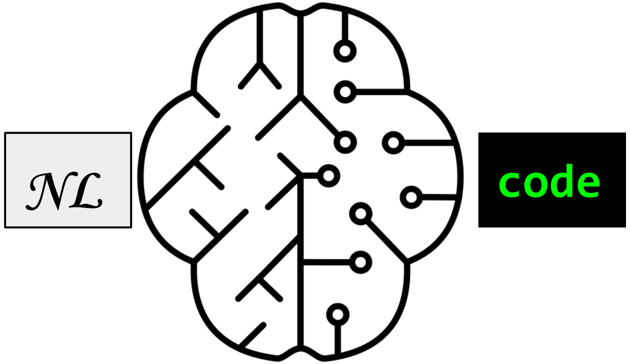


Yes	Yes	Yes	Yes	Yes
No	No	No	No	No



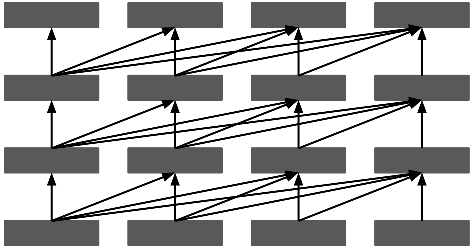
Model

Hypothesis Space



Bayes,
learnable neural prior

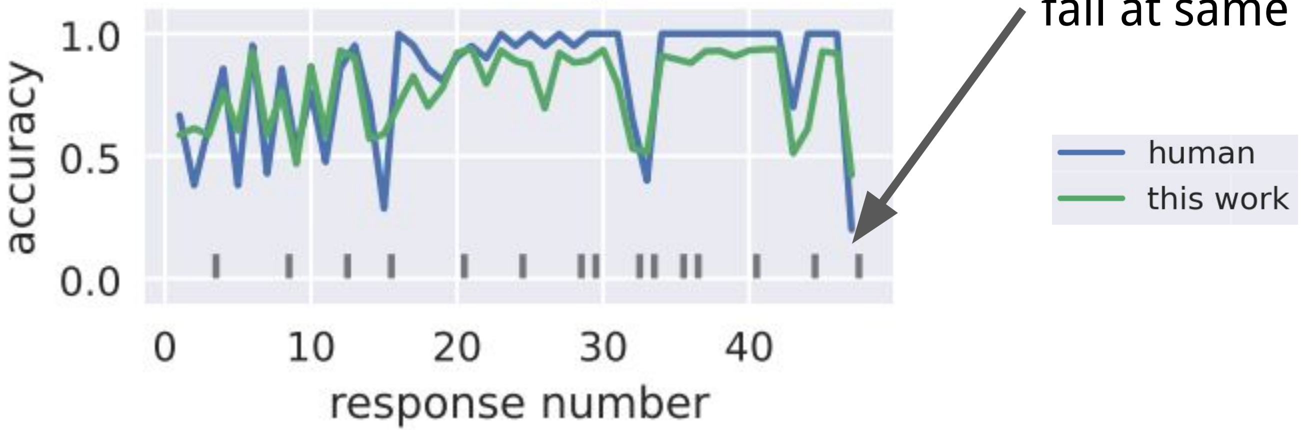
LLM Proposal distribution



Model Predicts Human Learning Dynamics

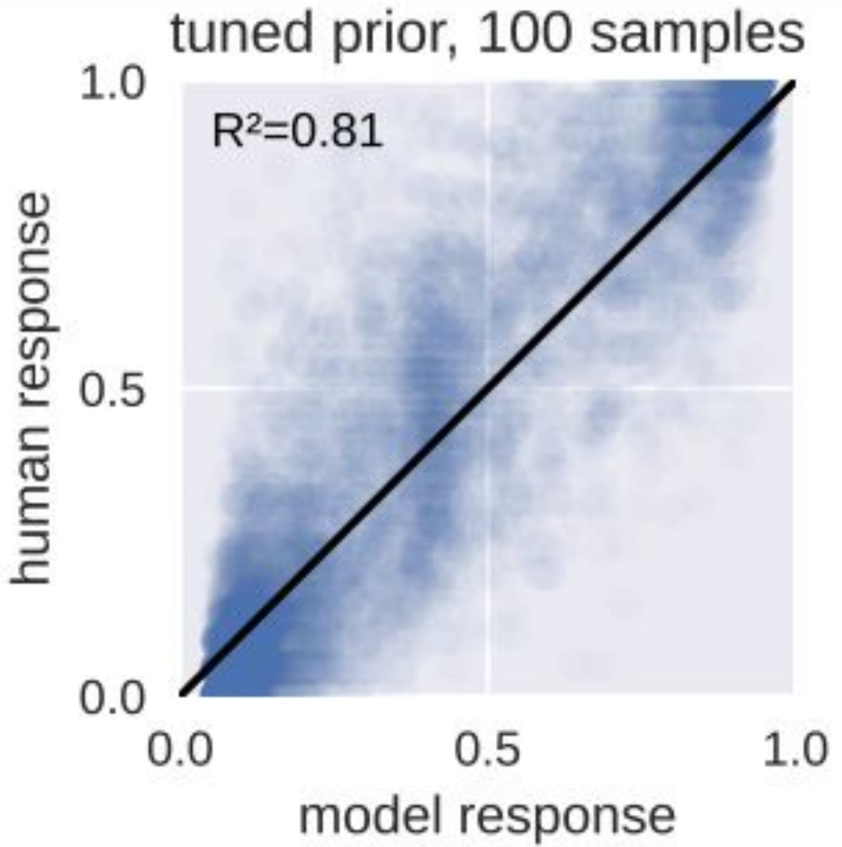
Hidden logical rule x.color = blue ∧ ∀y∈S. x.size≥y.size

Hypothesized Natural language it is the largest blue object in the example

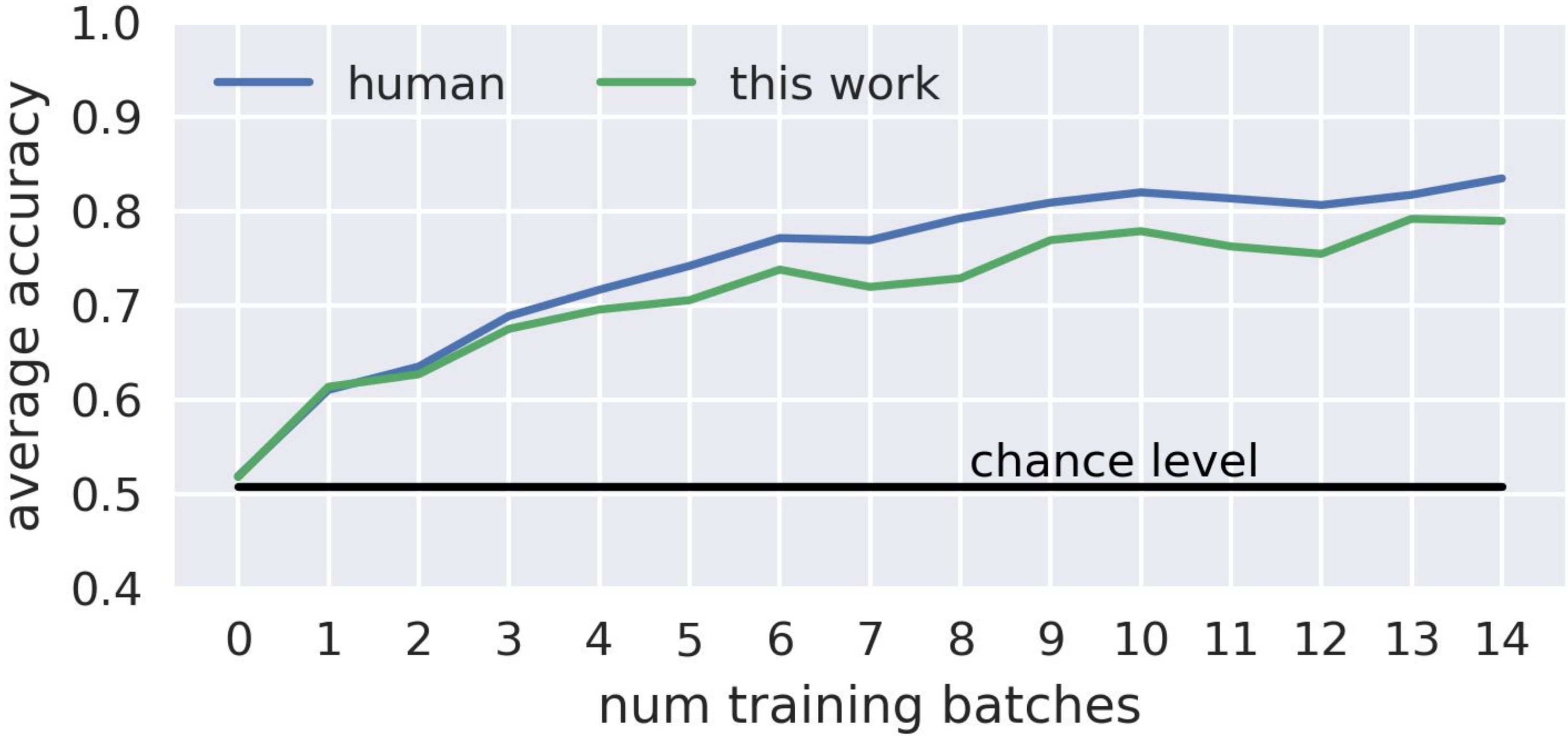


model and humans fail at same point

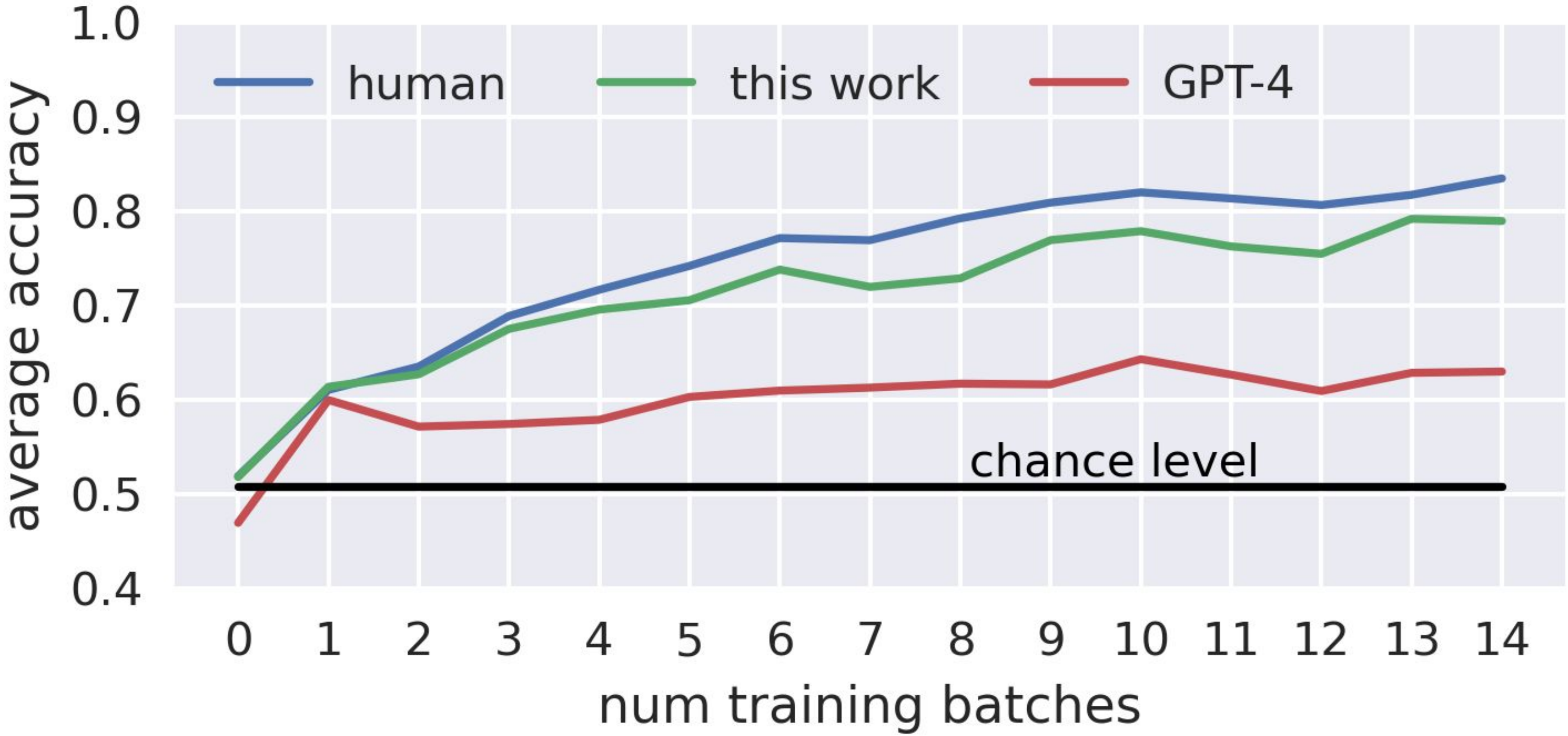
Model is Good but Not Perfect



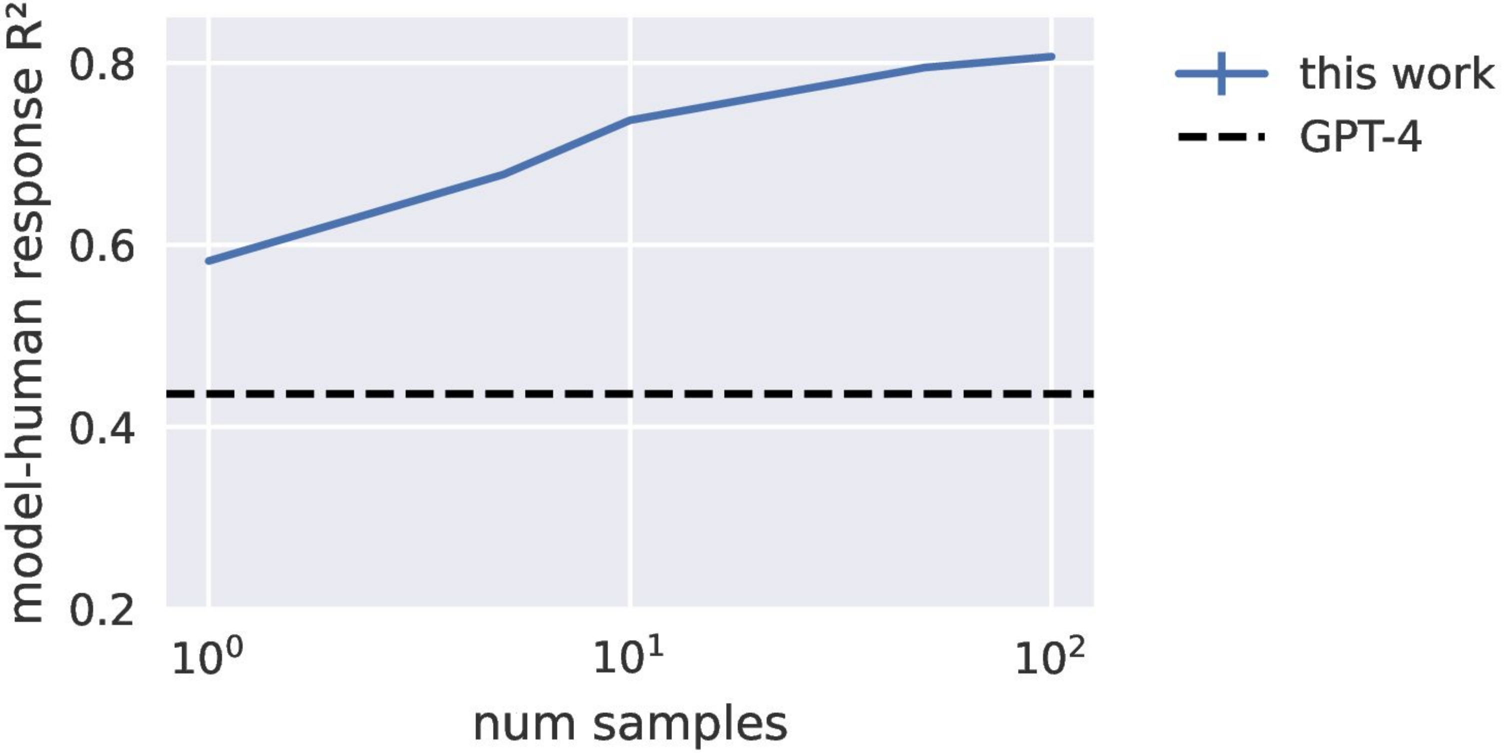
Efficient: Learns from few examples



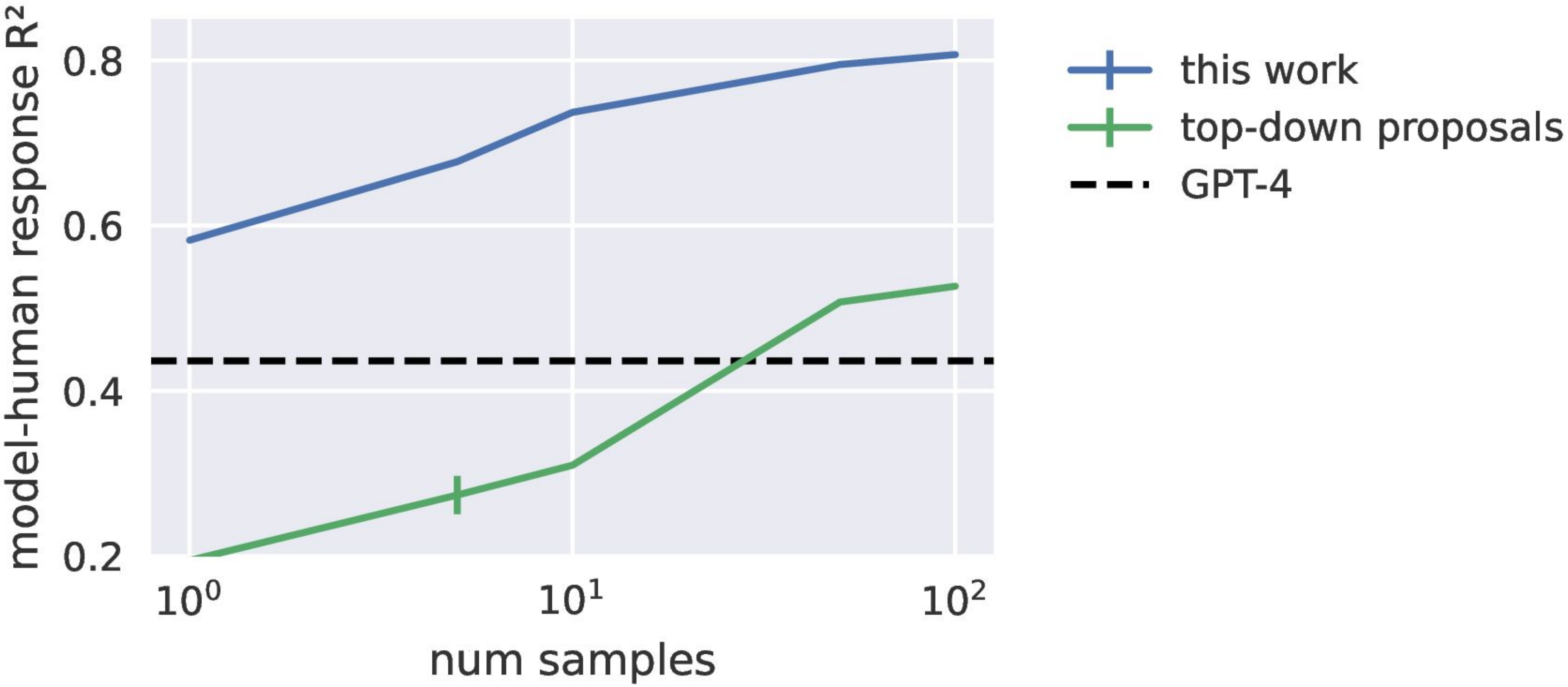
Efficient: Learns from few examples



Efficient: Requires modest compute



Efficient: Requires modest compute



Flexibility test:
Replicating to new subjects on out-of-distribution concepts

Majority color

Minority color

Flexibility test:
Replicating to new subjects on out-of-distribution concepts

Lessons

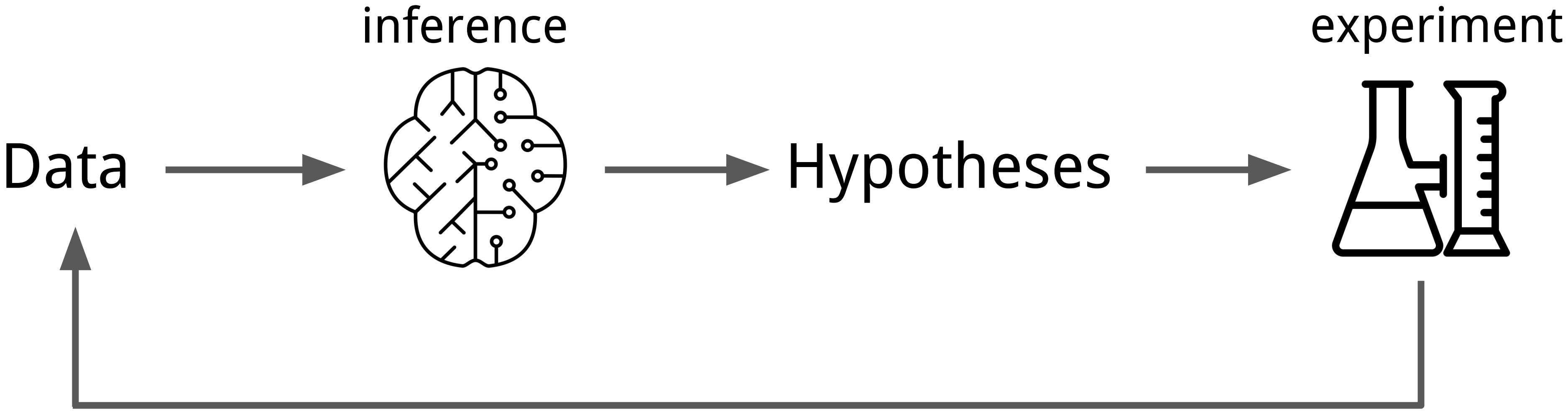
LLMs and the Curse of Compositionality:

Tractably index an infinite concept space w/ finite compute

Neural prior gives good inductive bias

Adding Experimentation and Active Learning

Piriyakulkij et al. NeurIPS '24



Everyday experiments in adulthood:

learning to use new devices, webpages, tools, fixing technical problems

Active learning during childhood development:

exploratory play; active inference during early visual learning

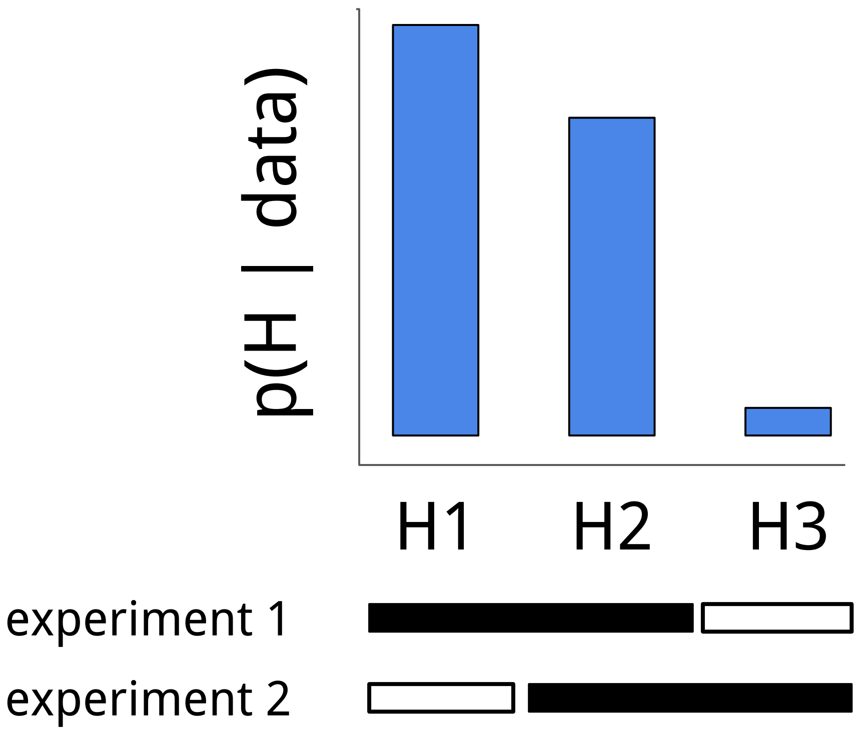
Piriyakulkij et al. NeurIPS ‘24

Probabilistic beliefs are important for active learning

H

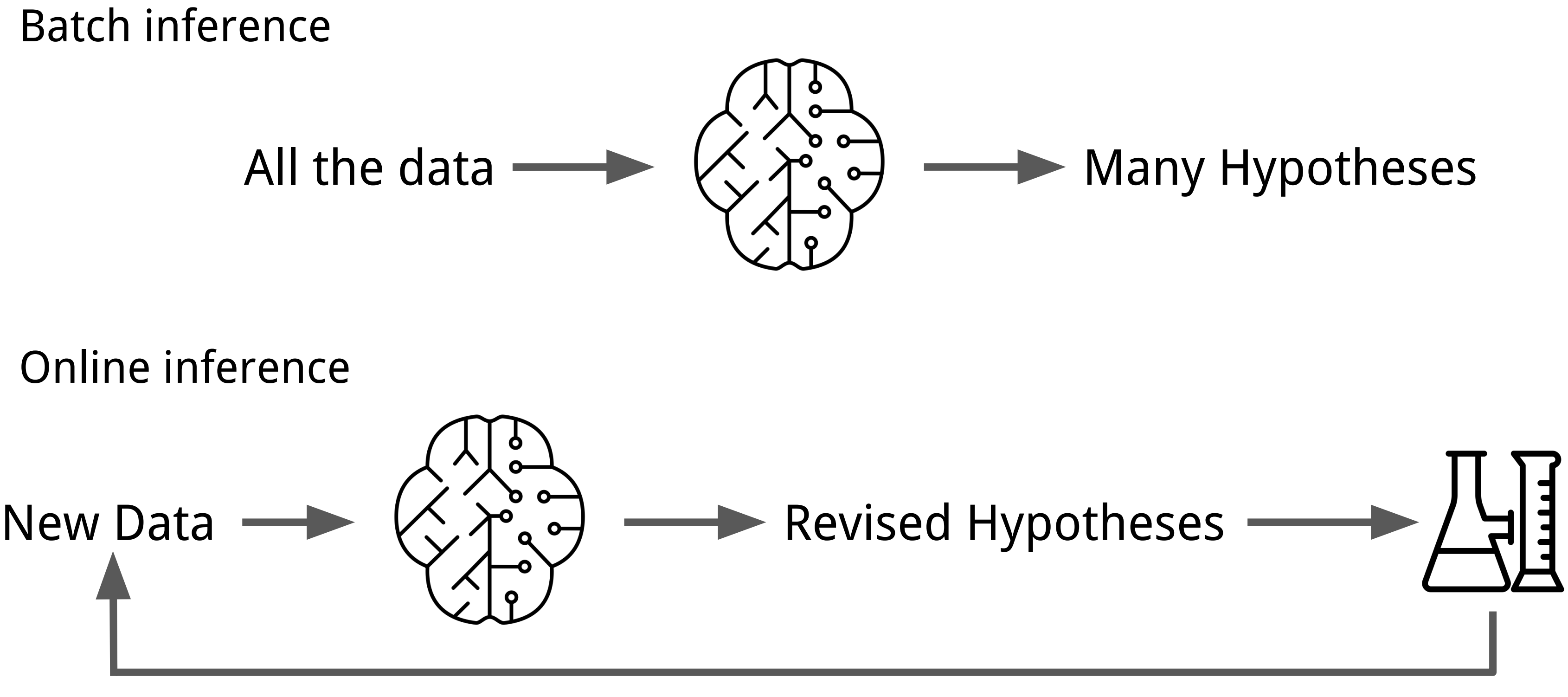
Piriyakulkij et al. NeurIPS '24

Probabilistic beliefs are important for active learning



Piriyakulkij et al. NeurIPS ‘24

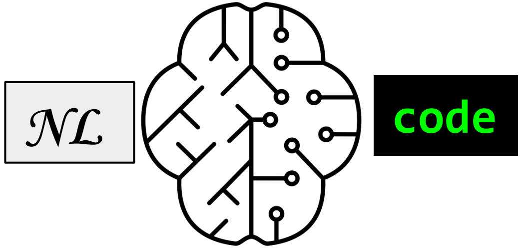
Online inference is important for active learning



Piriyakulkij et al. NeurIPS '24

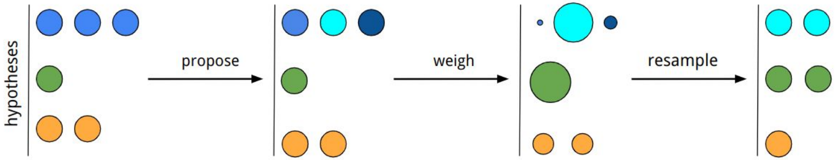
Active Learning Model

Hypothesis Space

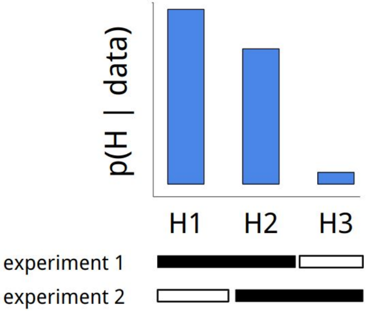


Bayes

Online LLM-guided SMC



Max InfoGain experiments



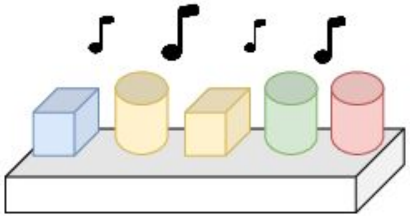
+fuzzy/noisy hypotheses
don't immediately "kill" partly correct proposals

Piriyakulkij et al. NeurIPS '24

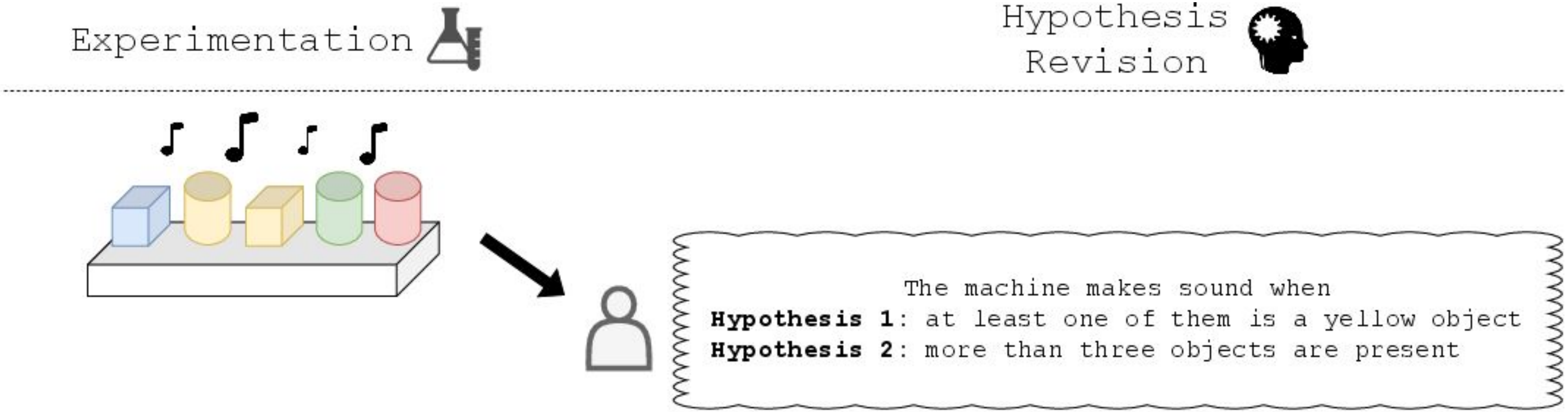
Basic Active Learning Domain:
“Blicket Detectors”

Piriyakulkij et al. NeurIPS ‘24

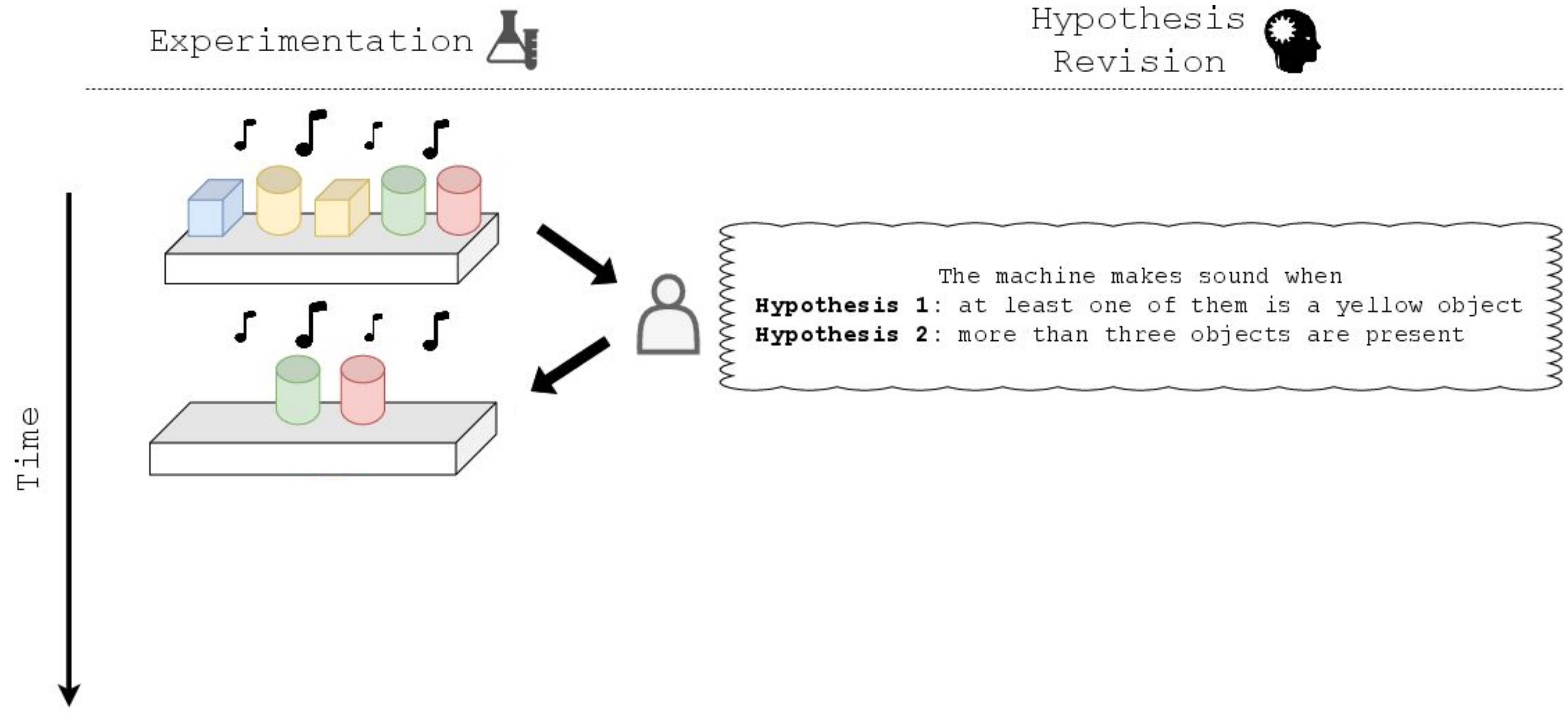
Experimentation



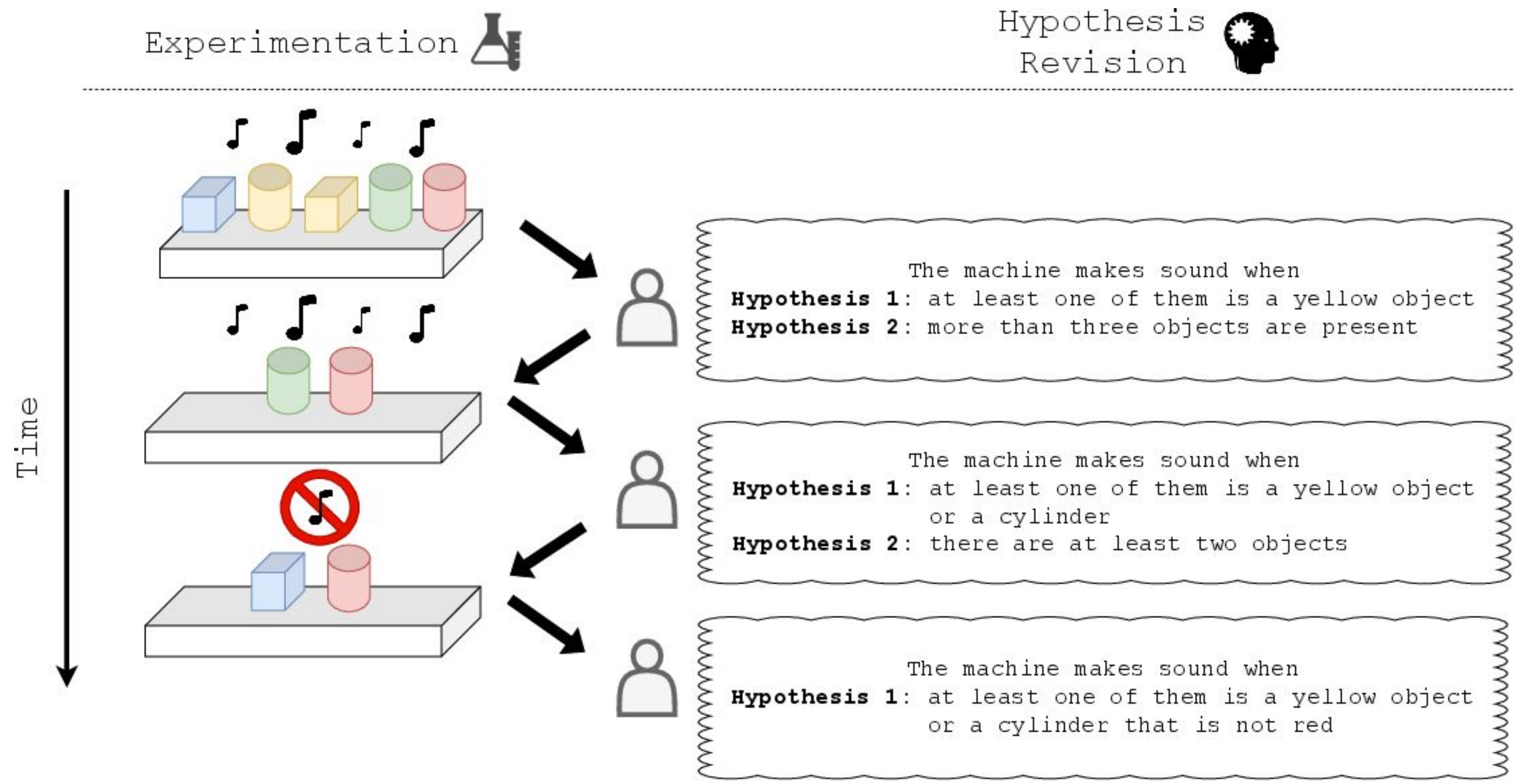
Piriyakulkij et al. NeurIPS ‘24



Piriyakulkij et al. NeurIPS ‘24



Piriyakulkij et al. NeurIPS ‘24

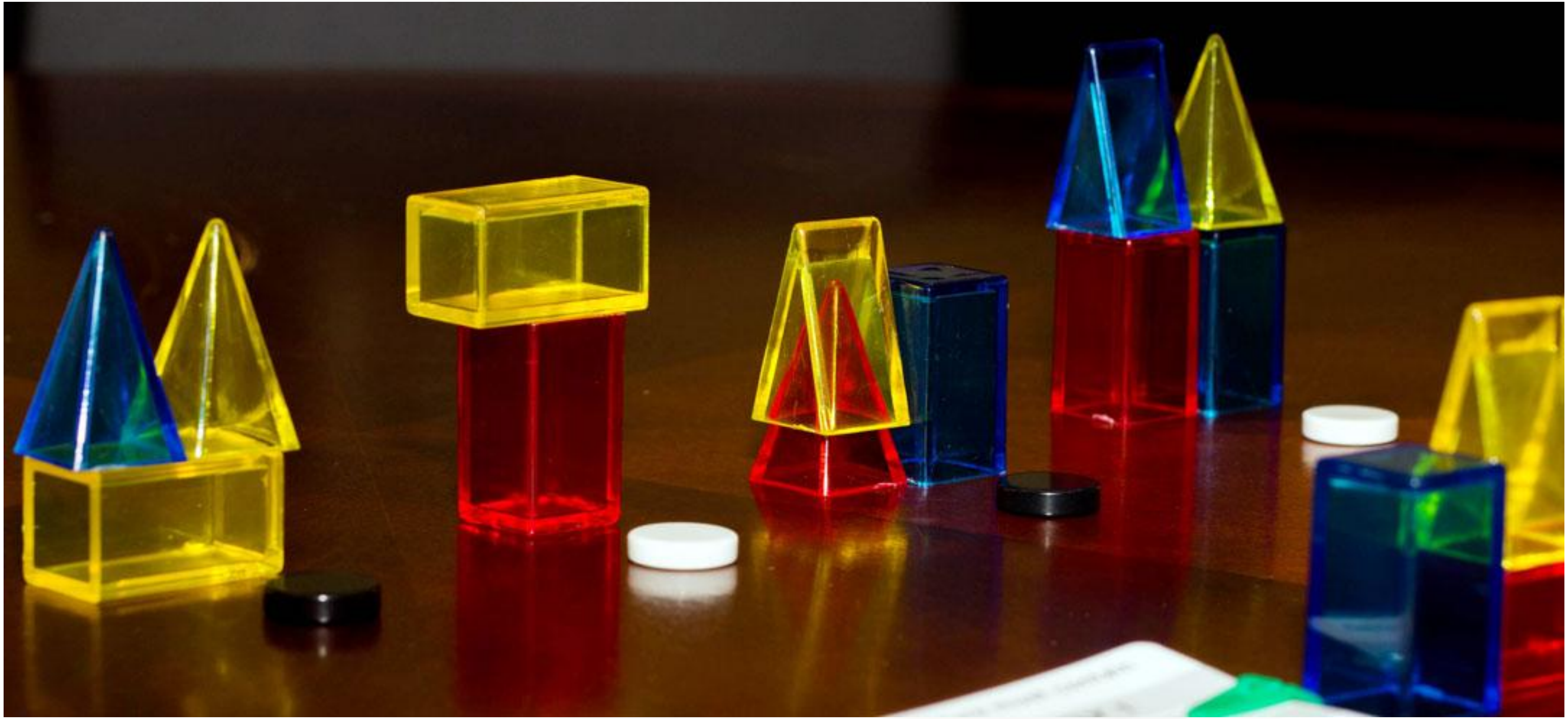


Piriyakulkij et al. NeurIPS ‘24

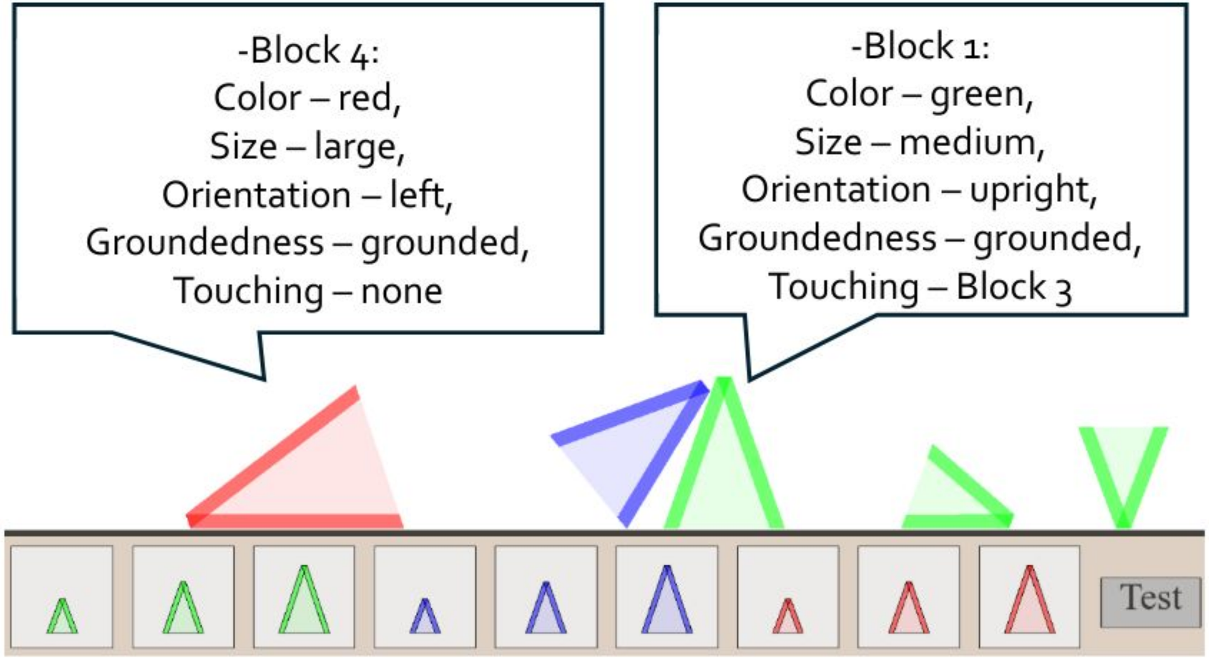
The Original Blicket Detector

https://www.nytimes.com/video/science/100000002187112/buckets-of-blickets-children-and-logic.html

Zendo: harder Blicket-style task

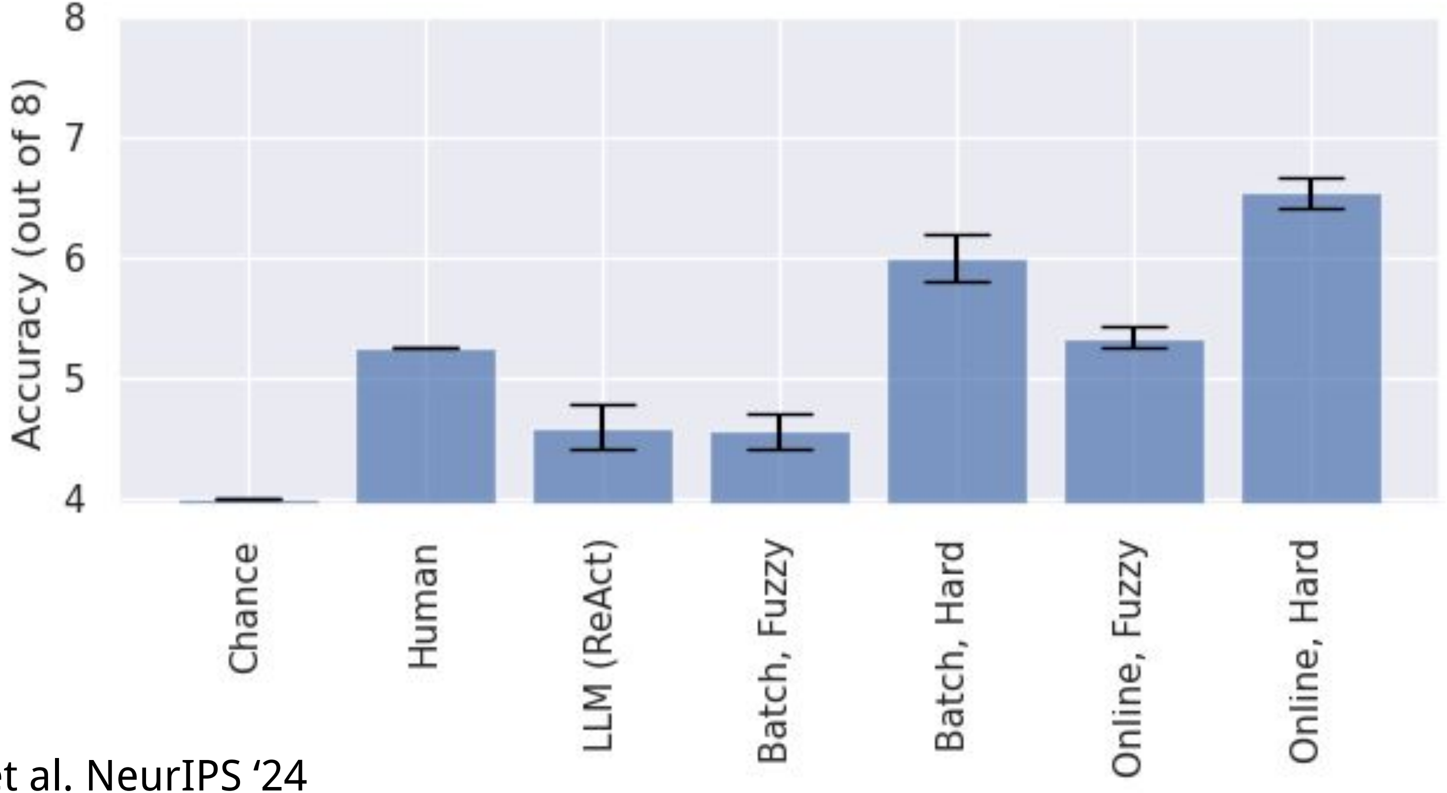


Zendo: harder Blicket-style task



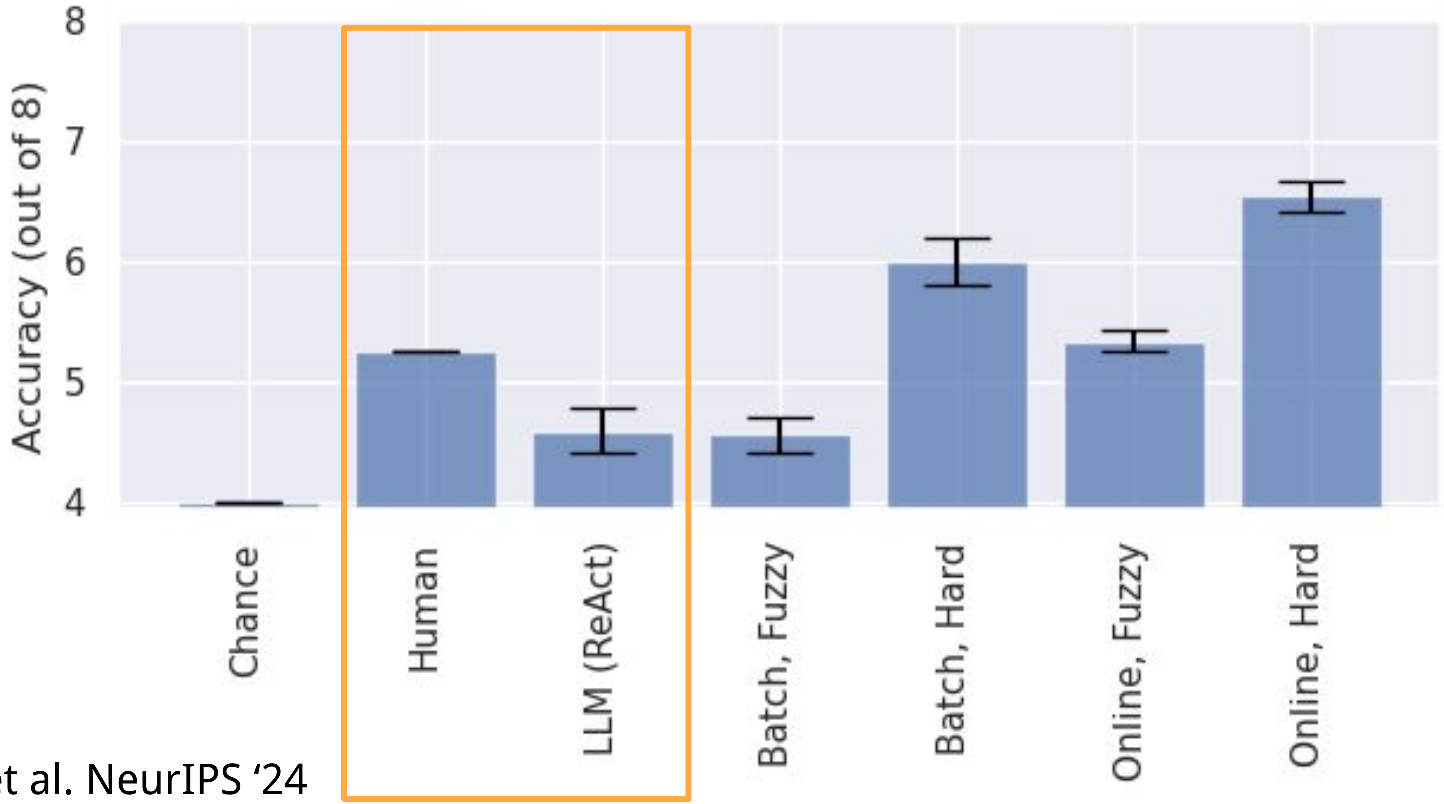
Task & Human Data from Bramley et al. 2018

Zendo: performance



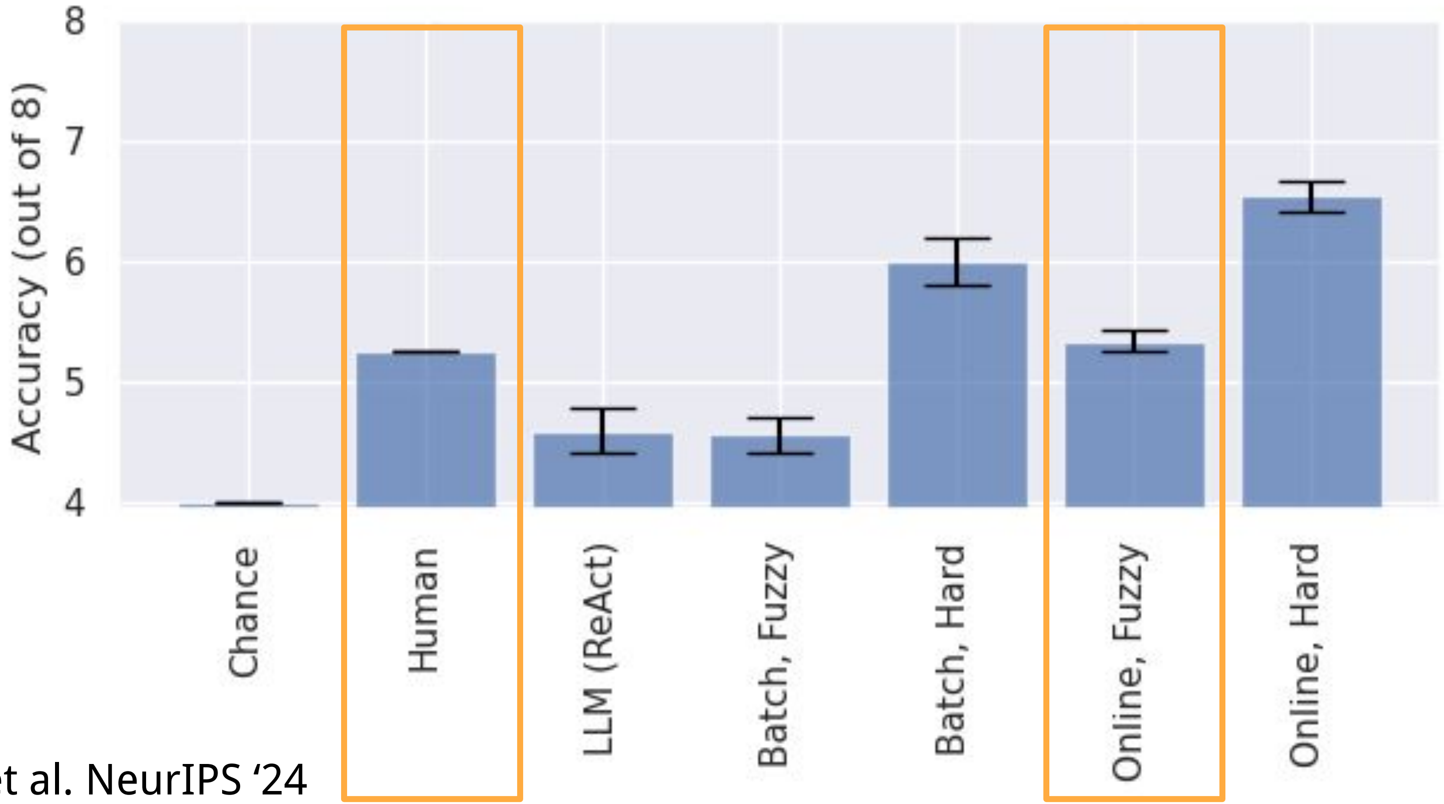
Piriyakulkij et al. NeurIPS ‘24

<Human Level with just prompting an LLM



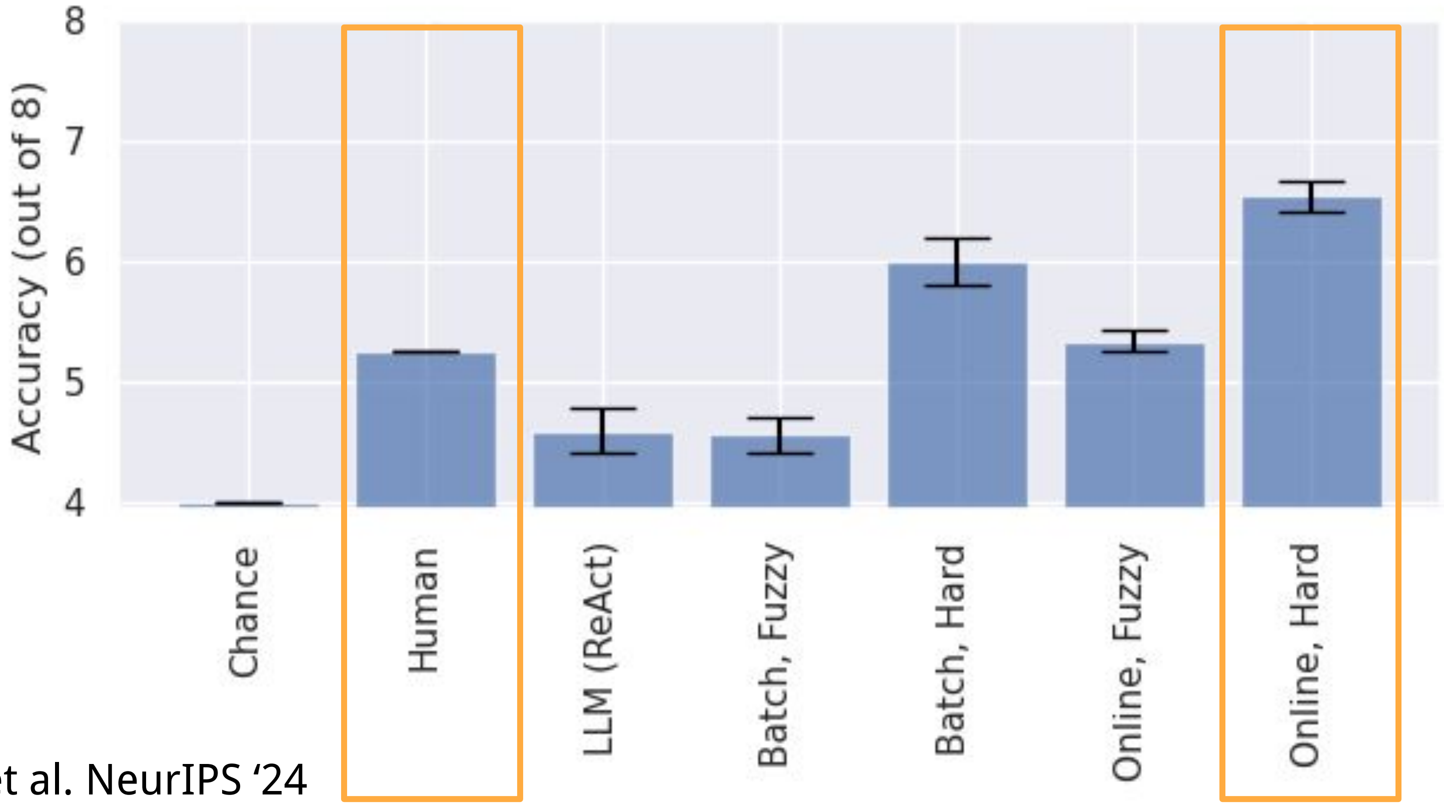
Piriyakulkij et al. NeurIPS ‘24

~Human Level with Fuzzy Probabilistic Rules



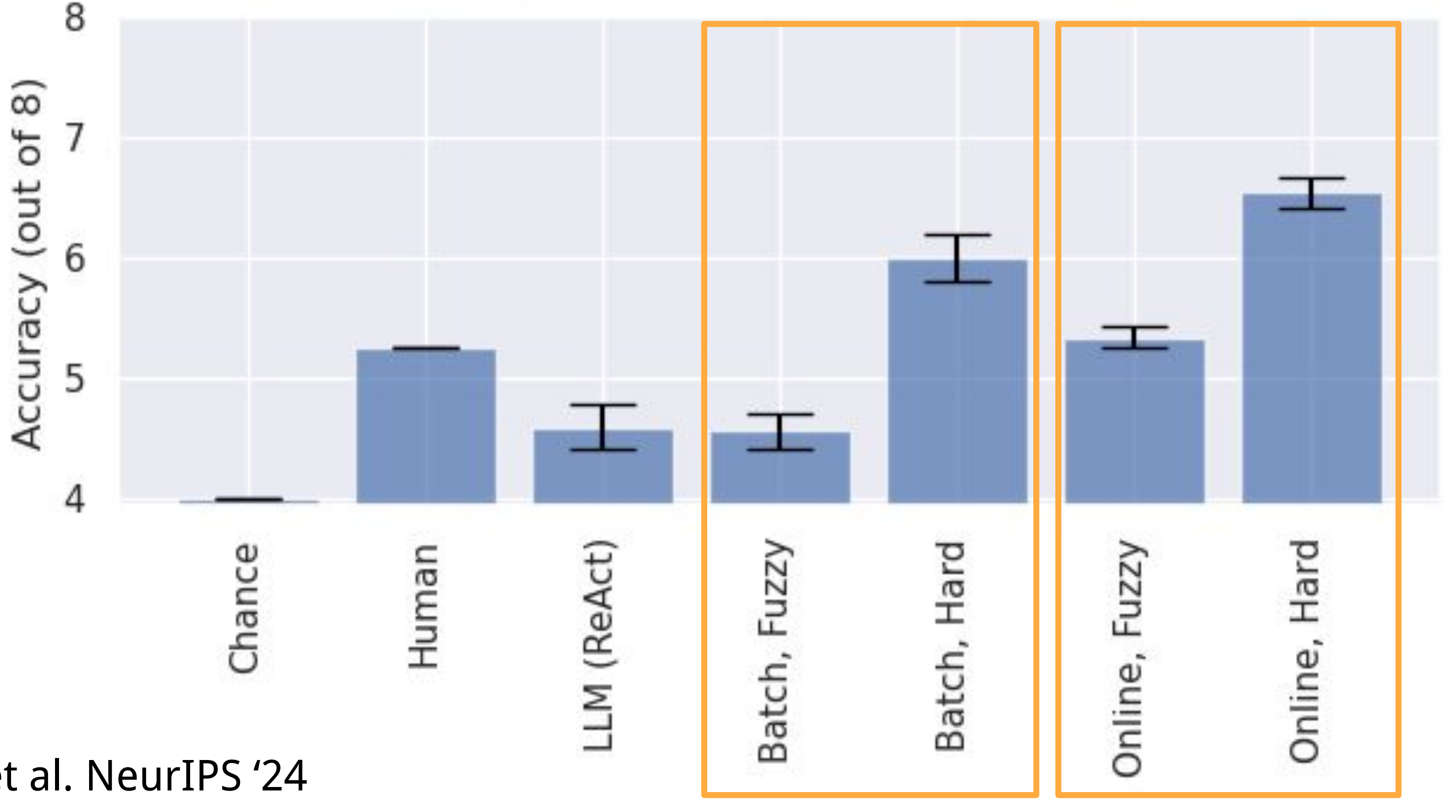
Piriyakulkij et al. NeurIPS ‘24

>Human Level with Deterministic Rules



Piriyakulkij et al. NeurIPS ‘24

Online Inference beats Batch Inference

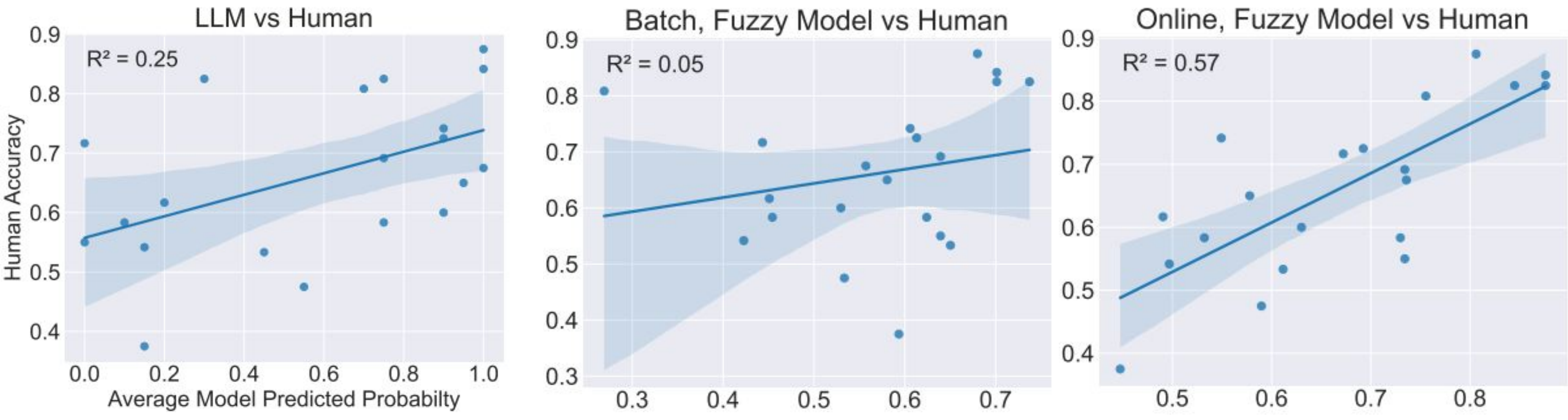


Piriyakulkij et al. NeurIPS ‘24

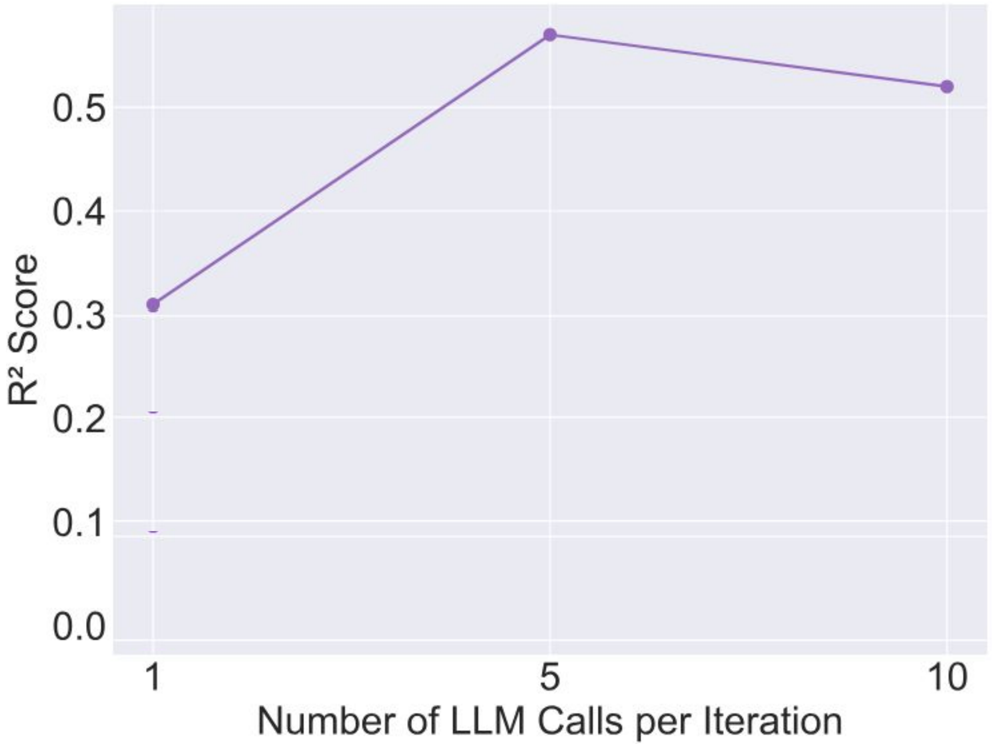
Human-level, Not quite Human-like

Human-level, Not quite Human-like, but
Online + fuzzy rules best predicts human responses

Human-level, Not quite Human-like, but Online + fuzzy rules best predicts human responses



Bounded Rationality:
Human-Model fit degrades with enough compute budget



*responses binned by
problem and ground-truth label

Piriyakulkij et al. NeurIPS '24

Lessons

Probability important for picking good experiments

Online inference is more effective, and more humanlike

See Top Piriyakulkij & Cassidy Lagenfeld’s NeurIPS ‘24 paper

Why are these models human-like?

Because they approximate rational inference over expressive, flexible representations

NOT because of LLMs: they're just proposal distributions

LLMs "just" give you tractable inference in expressive symbolic representations

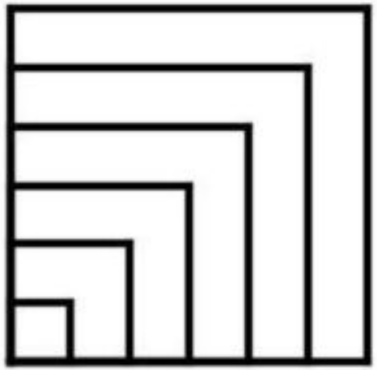
Part 2:

Engineering Program Learners:

Program Induction in New Domains

What if your pretrained model can’t propose good programs?

provided example



generated program

```
for i in range(7):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)
```

Li & Ellis, NeurIPS ‘24

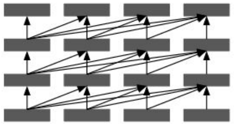
Finetune for program induction?

Where does the data come from?

Finetune on Synthetic Data

Human-Written Code

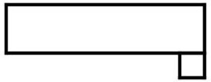
```
# a 9-pointed star
for i in range(9):
    forward(16)
    left(180.0 - 40.0)

# 4 concentric squares
for i in range(5):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)

# <dozens of examples>
```



LLM-Written Remix

```
# 5 rectangle perimeter
with a long dash and a
small background color
rectangle
for i in range(5):
    forward(2)
    left(90.0)
penup()
forward(2)
left(0.0)
pendown()
for i in range(2):
    forward(4)
    left(90.0)
    forward(16)
    left(90.0)
```

Code Output

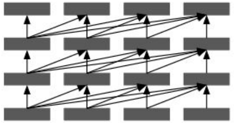


Li & Ellis, NeurIPS ‘24

Finetune on Synthetic Data

Human-Written Code

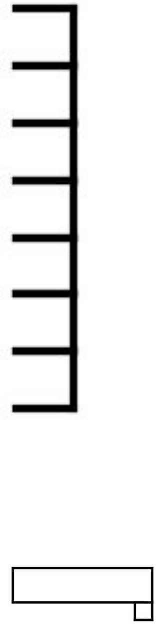
```
# a 9-pointed star
for i in range(9):
    forward(16)
    left(180.0 - 40.0)

# 4 concentric squares
for i in range(5):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)

# <dozens of examples>
```



LLM-Written Remix

```
# a spiral staircase
for i in range(7):
    forward(2)
    left(90.0)

    forward(2)
    left(90.0)

    forward(2)
    left(180.0)
```

Code Output

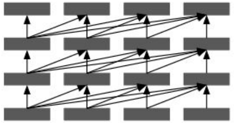


Li & Ellis, NeurIPS ‘24

Finetune on Synthetic Data

Human-Written Code

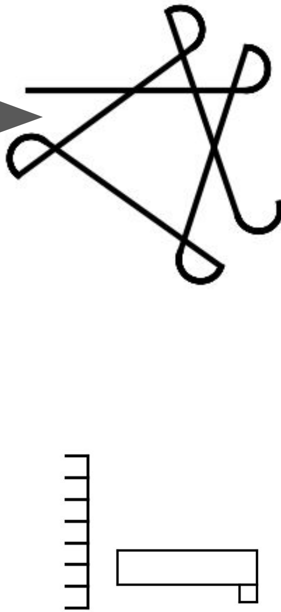
```
# a 9-pointed star
for i in range(9):
    forward(16)
    left(180.0 - 40.0)

# 4 concentric squares
for i in range(5):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)

# <dozens of examples>
```



LLM-Written Remix

```
# 5 sided snowflake with a
medium line and a small
semicircle as arms
for j in range(5):
  forward(10)
  for i in range(HALF_INF):
      forward(EPS_DIST*1)
      left(EPS_ANGLE)
   forward(0)
   left(72.0)
```

Code Output

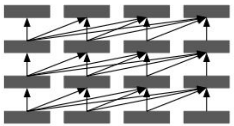


Li & Ellis, NeurIPS ‘24

Finetune on Synthetic Data

Human-Written Code

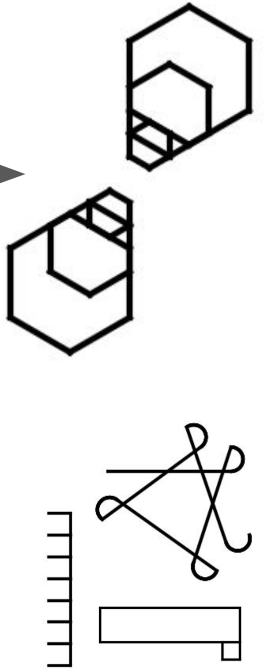
```
# a 9-pointed star
for i in range(9):
    forward(16)
    left(180.0 - 40.0)

# 4 concentric squares
for i in range(5):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)

# <dozens of examples>
```



LLM-Written Remix

```
# series of increasingly
rotated hexagonal shapes
for i in range(1, 8):
    for j in range(6):
        forward(4-i)
        left(60.0)
    penup()
    forward(2)
    pendown()
```

Code Output



Li & Ellis, NeurIPS ‘24

Finetune on Synthetic Data

Human-Written Code

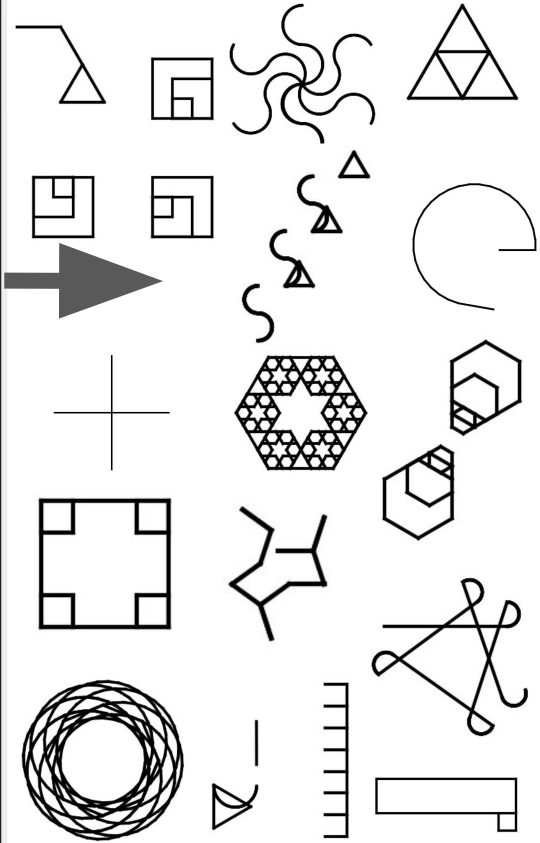
```
# a 9-pointed star
for i in range(9):
    forward(16)
    left(180.0 - 40.0)

# 4 concentric squares
for i in range(5):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)

# <dozens of examples>
```

Li & Ellis, NeurIPS ‘24

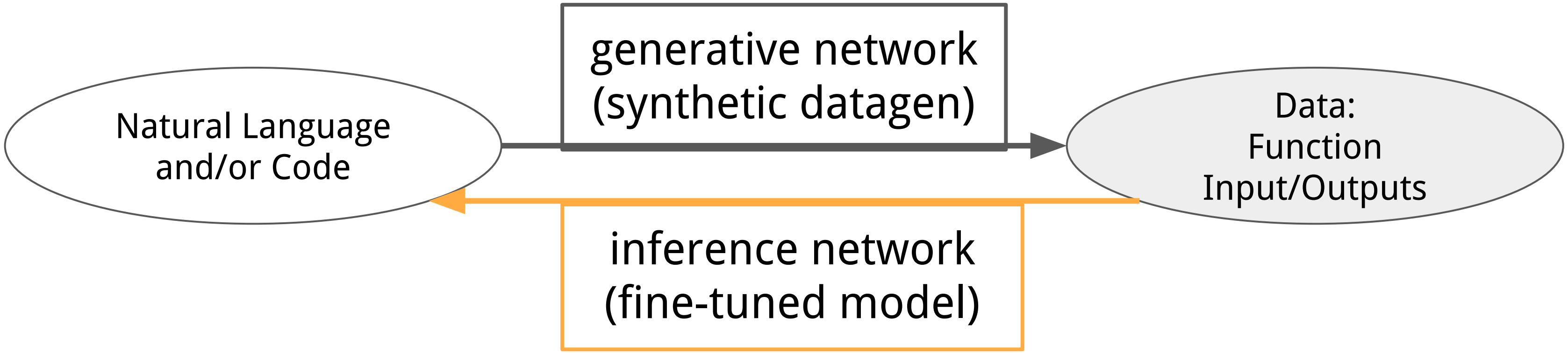
LLM-Written Remix

```
# 5 rectangle perimeter with
a long dash and a small
background color rectangle
for i in range(5):
    forward(2)
    left(90.0)
penup()
forward(2)
left(0.0)
pendown()
for i in range(2):
    forward(4)
    left(90.0)
    forward(16)
    left(90.0)
```

Code Outputs



Wake-Sleep

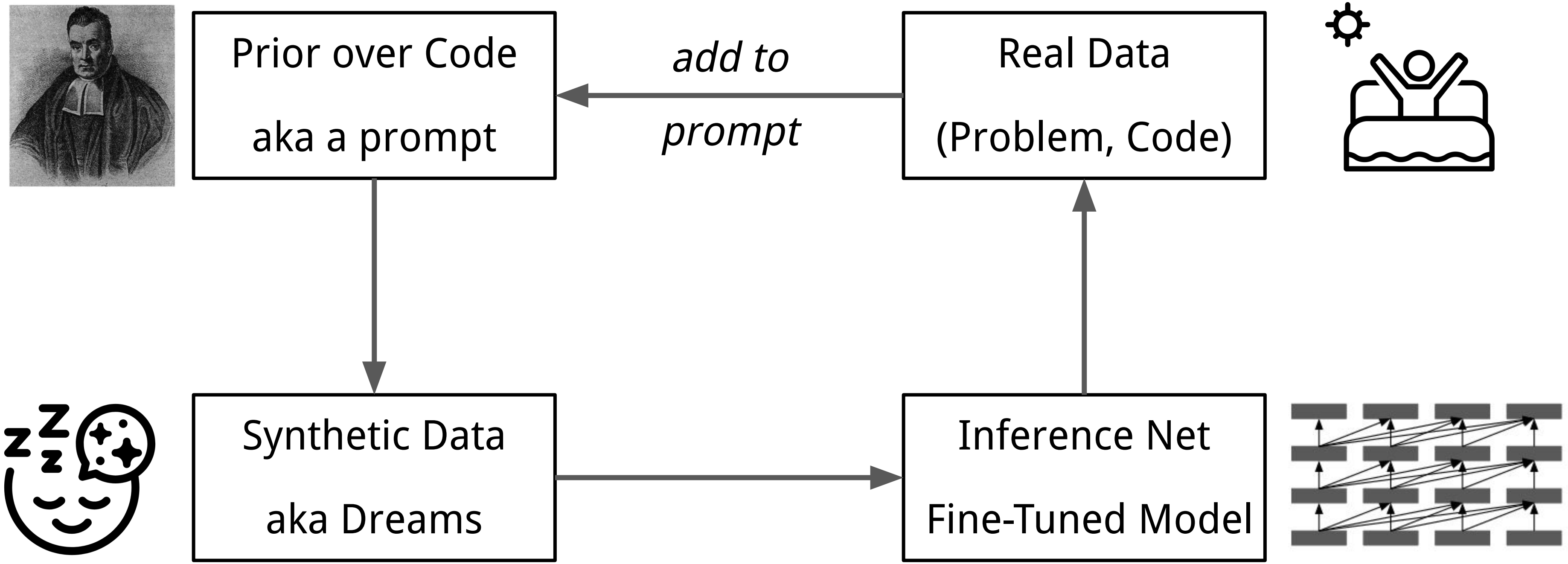


Two models that train each other:

Generative path makes synthetic data

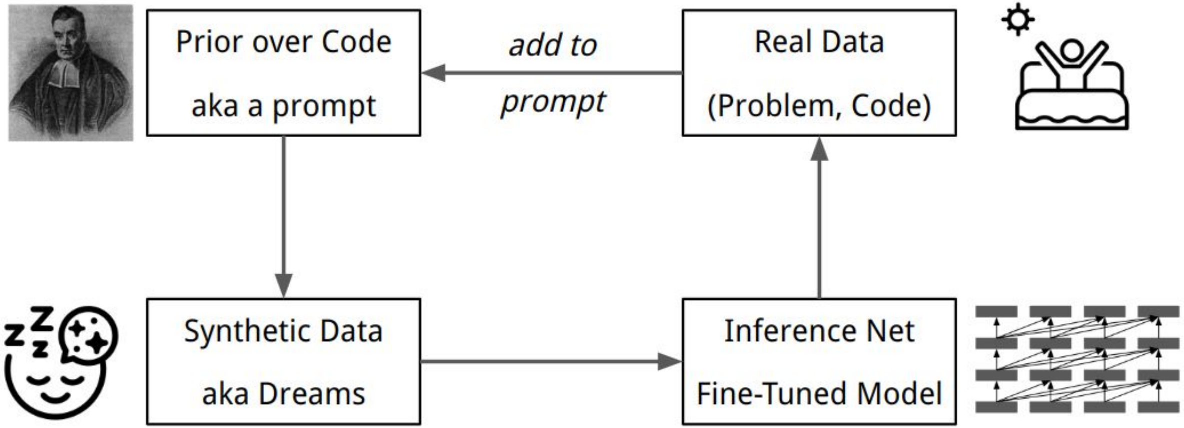
Inference network updates prompt for generative path

Wake-Sleep Fine-Tuning



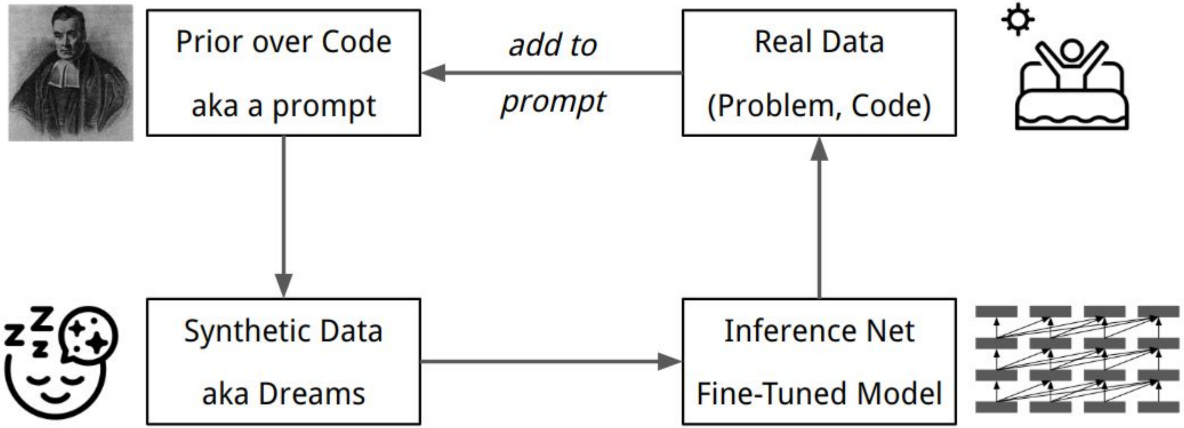
Data Efficient:

Needs relatively little non-synthetic data



But needs to be “warm started” with a good prompt / prior

Otherwise, might get no learning signal

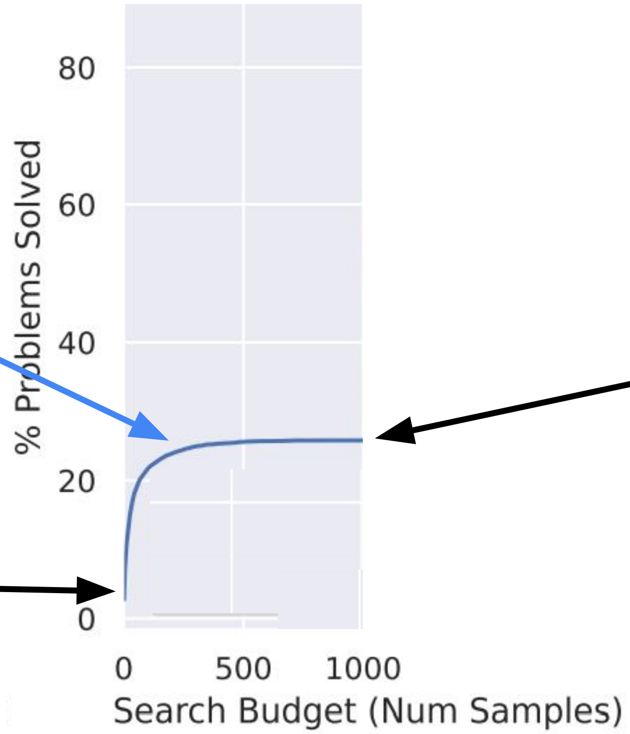


Wake-Sleep Fine-Tuning

1st wake phase

Bad initial prior:
Only short programs

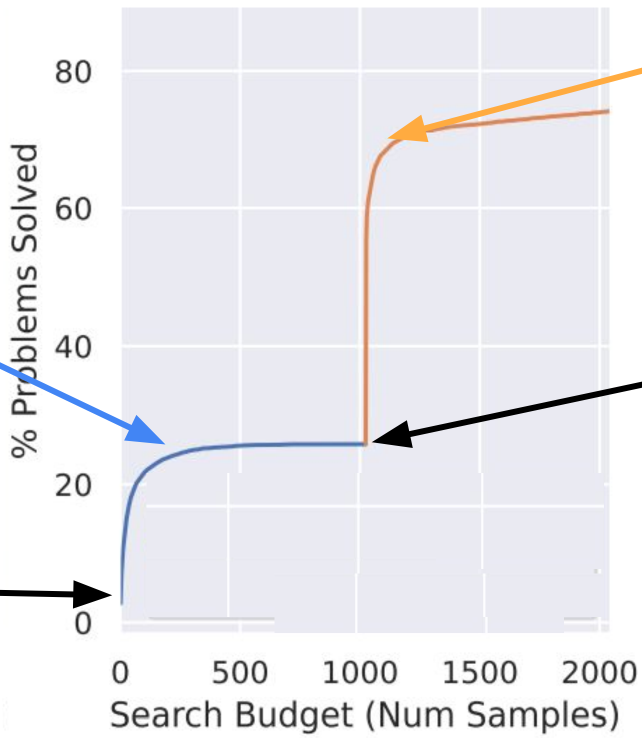
1st sleep phase



2nd sleep phase

Li & Ellis, NeurIPS ‘24

Wake-Sleep Fine-Tuning



2nd wake phase

1st wake phase

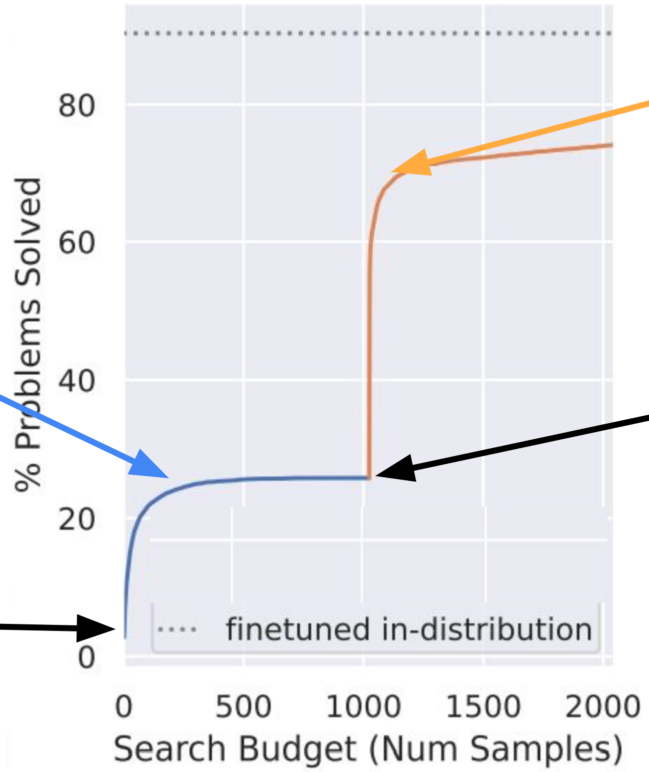
2nd sleep phase

Bad initial prior:
Only short programs

1st sleep phase

Li & Ellis, NeurIPS ‘24

Wake-Sleep Fine-Tuning



2nd wake phase

1st wake phase

2nd sleep phase

Bad initial prior:
Only short programs

1st sleep phase

Why Wake-Sleep

Fine-tuning on synthetic data is conceptually simple [self-instruct]

Why make things complicated?

- Might not know the distribution of programs we care about
- Connections to biological learning
  - Learning to be a good Bayesian over the timespan of an individual lifetime
  - Cf. Tom Griffith’s talk: learning to be Bayesian via evolution

Li & Ellis, NeurIPS ‘24

DOMAIN: lists

provided examples

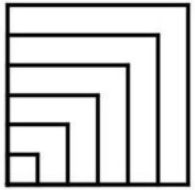
INPUT	OUTPUT
4,2,8	2,0,6
9,9,9,9	0,0,0,0
-7,0,2	0,7,9

generated program

```
# Check if the list is empty
if not input_list:
    return input_list
# Find min number in the list
min_num = min(input_list)
# Subtract from each element
return [num - min_num
        for num in input_list]
```

DOMAIN: graphics

provided example



generated program

```
for i in range(7):
    with fork_state():
        for j in range(4):
            forward(2*i)
            left(90.0)
```

DOMAIN: text editing macros

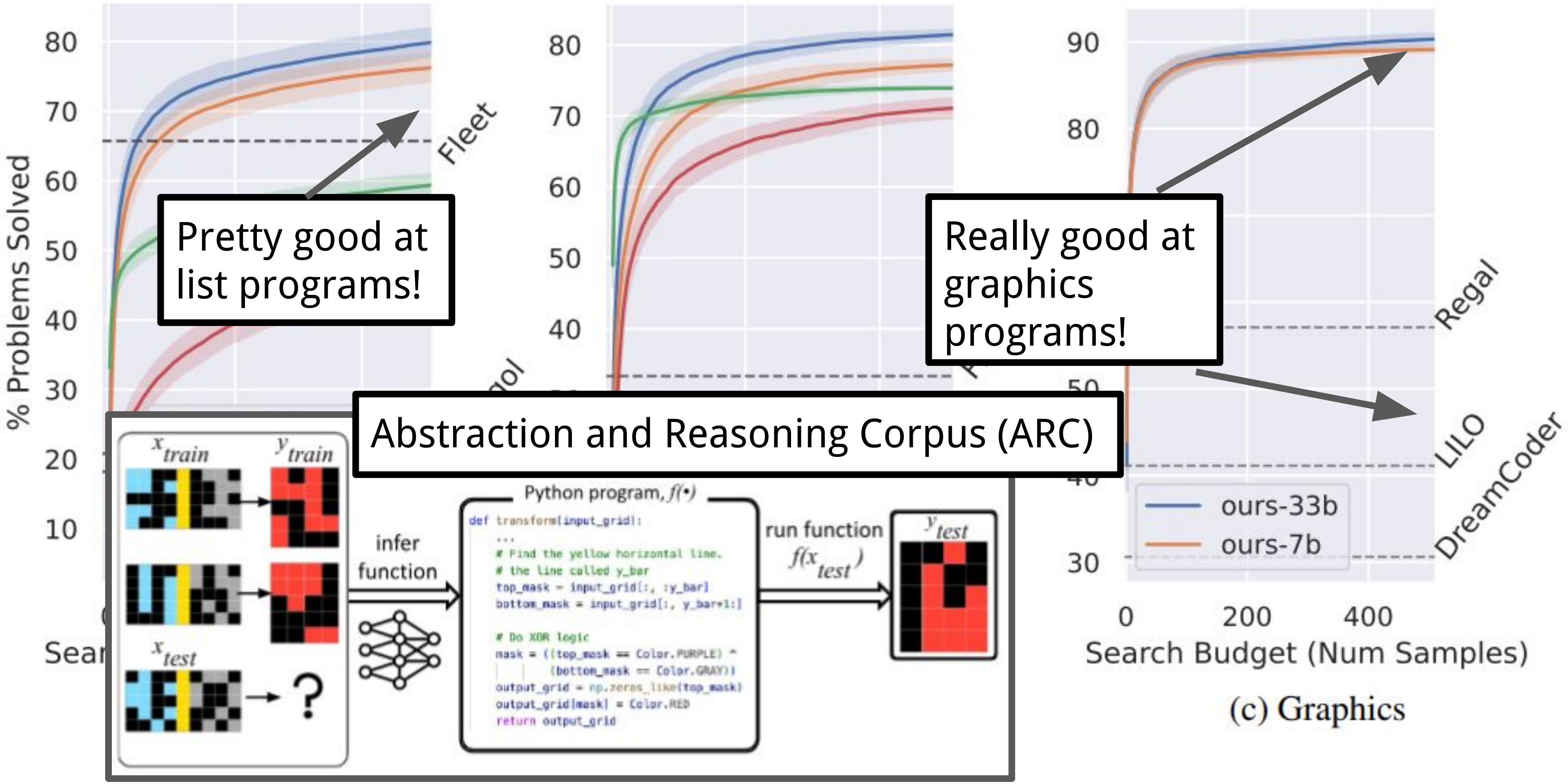
provided examples

INPUT	OUTPUT
18:25:57	6PM to 8PM
21:44:40	8PM to 10PM
07:00:20	6AM to 8AM
23:34:17	10PM to 12AM

generated program

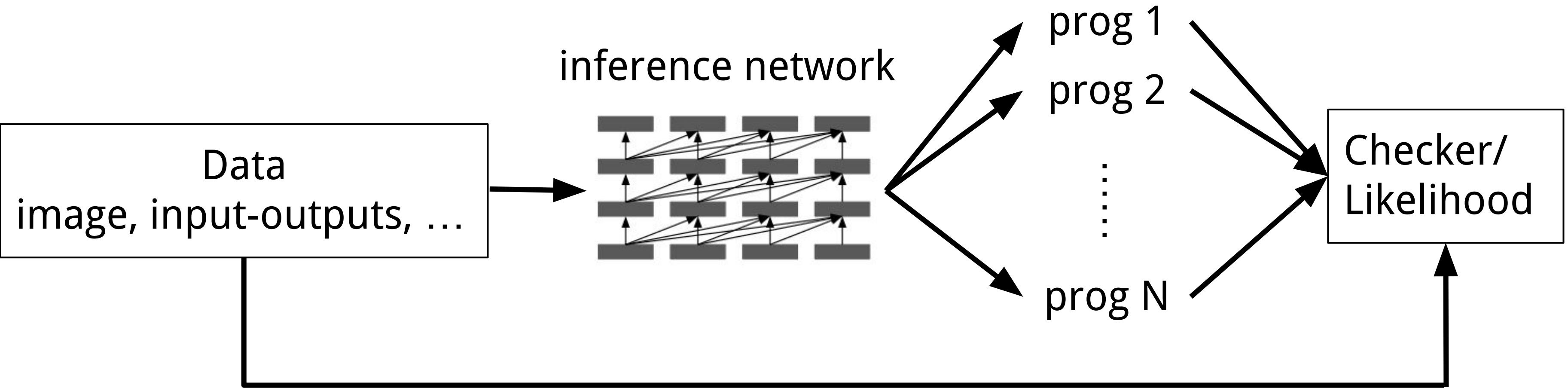
```
original_time = datetime.strptime(input_str, '%H:%M:%S')
hour = original_time.hour
start_hour = hour - (hour % 2)
end_hour = start_hour + 2
start_hour_12 = start_hour % 12 or 12
end_hour_12 = end_hour % 12 or 12
start_ampm = "AM" if start_hour < 12 else "PM"
end_ampm = "AM" if end_hour < 12 or end_hour == 24 else "PM"
return f"{start_hour_12}{start_ampm} to {end_hour_12}{end_ampm}"
```

Li & Ellis, NeurIPS '24



Li & Ellis, NeurIPS '24

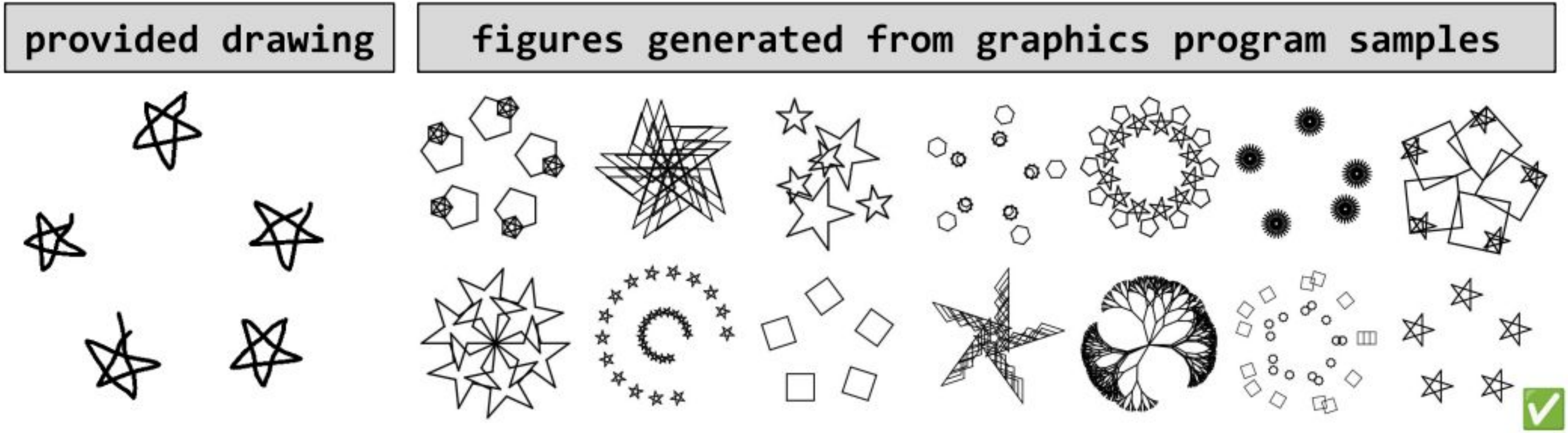
Caveat: Need to be about to check if an answer is correct



Caveat: Need to be about to check if an answer is correct

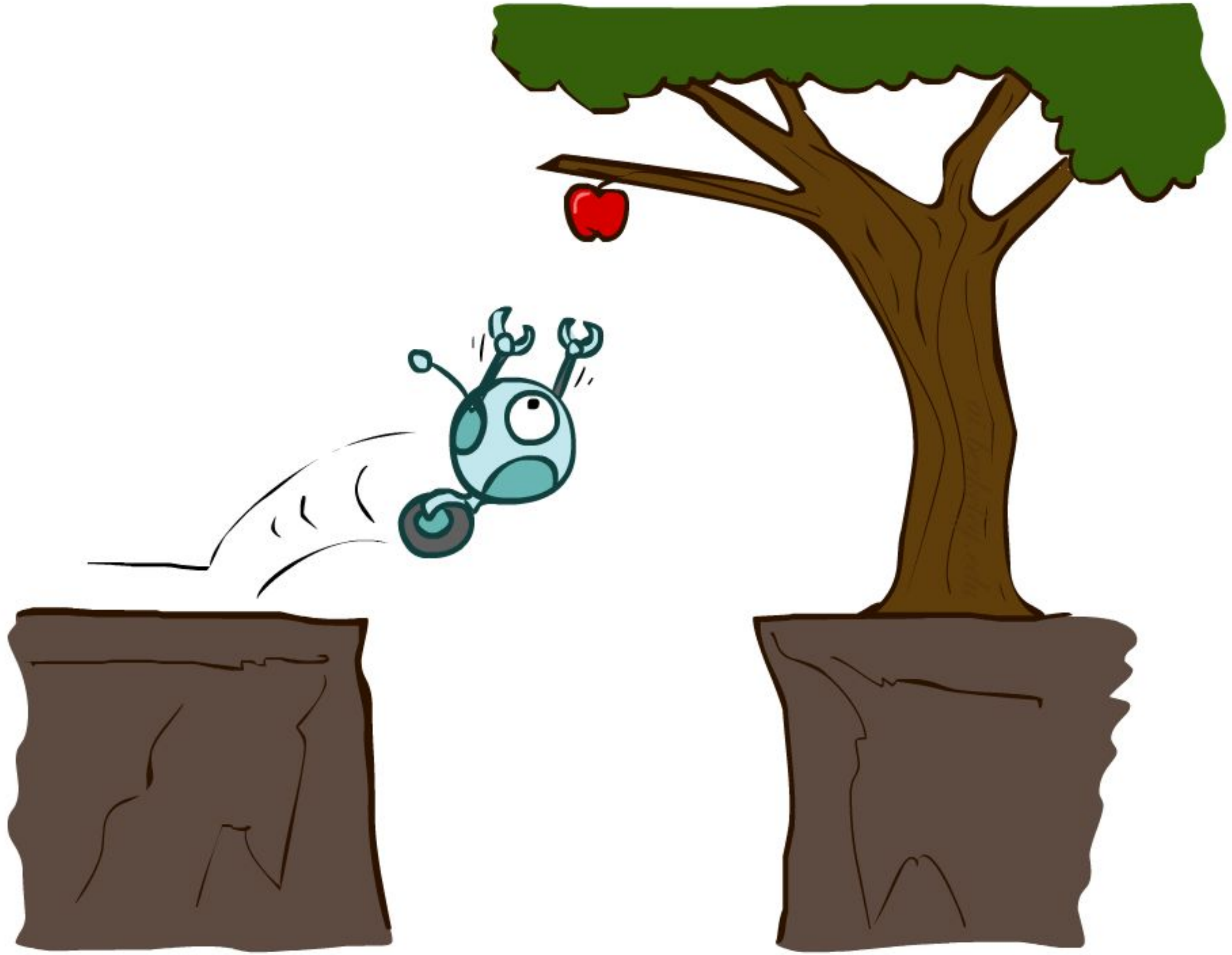
provided drawing

Caveat: Need to be about to check if an answer is correct



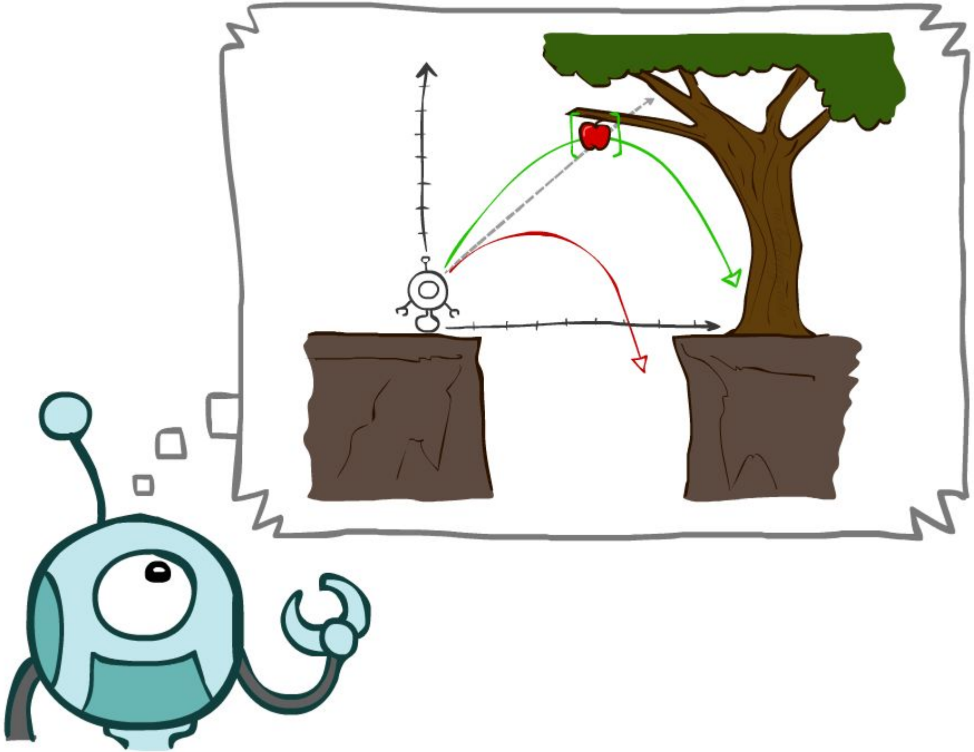
From programs that describe images,

to programs that describe how the world works



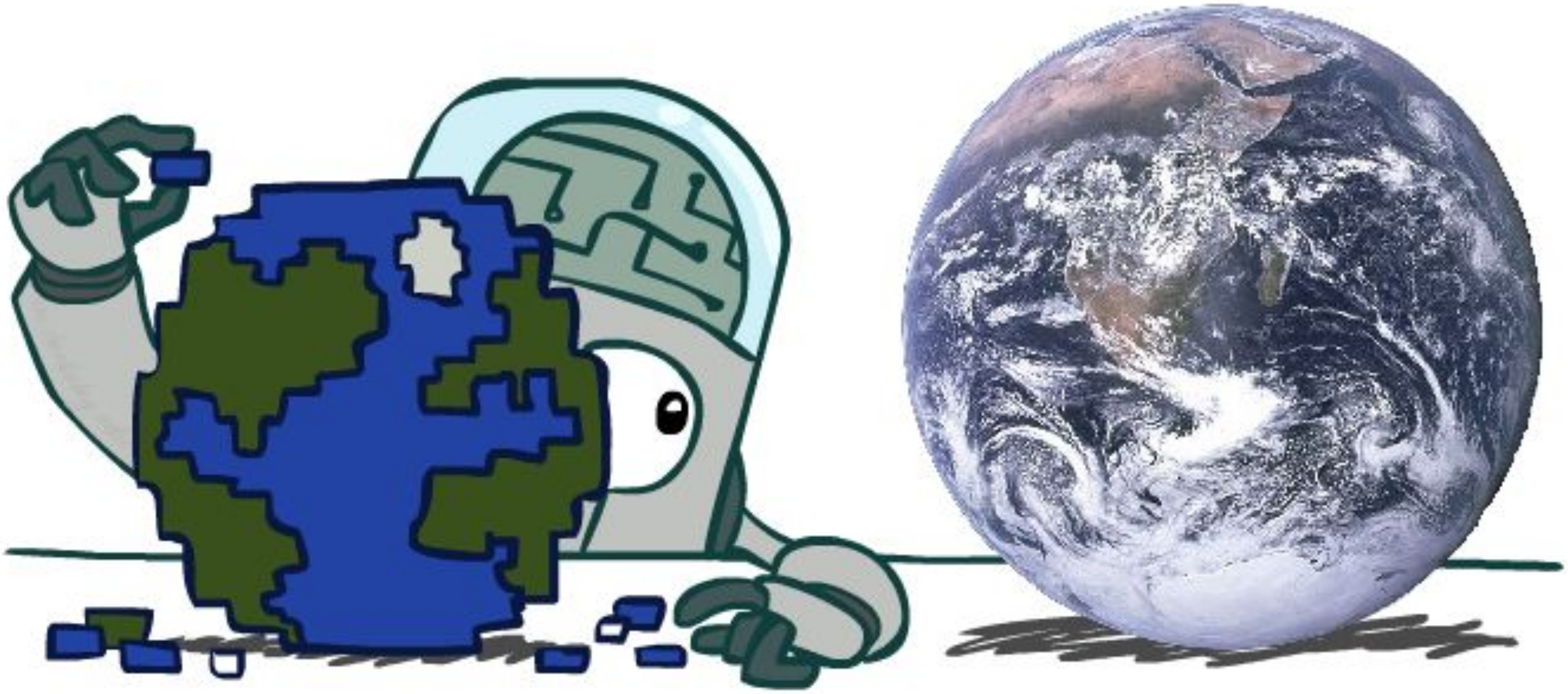
Picture credit:
Berkeley CS188

World Models allow imagining the future



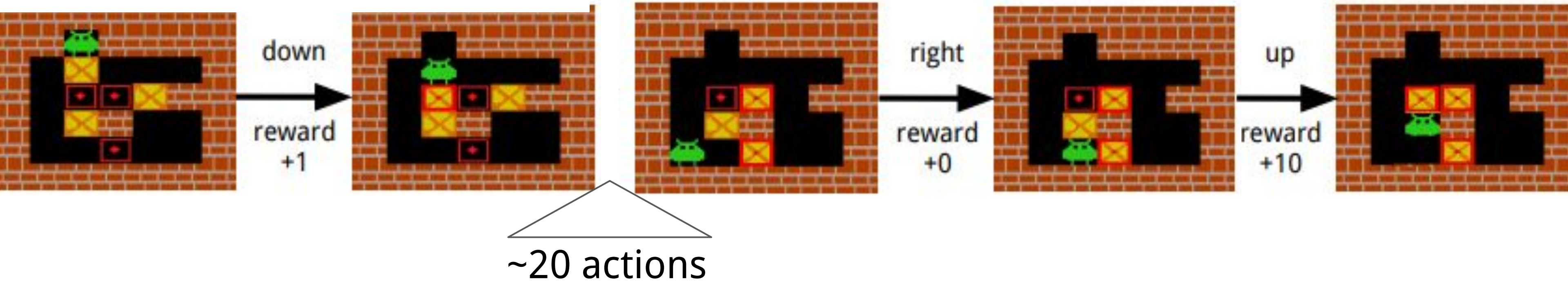
Picture credit:
Berkeley CS188

World Models should be learned

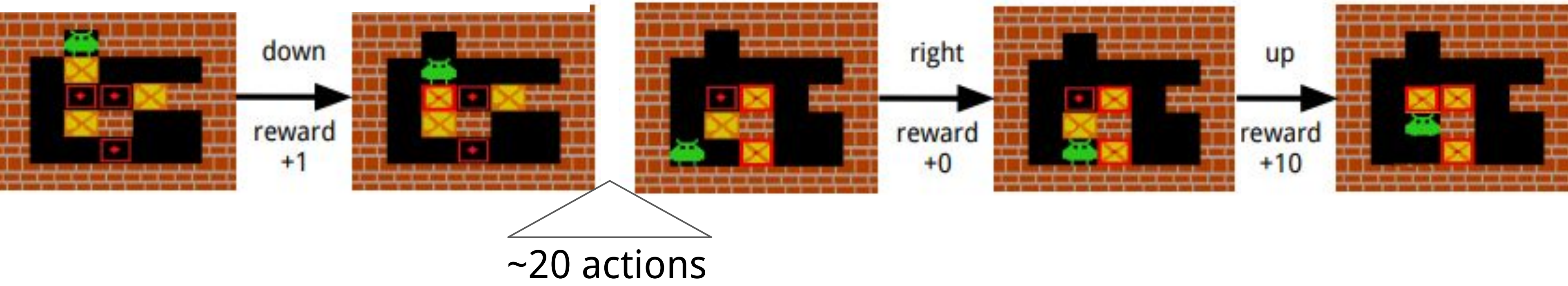


Picture credit:
Berkeley CS188

WorldModel : (State, Action)⟶(NewState, Reward)



WorldModel : (State, Action)⟶(NewState, Reward)

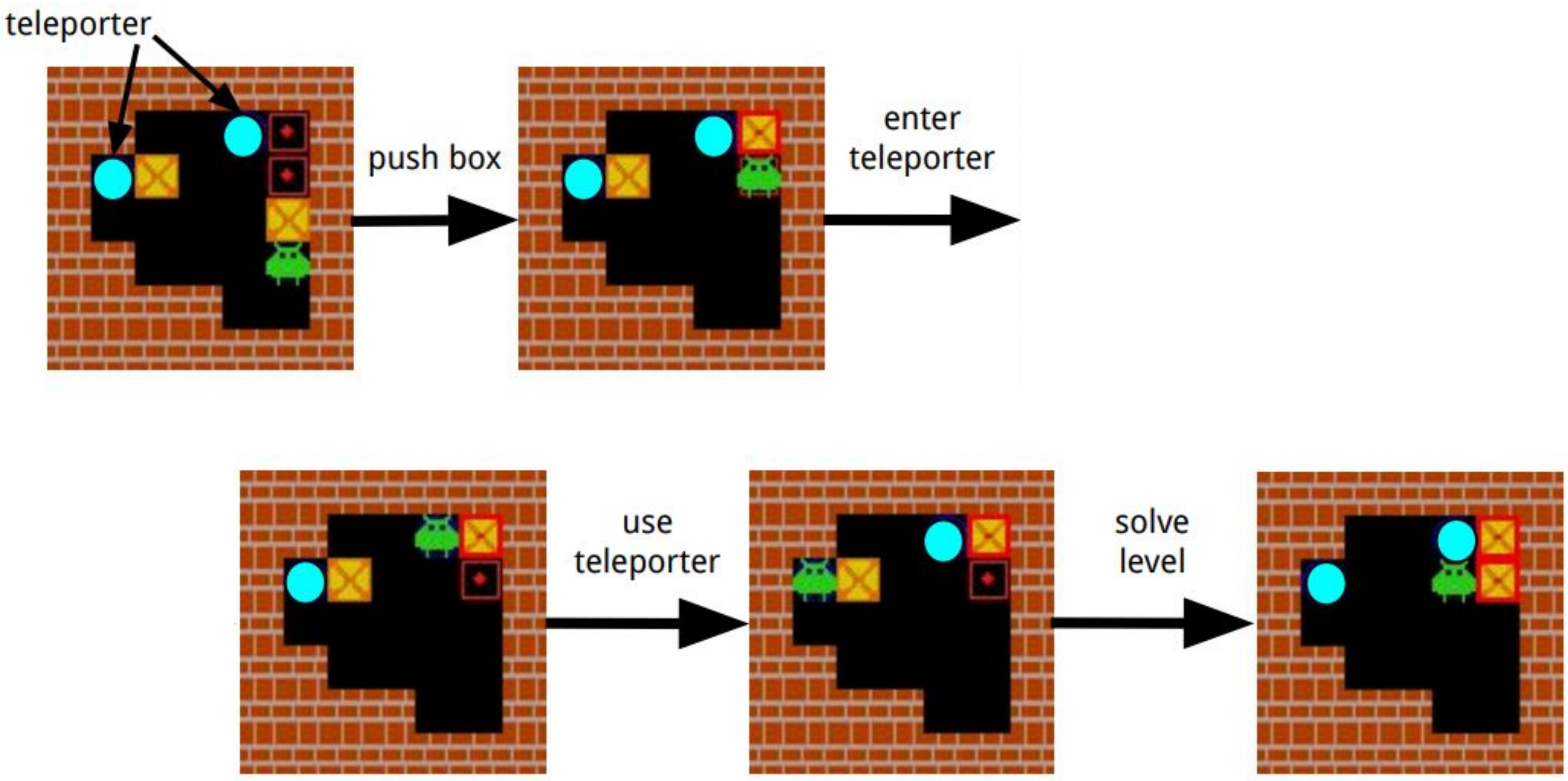



```python
def transition(state, action):
    """
    Args:
        state: a set of entities representing the state of the environment
        action: the action can be "move right", "move left", "move up", "
        move down"
    Returns:
        next_state: the next state of the environment
    """
    # here we define how the player coordinates change for each action
    action_to_delta = {
        "move right": (1, 0),
        "move left": (-1, 0),
        "move up": (0, -1),
        "move down": (0, 1)
    }
```

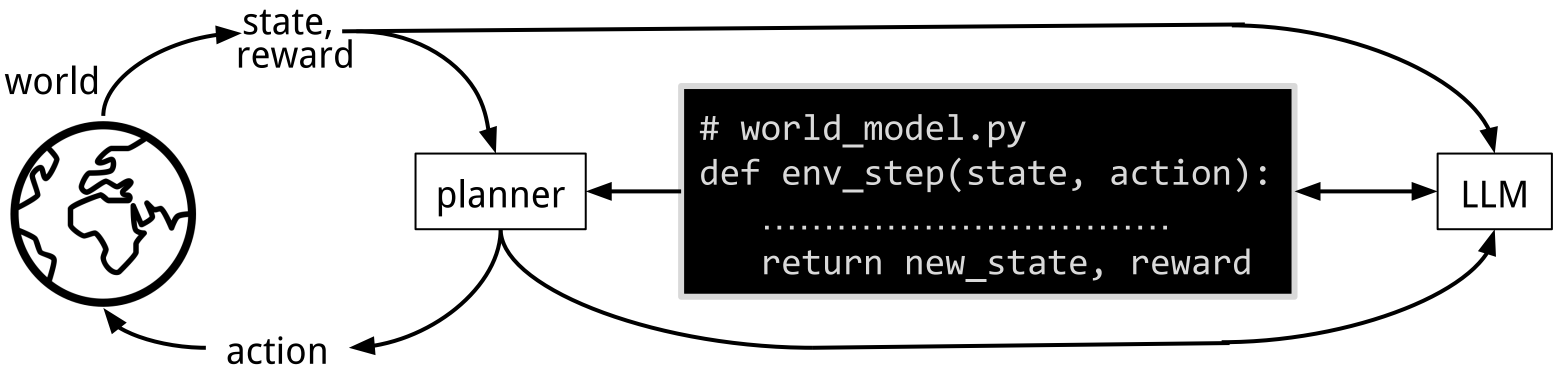
```python
    # Here we get the player and the boxes in the current state
    player = get_entities_by_name(state, 'Player')[0]
    boxes = get_entities_by_name(state, 'Box')
    walls = get_entities_by_name(state, 'Wall')
    # Then, we calculate the new player position according to the action
    delta_x, delta_y = action_to_delta[action]
    new_player_x = player.x + delta_x
    new_player_y = player.y + delta_y
    # We check if the new player position is a Wall
    if get_entities_by_position(walls, new_player_x, new_player_y):
        # If so, the player does not move
        pass
    else:
        # If not, the player moves to the new position
        pushed_box = get_entities_by_position(boxes, new_player_x,
        new_player_y)
```

Hao Tang et al. NeurIPS '24

Not in pretraining: Sokoban + Teleporter



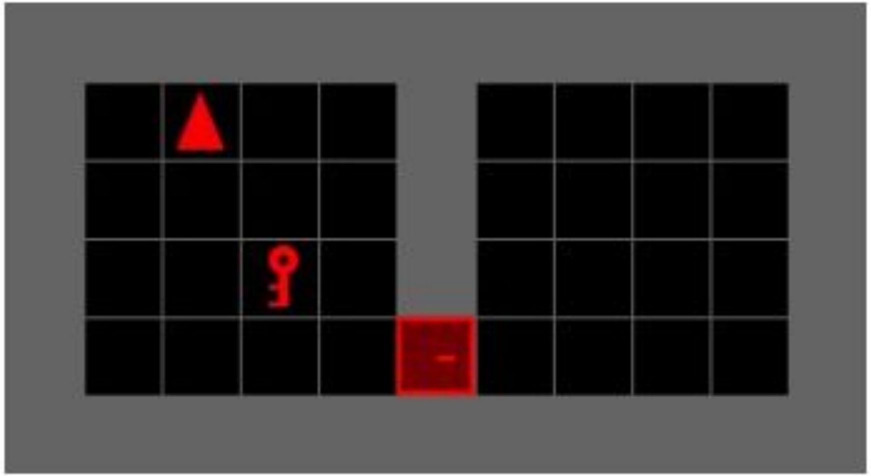
Agent Architecture



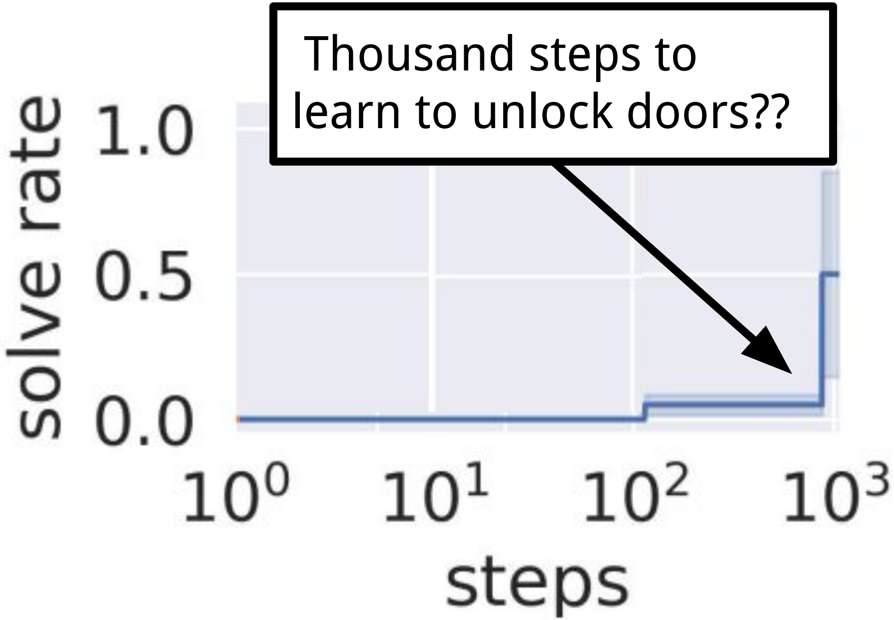
Like EMPA: Tsividis et al. 2021

Hao Tang et al. NeurIPS ‘24

Learning is Program Synthesis => Sample/Data Efficient
…if you have successful trajectories to learn from

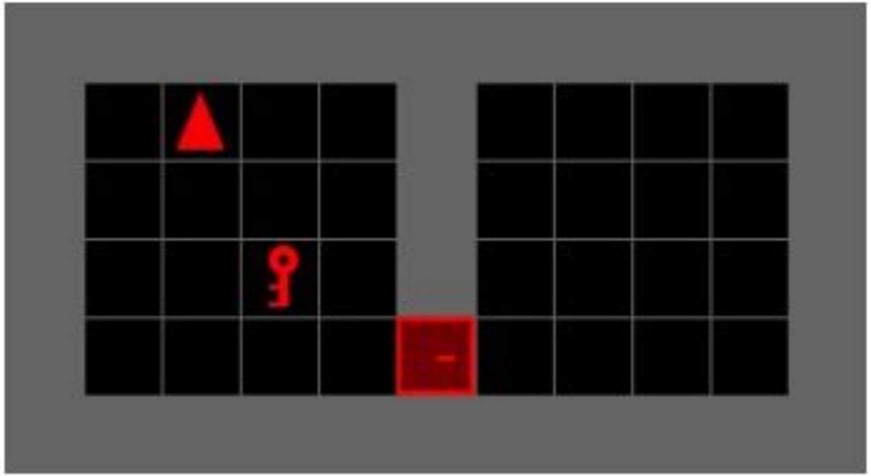


“open the door”



Hao Tang et al. NeurIPS ‘24

Learning is Program Synthesis => Sample/Data Efficient
…if you have successful trajectories to learn from



"open the door"

+optimism: inductive bias favoring optimistic world models

WorldCoder Lessons

A new prior over programs: **OPTIMISM**

Previously we'd favored simplicity

Online inference BUT no Bayesian framing

…but we're finding those framings useful for harder world models

Hao Tang et al. NeurIPS '24

Learning Abstract World Models

Yichao Liang et al. arXiv '24

Not Abstract World Models

genie: prompt->game

\+ sora: prompt->video

+Model Based RL more generally:

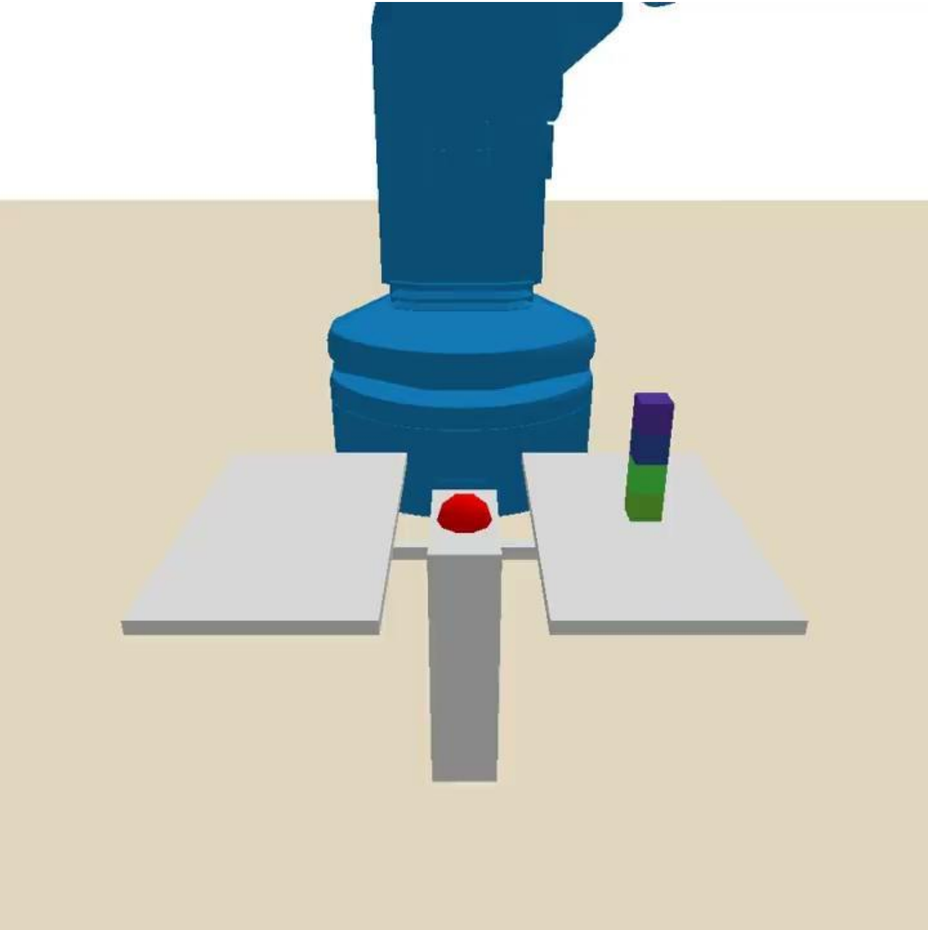
Dreamer, PlaNet, MuZero, …

Unclear if pixel-learners understand that:

Going to college helps get a good job
Balance beams can tell if masses are equal
Raising prices lowers demand
Water expands when frozen
Trees can live a long time

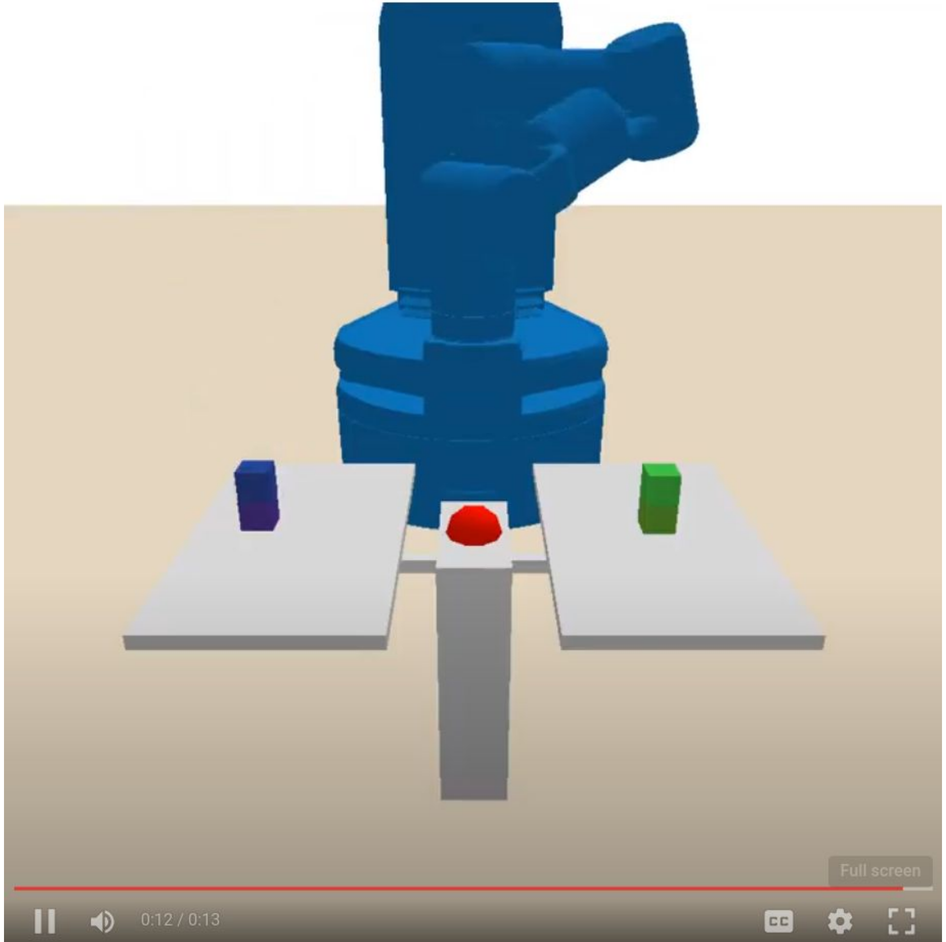
Yichao Liang et al. arXiv '24

Abstract World Models



Yichao Liang et al. arXiv ‘24

Abstract World Models



is it balanced?

↑

how much stuff is each platter supporting?

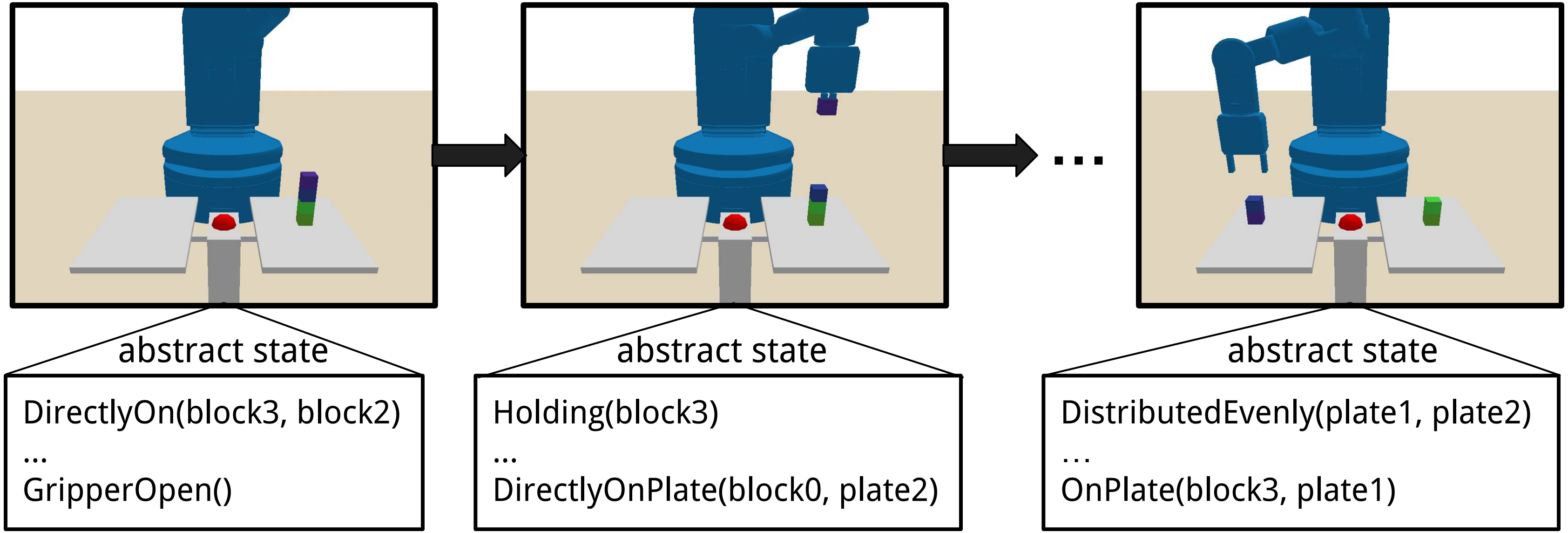
↑

is object A on top of object B?

↑

raw pixels

Yichao Liang et al. arXiv ‘24

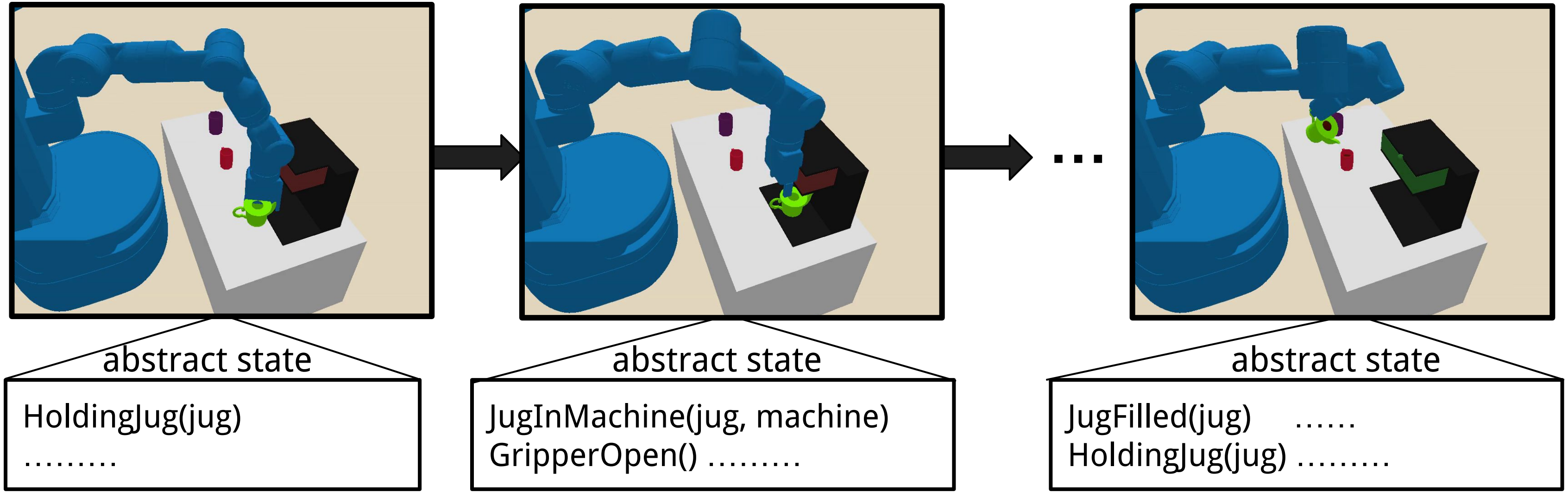


learned state abstraction

```
def DistributedEvenly(state, plate1, plate2):
 if plate1 == plate2: return False
 count1, count2 = 0, 0
 for obj in state.objects:
   if OnPlate(state, obj, plate1): count1 += 1
   if OnPlate(state, obj, plate2): count2 += 1
 return count1 == count2
```

learned abstract state dynamics

```
def press_button(state, plate1, plate2):
    # Precondition
    assert distributed_evenly(state, plate1, plate2)
    # Postcondition
    new_state = state.copy()
    new_state['machine_on'] = True
    return new_state
```

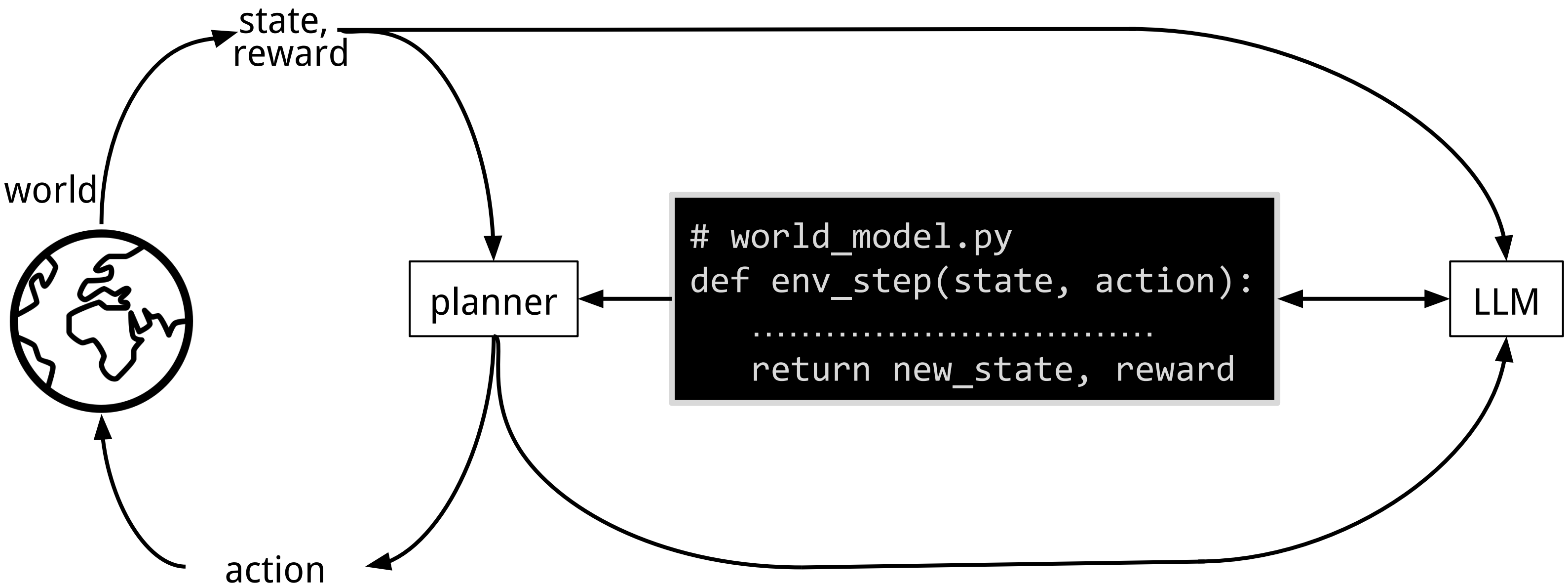


learned state abstraction

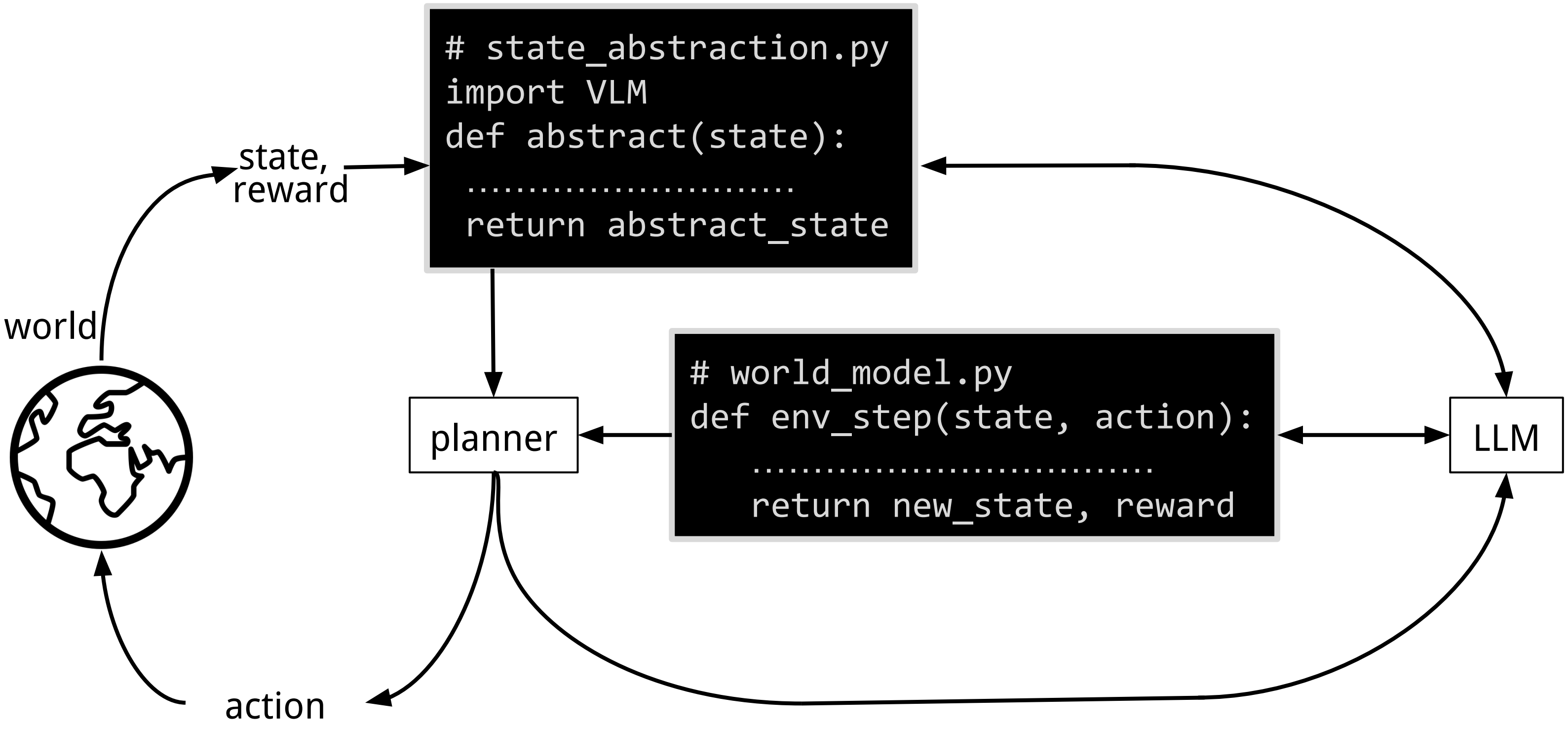
```python
def JugInMachine(state, jug, machine):
 # If the jug is held, it cannot be in the machine.
 if Holding(state, state.robot, jug):
   return False
 # Crop to focus on jug and coffee machine
 attention_img = state.crop_to_objects([jug,machine])
 return attention_img.query_VLM(
       f"{jug.name} is placed inside {machine.name}.")
```

learned abstract state dynamics

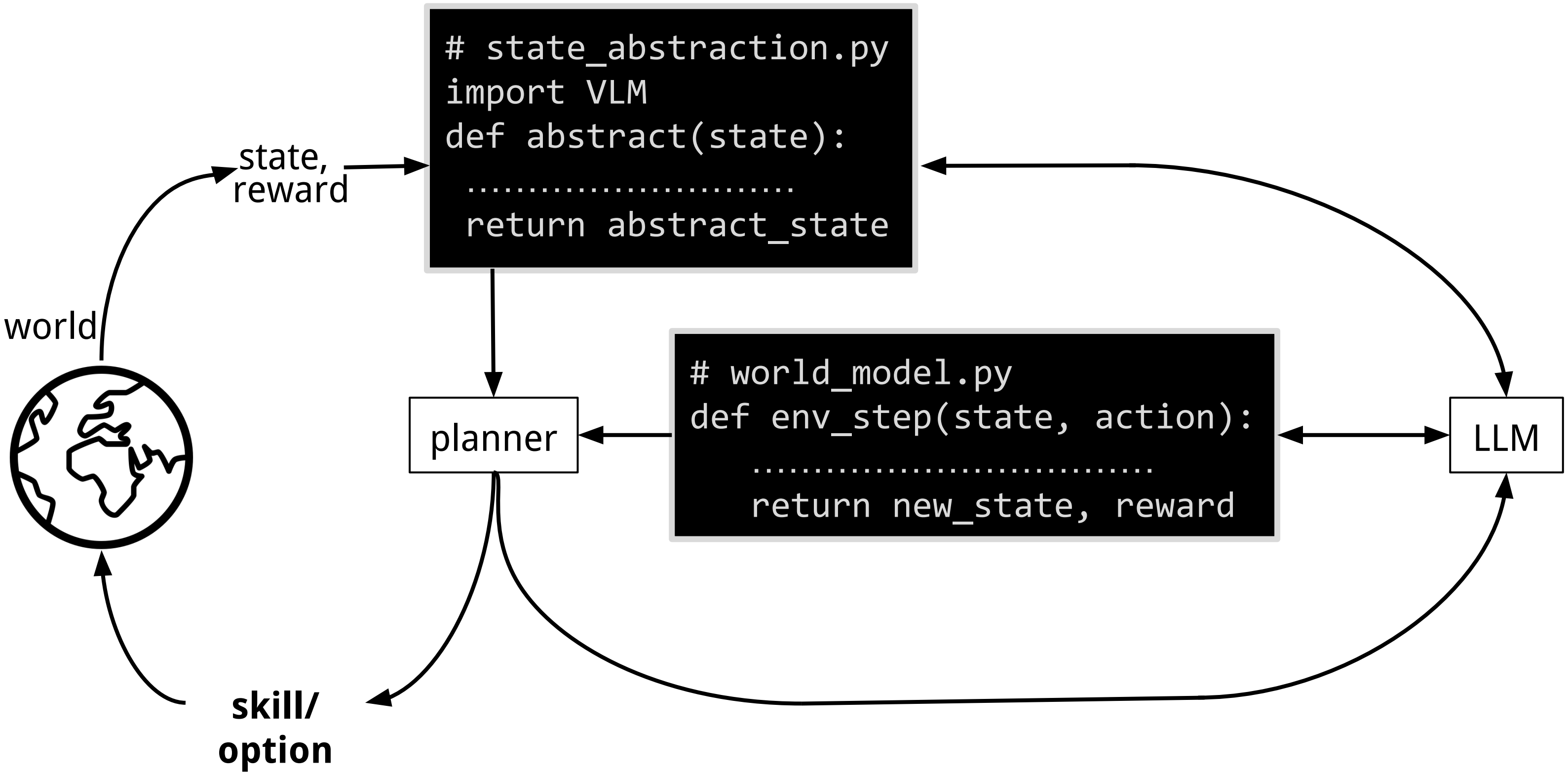
```python
def turn_on(state, coffee_machine, jug, robot):
    # Precondition
    assert JugInMachine(state, jug, coffee_machine)

    # Postcondition
    new_state = state.copy()
    new_state['jug_filled'][jug] = True
    return new_state
```



Yichao Liang et al. arXiv ‘24



Yichao Liang et al. arXiv ‘24



Assume pretrained temporally-extended high-level actions:
“Skills”/Option

Yichao Liang et al. arXiv ‘24

GenAI World Model vs Abstract World Model

<u>GenAI world models:</u>

Precise model of the world

Requires big training data

Cannot adapt on-the-fly to new dynamics

<u>Abstract world model:</u>

Incomplete model of the world

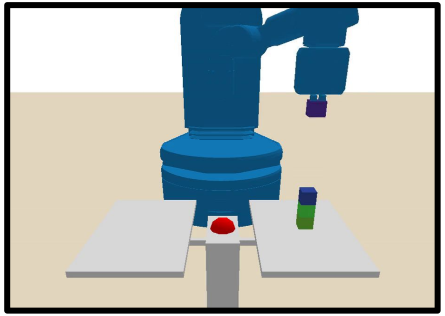
Requires big **pre**training data

Quickly learn new dynamics

Hierarchical Abstraction

Limitations of this work in particular: full observability, determinism, fixed skills

Yichao Liang et al. arXiv ‘24



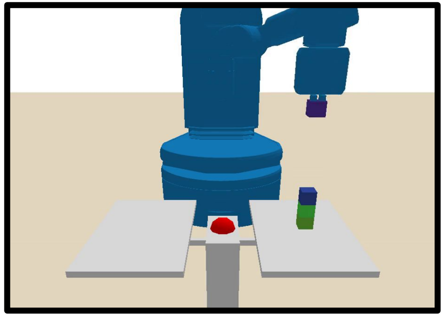
```
def DirectlyOn(state, x, y):
 img = state.crop_to_objects([x,y])
 return img.query_VLM(
     f"{x.name} is on top of {y.name}.")
```

```
def OnPlate(state, x, y):
 if DirectlyOnPlate(state,x,y): return True
 for other_block in state.other_blocks:
  if DirectlyOn(state, x, other_block) \
  and OnPlate(state, other_block, y):
   return True
 return False
```

```
def DistributedEvenly(state,plate1,plate2):
 if plate1 == plate2: return False
 cnt1, cnt2 = 0, 0
 for obj in state.objects:
  if OnPlate(state, obj, plate1): cnt1 += 1
  if OnPlate(state, obj, plate2): cnt2 += 1
 return count1 == count2
```

Hierarchy:
State abstractions
recursively build
on each other

Yichao Liang et al. arXiv '24



```python
def DirectlyOn(state, x, y):
 img = state.crop_to_objects([x,y])
 return img.query_VLM(
     f"{x.name} is on top of {y.name}.")
```

Note:

[+] OnPlate calls itself!

[-] Don’t actually learn DirectlyOn

```python
def OnPlate(state, x, y):
 if DirectlyOnPlate(state,x,y): return True
 for other_block in state.other_blocks:
  if DirectlyOn(state, x, other_block) \
  and OnPlate(state, other_block, y):
   return True
 return False
```

```python
def DistributedEvenly(state,plate1,plate2):
 if plate1 == plate2: return False
 cnt1, cnt2 = 0, 0
 for obj in state.objects:
  if OnPlate(state, obj, plate1): cnt1 += 1
  if OnPlate(state, obj, plate2): cnt2 += 1
 return count1 == count2
```

Hierarchy:
State abstractions
recursively build
on each other

Yichao Liang et al. arXiv ‘24

Lessons

Can't actually model the whole world in code!

Build symbolic abstractions of the world, and model those instead

Beyond 2-level hierarchy:

Abstractions can recursively build on top of abstractions

Yichao Liang et al. arXiv '24

Last part:

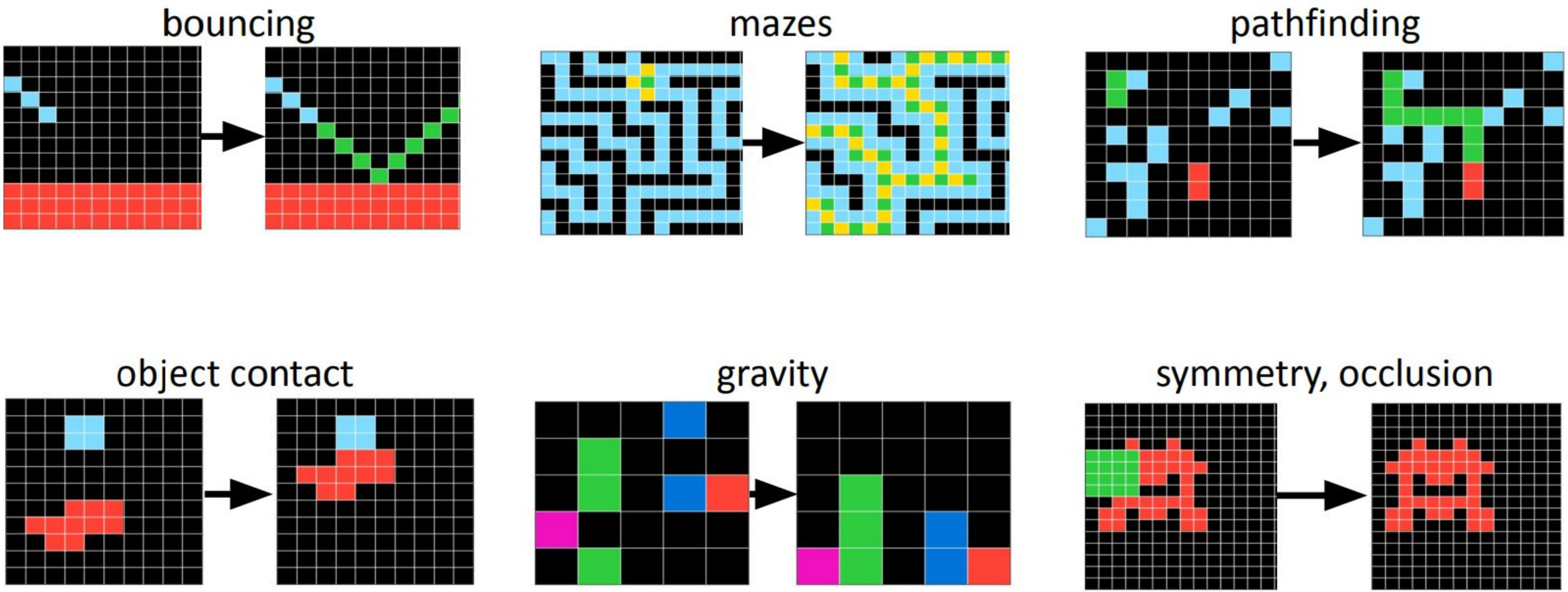
How much of the world can/should we model in symbolic code?

Wen-Ding Li & Keya Hu et al. arXiv '24

Abstraction and Reasoning Corpus [Chollet 2019]

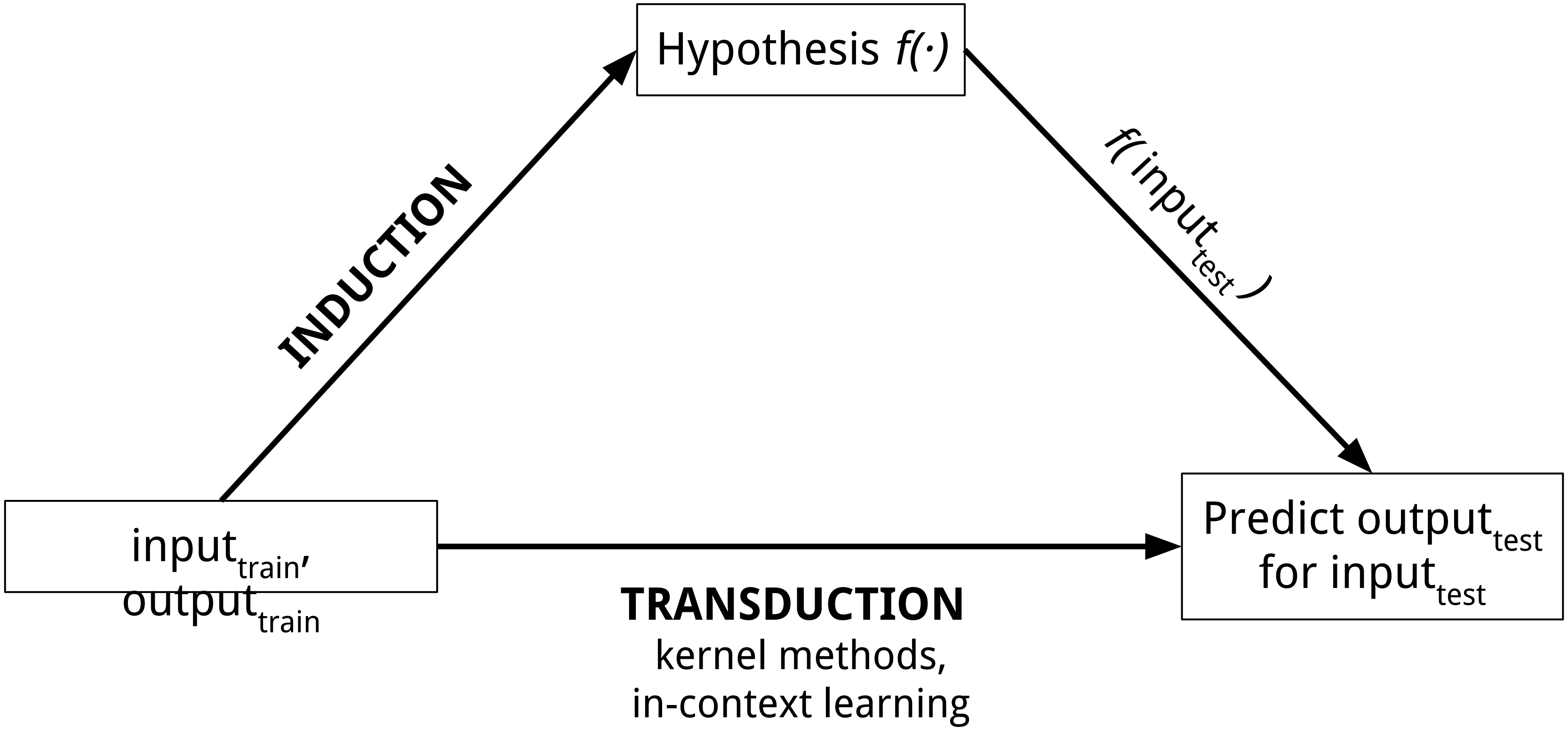
Wen-Ding Li & Keya Hu et al. arXiv ‘24

Abstraction and Reasoning Corpus [Chollet 2019]



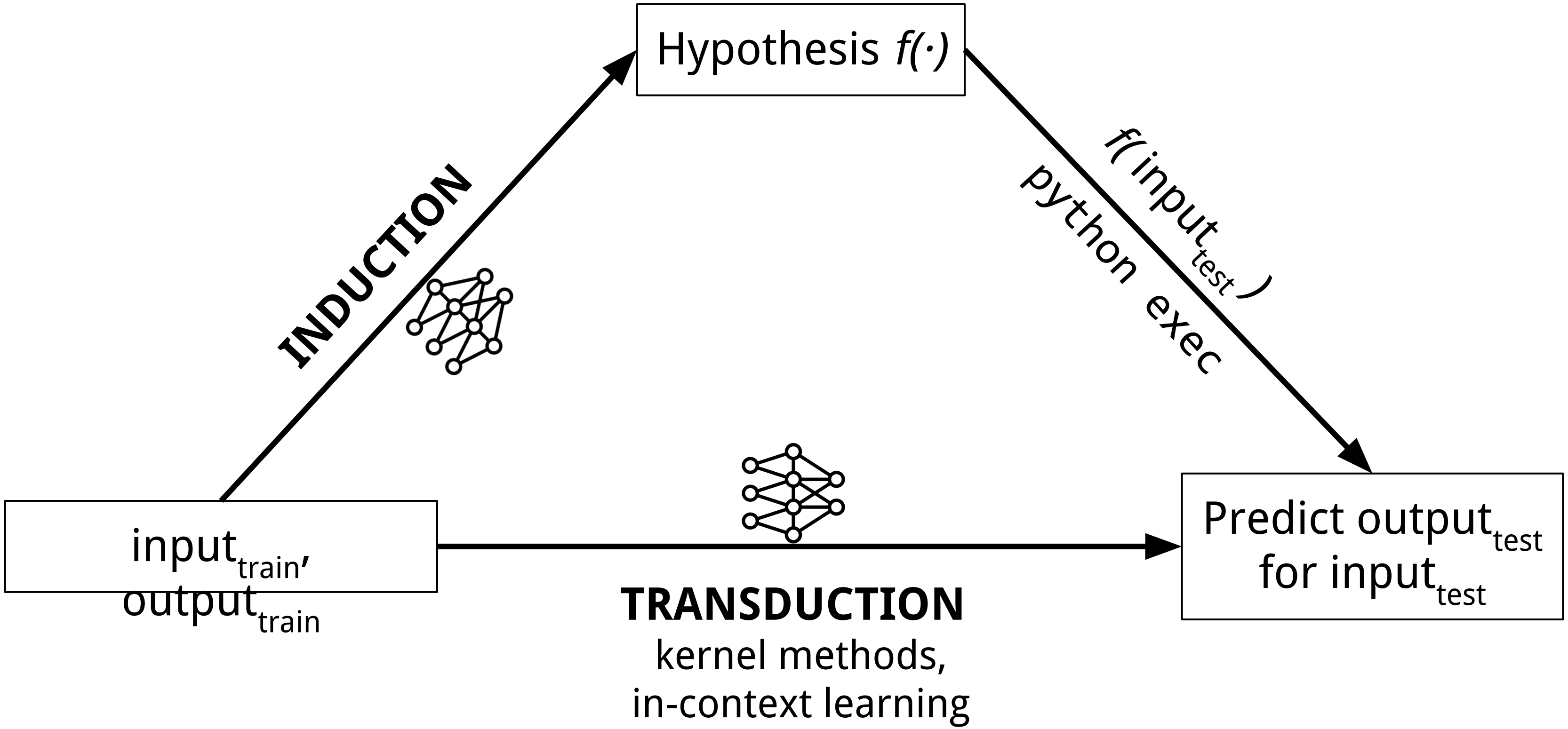
Wen-Ding Li & Keya Hu et al. arXiv '24

Frameworks for function learning



Wen-Ding Li & Keya Hu et al. arXiv '24

Neural Networks for Induction and Transduction



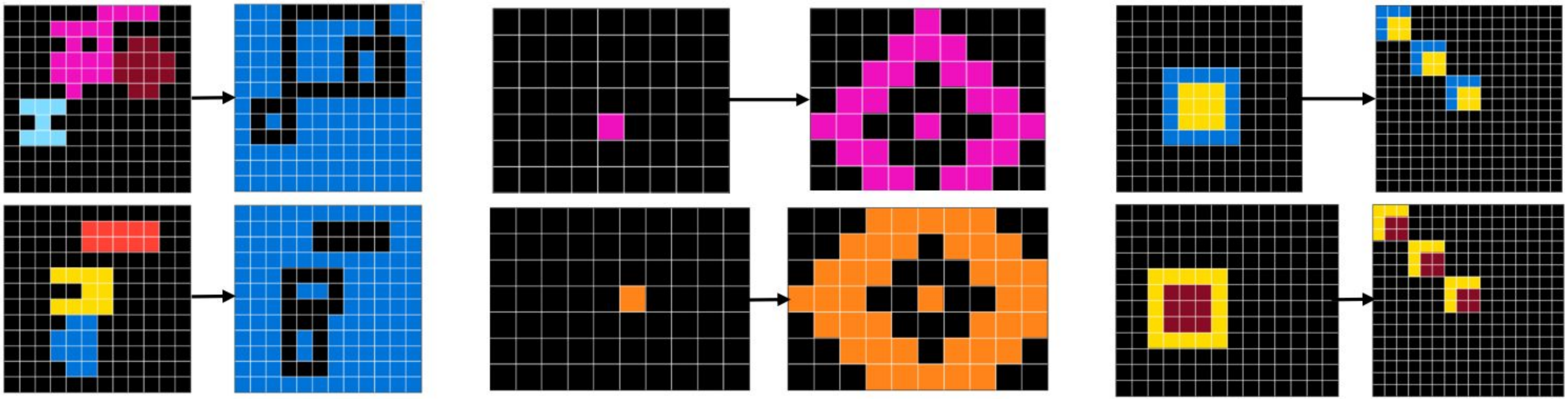
Wen-Ding Li & Keya Hu et al. arXiv '24

Meta-Learning Induction and Transduction

Metalearning dataset: Tuples of

<$\text{input}_{\text{train}}$, $f(\cdot)$, $\text{input}_{\text{test}}$>

400k synthetic problems, 100% explainable by Python code



Induction (Program Synthesis)
VS
Transduction (In-Context Learning)

It’s a tie!

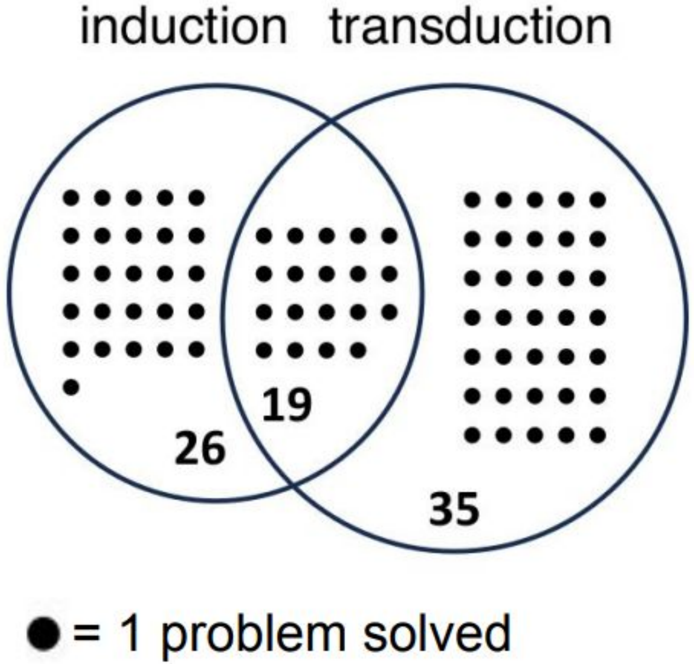
But solve different problems

Weird!

- same training problems,

  all solvable with programs

- same neural architecture



Combining Induction and Transduction

Generate ONE dataset of few-shot learning problems

Fine-tune TWO models (induction&transduction)

Ensemble them

=> New SOTA on ARC, 54.4% on validation (human=60.2%)

Wen-Ding Li & Keya Hu et al. arXiv '24

What did we learn?

Generating symbolic hypotheses [Induction]

Imitating symbolic hypotheses [Transduction]

Complementary—*even controlling for neural architecture and training problems!*

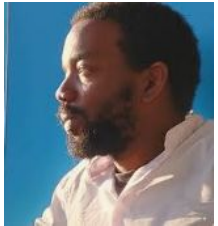
Speculation: System 1/2 divide is normative-ish

Not an incidental consequence of architectural decisions

Wen-Ding Li: on industry job market

Keya Hu: applying to PhD's

Zenna Tavares: we're hiring for next steps



Induction can count, and pinpoint the center of an object
Transduction knows qualitative object relations/properties