上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Big Models for Decision Making

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

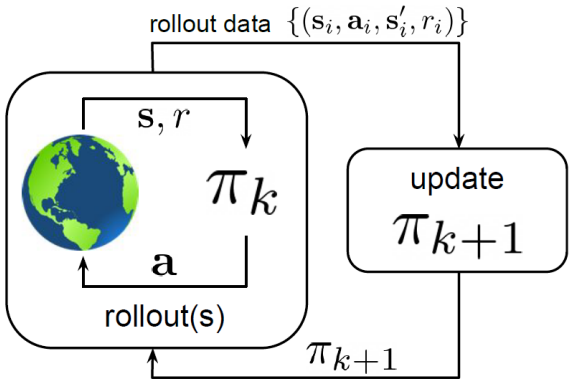
May 2022

Content

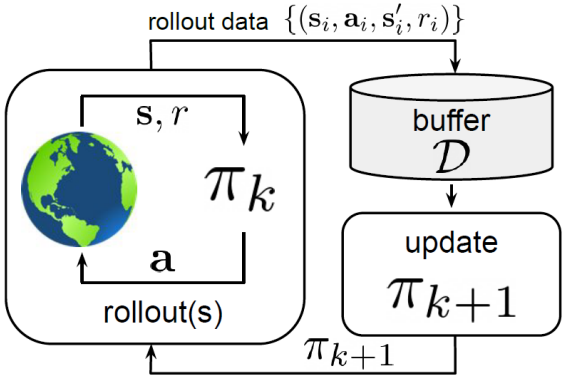
- Review the process of reinforcement learning
- Transformers in decision-making tasks
- Gato: a generalist agent
- Discussion and prospects

Overview of RL Training Paradigms

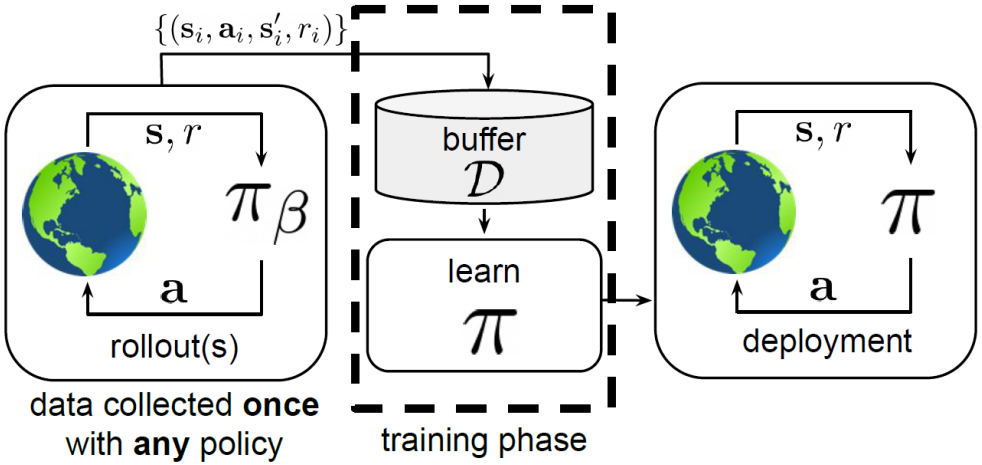
(a) online reinforcement learning



(b) off-policy reinforcement learning



(c) offline reinforcement learning

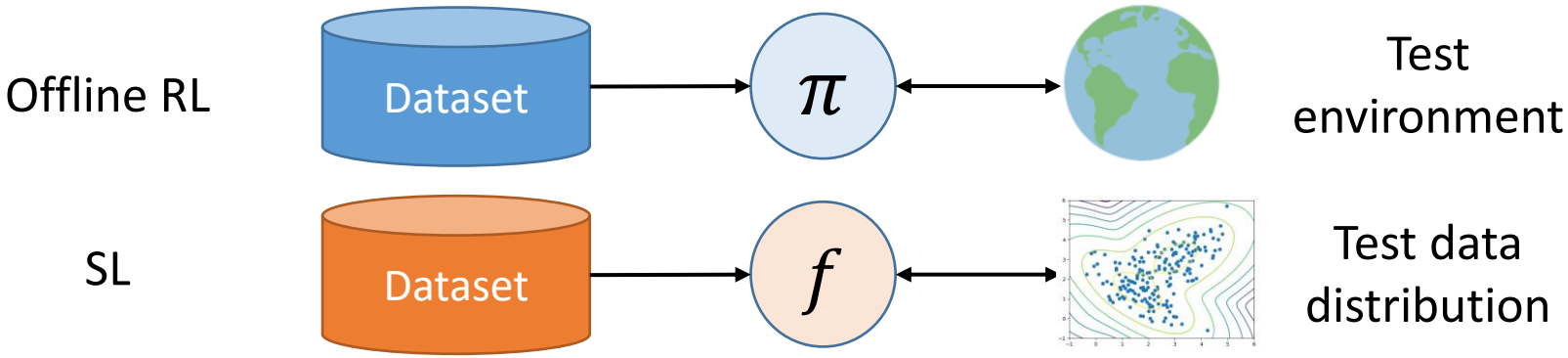


- Though both off-policy RL and offline RL evaluate the policy using the data sampled from a replay buffer, they are different.
- The key difference is whether the agent can interact with the environment while learning
- Offline RL techniques help deploy RL to real-world applications

Levine, Sergey, et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." *arXiv 2005.01643* (2020).

Advantages of Offline RL

- Offline RL can help
  1. Pretrain an RL agent using an existing dataset
  2. Empirically evaluate RL algorithms based on their ability to exploit a fixed dataset of interactions
  3. Bridge the gap between academic interest in RL and real-world applications
- Offline RL makes RL more like supervised learning



Gulcehre, Caglar, et al. "RL unplugged: Benchmarks for offline reinforcement learning." *arXiv preprint arXiv:2006.13888* (2020).

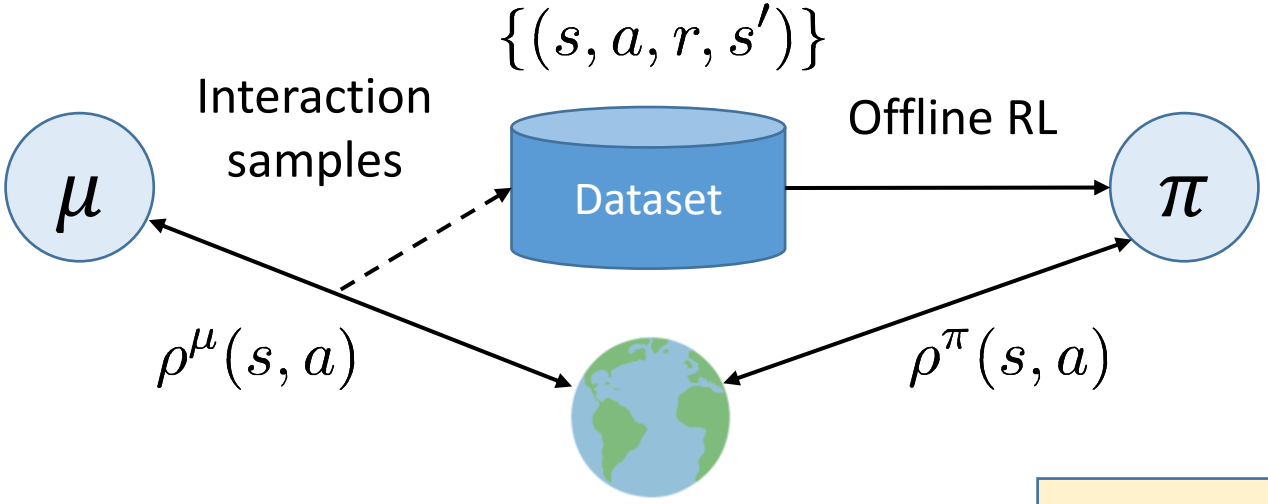
Key scientific problem

Extrapolation Error in Offline RL

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$$

What if a' is an out-of-distribution action?

- Extrapolation error is introduced by the mismatch between the dataset and true state-action visitation of the current policy.



$$\rho^{\pi} \neq \rho^{\mu}$$

Overview of Offline RL Methods

Model-free Methods

- Conservative value / policy
  - BCQ
  - BEAR
  - BRAC
  - CQL …
- Self-imitation
  - BC (%)
  - AWR
  - BAIL
  - …

Model-based Methods

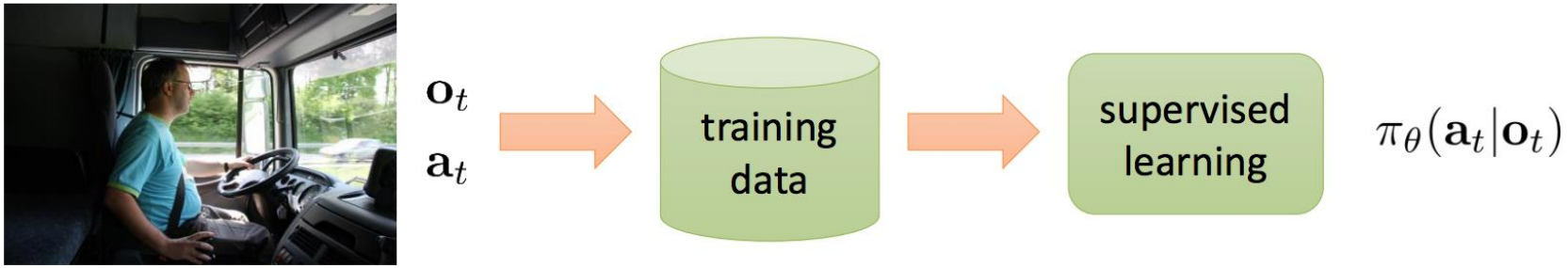
- Uncertainty estimation with the learned model
  - MOReL
  - MOPO
  - COMBO
  - …

- The most severe problem that offline RL faces is the extrapolation error, i.e., the out-of-distribution problem: What if the agent performs unseen state-action?
- Implicit constraint methods directly learn the policy on the data support
  - with data instance selection or weighting

Behavior Cloning as Offline RL

- Behavior cloning, the simplest imitation learning method, requires no environment interaction

Behavioral policy



- Learning objective of BC

Behavioral policy

$$\hat{\pi}^* = \arg\max_{\pi} \mathbb{E}_{\mathbf{s}\sim d_{\pi_E}} \mathbb{E}_{\mathbf{a}\sim \pi_E(\mathbf{s})} [\log \pi(\mathbf{a}|\mathbf{s})]$$

- Obvious shortcomings of BC
  1. The policy upper bound is the behavioral policy
  2. Distribution shift

AWR: Advantage-Weighted Regression

- Policy optimization objective

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] = \mathbb{E}_{\mathbf{s} \sim d_\pi(\mathbf{s})} \mathbb{E}_{a \sim \pi(\mathbf{a}|\mathbf{s})} \left[r(\mathbf{s}, \mathbf{a}) \right]$$

Expected improvement

$$\eta(\pi) = J(\pi) - J(\mu) \quad \text{[as derived in TRPO]}$$
$$= \mathbb{E}_{\mathbf{s} \sim d_\pi(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \left[A^\mu(\mathbf{s}, \mathbf{a}) \right] = \mathbb{E}_{\mathbf{s} \sim d_\pi(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} \left[\mathcal{R}^\mu_{\mathbf{s},\mathbf{a}} - V^\mu(\mathbf{s}) \right]$$

- Based on reward-weighted regression (RWR)

$$\pi_{k+1} = \arg\max_\pi \quad \mathbb{E}_{\mathbf{s} \sim d_{\pi_k}(\mathbf{s})} \mathbb{E}_{\mathbf{a} \sim \pi_k(\mathbf{a}|\mathbf{s})} \left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta} \mathcal{R}_{\mathbf{s},\mathbf{a}} \right) \right]$$

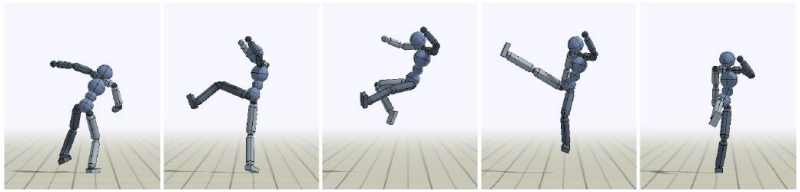
return

- Regarded as solving a maximum likelihood problem that fits a new policy to samples collected under the current policy, where the likelihood is weighted by the exponentiated return.

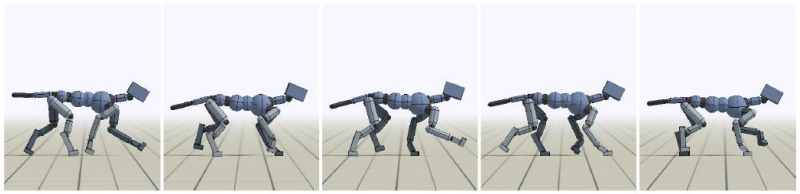
Xue Bin Peng et al . Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019.

AWR: Advantage-Weighted Regression

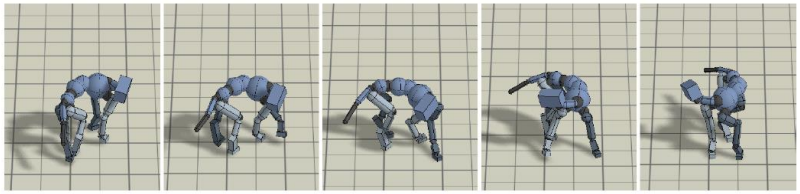
(a) Humanoid: Cartwheel



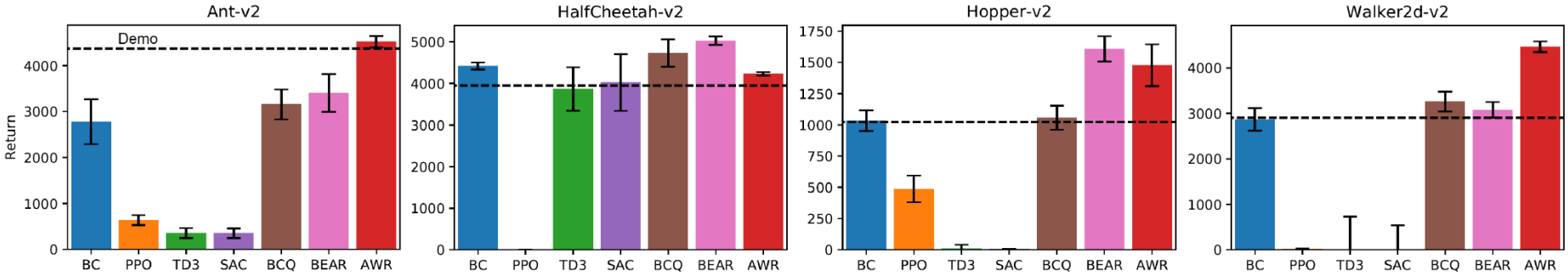
(b) Humanoid: Spinkick



(c) Dog: Trot



(d) Dog: Turn



- Performance of various algorithms on off-policy learning tasks with static datasets. AWR is able to learn policies that are comparable or better than the original demo policies.

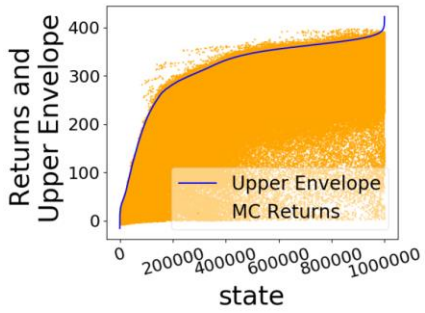
Xue Bin Peng et al . Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019.

BAIL: Best-Action Imitation Learning

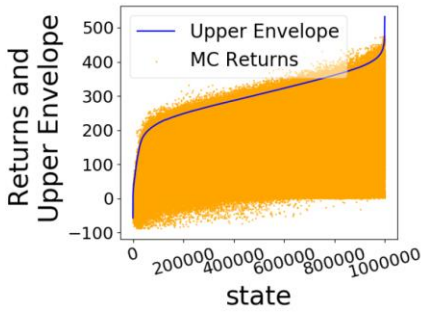
- BAIL does not suffer from the extrapolation error, since it does not maximize over the actions space.
- Step 1: learn an upper envelope of state by solving a constrained optimization problem:

$$L^K(\phi) = \sum_{i=1}^{m} (V_\phi(s_i) - G_i)^2 \{\mathbb{1}_{(V_\phi(s_i) \geq G_i)} + K \cdot \mathbb{1}_{(V_\phi(s_i) < G_i)}\} + \lambda \|w\|^2$$

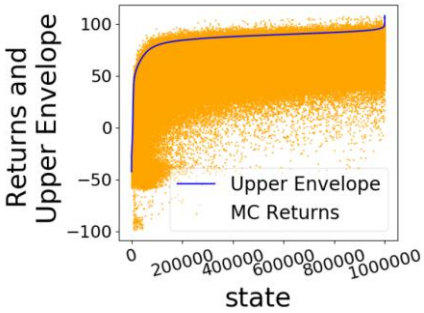
- Step 2: Select actions satisfying $G_i > x\, V(s_i)$ to perform simple imitation learning (BC). x is set such that 25% samples are selected.
- Step 3: perform supervised learning (BC) on selected data



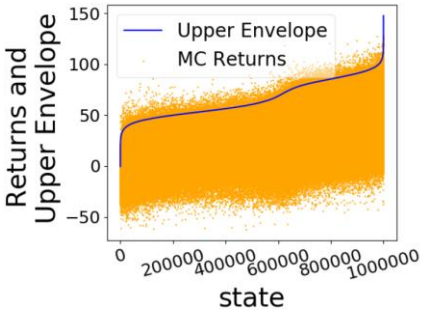
Hopper



Walker2d



HalfCheetah



Ant

X. Chen et al. BAIL: Best-action imitation learning for batch deep reinforcement learning. NeuIPS 2020.

BAIL: Best-Action Imitation Learning

Environment	BAIL	BCQ	BEAR	BC	MARWIL
$\sigma = 0.1$ Hopper B1	$\mathbf{2173 \pm 291}$	1219 ± 114	505 ± 285	626 ± 112	827 ± 220
$\sigma = 0.1$ Hopper B2	$\mathbf{2078 \pm 180}$	1178 ± 87	985 ± 3	579 ± 141	620 ± 336
$\sigma = 0.1$ Walker B1	$\mathbf{1125 \pm 113}$	576 ± 309	610 ± 212	514 ± 17	436 ± 24
$\sigma = 0.1$ Walker B2	$\mathbf{3141 \pm 300}$	2338 ± 388	2707 ± 425	1741 ± 239	1810 ± 200
$\sigma = 0.1$ HC B1	$\mathbf{5746 \pm 29}$	$\mathbf{5883 \pm 43}$	0 ± 0	$\mathbf{5546 \pm 29}$	$\mathbf{5573 \pm 35}$
$\sigma = 0.1$ HC B2	$\mathbf{7212 \pm 43}$	$\mathbf{7562 \pm 31}$	0 ± 0	6765 ± 108	$\mathbf{6828 \pm 111}$
$\sigma = 0.5$ Hopper B1	$\mathbf{2054 \pm 158}$	1145 ± 300	203 ± 42	919 ± 52	946 ± 103
$\sigma = 0.5$ Hopper B2	$\mathbf{2623 \pm 282}$	1823 ± 555	241 ± 239	694 ± 64	818 ± 112
$\sigma = 0.5$ Walker B1	$\mathbf{2522 \pm 51}$	1552 ± 455	1248 ± 181	2178 ± 178	2111 ± 52
$\sigma = 0.5$ Walker B2	$\mathbf{3115 \pm 133}$	2785 ± 123	2302 ± 630	2483 ± 94	2364 ± 228
$\sigma = 0.5$ HC B1	1055 ± 9	$\mathbf{1222 \pm 38}$	924 ± 579	570 ± 35	512 ± 43
$\sigma = 0.5$ HC B2	$\mathbf{7173 \pm 120}$	5807 ± 249	-114 ± 140	$\mathbf{6545 \pm 171}$	$\mathbf{6668 \pm 93}$
SAC Hopper B1	$\mathbf{3296 \pm 105}$	2681 ± 438	1000 ± 110	2853 ± 318	2897 ± 227
SAC Hopper B2	1831 ± 915	$\mathbf{2134 \pm 917}$	1139 ± 317	$\mathbf{2240 \pm 367}$	$\mathbf{2063 \pm 168}$
SAC Walker B1	$\mathbf{2455 \pm 211}$	$\mathbf{2408 \pm 84}$	-3 ± 5	1674 ± 277	1484 ± 140
SAC Walker B2	$\mathbf{4767 \pm 130}$	3794 ± 398	325 ± 75	2599 ± 145	2651 ± 268
SAC HC B1	$\mathbf{10143 \pm 77}$	8607 ± 473	7392 ± 257	8874 ± 221	9105 ± 90
SAC HC B2	$\mathbf{10772 \pm 59}$	$\mathbf{10106 \pm 134}$	7217 ± 273	9523 ± 164	9488 ± 136
SAC Ant B1	$\mathbf{4284 \pm 64}$	$\mathbf{4042 \pm 113}$	3452 ± 128	$\mathbf{3986 \pm 112}$	$\mathbf{4033 \pm 130}$
SAC Ant B2	$\mathbf{4946 \pm 148}$	$\mathbf{4640 \pm 76}$	3712 ± 236	$\mathbf{4618 \pm 111}$	$\mathbf{4589 \pm 130}$
SAC Humanoid B1	$\mathbf{3852 \pm 430}$	1411 ± 250	0 ± 0	543 ± 378	589 ± 121
SAC Humanoid B2	$\mathbf{3565 \pm 153}$	1221 ± 207	0 ± 0	1216 ± 826	1033 ± 257

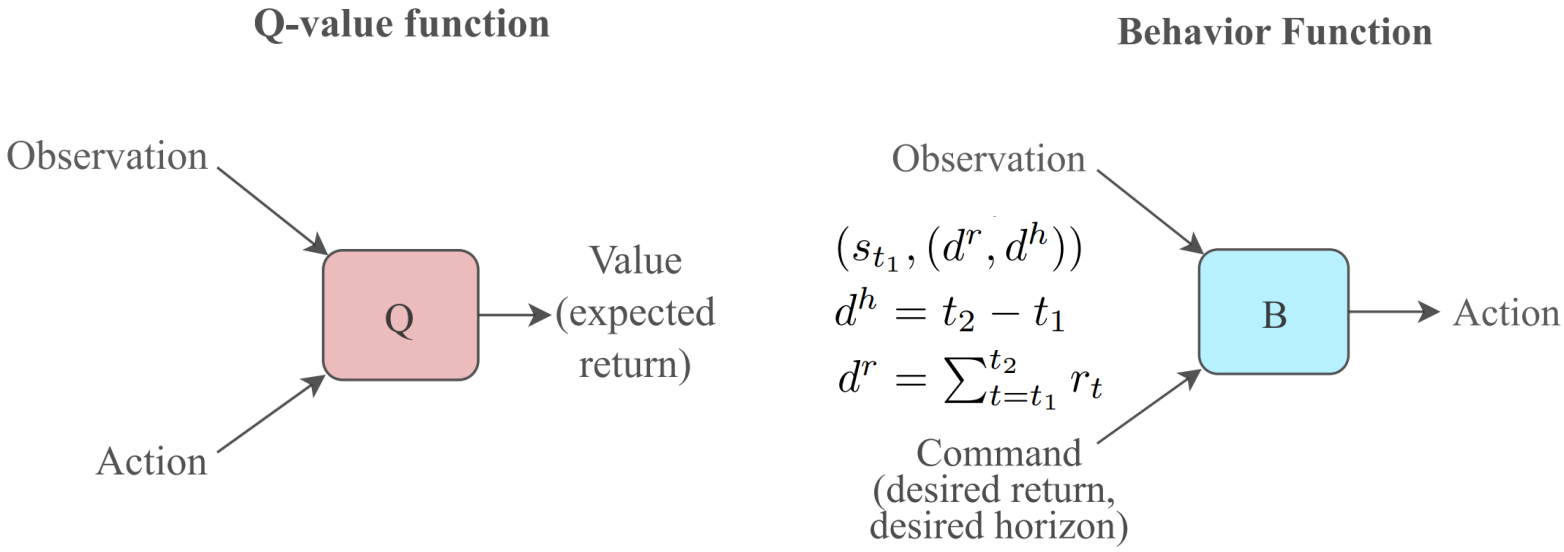
X. Chen et al. BAIL: Best-action imitation learning for batch deep reinforcement learning. NeuIPS 2020.

Content

- Review the process of reinforcement learning
- Transformers in decision-making tasks
- Gato: a generalist agent
- Discussion and prospects

Upside-Down RL

- Solve RL Problem with supervised learning methods?

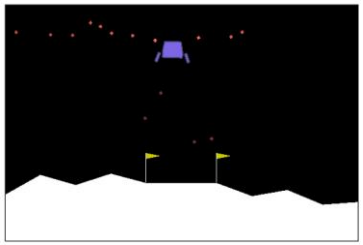


Key paper references

- Srivastava, Rupesh Kumar, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. "Training agents using upside-down reinforcement learning." *arXiv preprint arXiv:1912.02877* (2019).
- Schmidhuber, Juergen. "Reinforcement Learning Upside Down: Don't Predict Rewards--Just Map Them to Actions." *arXiv preprint arXiv:1912.02875* (2019).

Upside-Down RL

- Algorithms and experiments



(a) LunarLander-v2

(b) TakeCover-v0

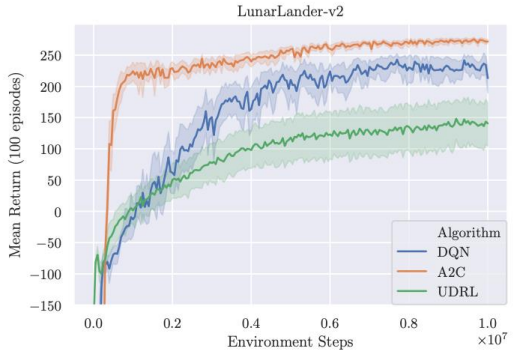
Algorithm 1 Upside-Down Reinforcement Learning: High-level Description.

```
Initialize replay buffer with warm-up episodes using random actions
Initialize a behavior function
while stopping criteria is not reached do
    Improve the behavior function by training on replay buffer
    Sample exploratory commands based on replay buffer
    Generate episodes using Algorithm 2 and add to replay buffer
    if evaluation required then
        Evaluate current agent using Algorithm 2
    end if
end while
```

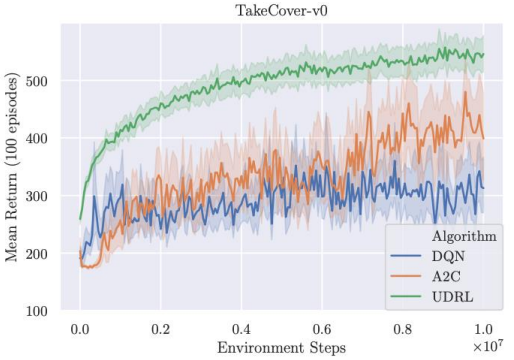
Algorithm 2 Generates an Episode using the Behavior Function.

Input: Initial command $c_0 = (d_0^r, d_0^h)$, Initial state s_0, Behavior function $B(;\theta)$
Output: Episode data E

```
E ← ∅
t ← 0
while episode is not over do
    Compute P(a_t|s_t, c_t) = B(s_t, c_t; θ)
    Execute a_t ~ P(a_t|s_t, c_t) to obtain reward r_t and next state s_{t+1} from the environment
    Append (s_t, a_t, r_t) to E
    s_t ← s_{t+1}                          // Update state
    d_t^r ← d_t^r − r_t                    // Update desired reward
    d_t^h ← d_t^h − 1                      // Update desired horizon
    c_t ← (d_t^r, d_t^h)
    t ← t + 1
end while
```



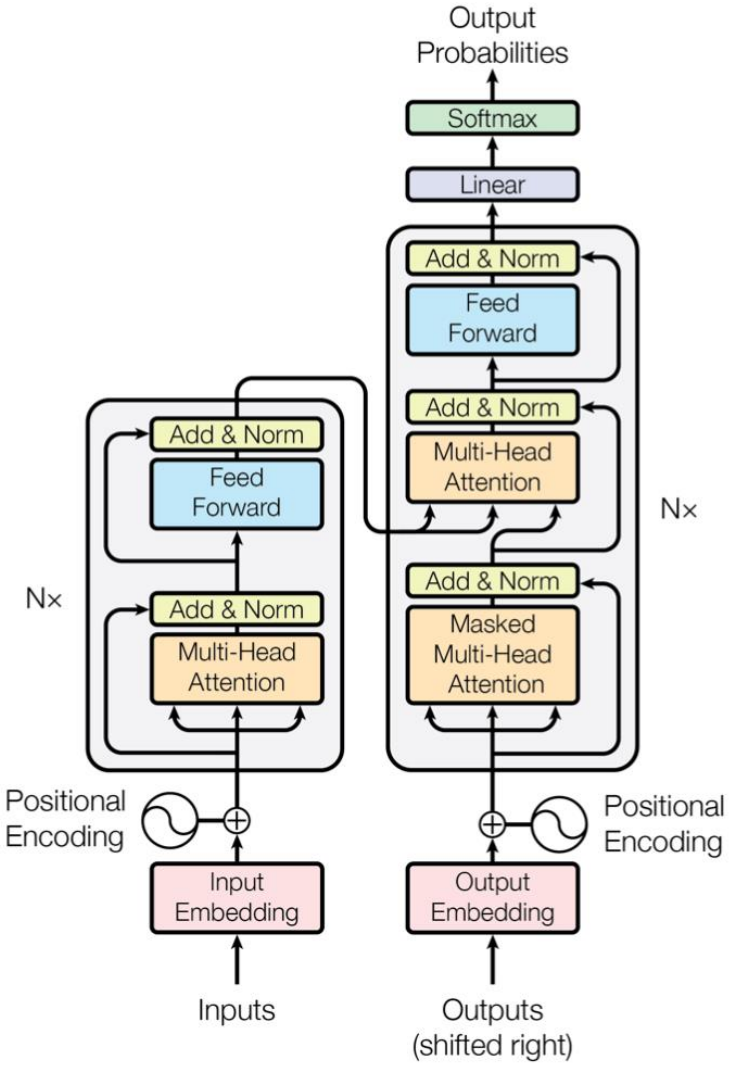
(a) On LunarLander-v2, TЯ is able to train agents that land the spacecraft, but is beaten by traditional RL algorithms.



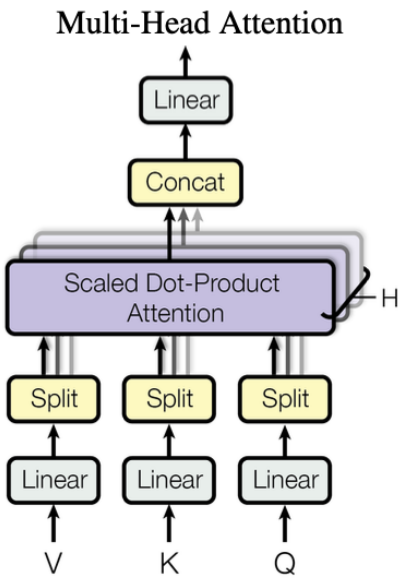
(b) On TakeCover-v0, TЯ is able to consistently yield high-performing agents, while outperforming DQN and A2C.

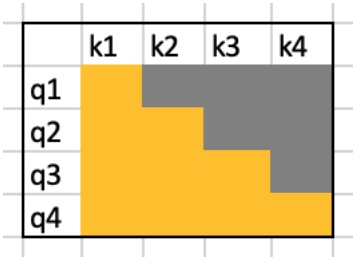
Transformer for Sequence Modeling

- Encoder
  - Inputs: A sequence of vectors (words/states)
  - Outputs: A sequence of representation
- Decoder
  - Q: Start token + previous results -> Q
  - K & V: Outputs of encoder
  - Outputs: A sequence of symbol (labels/actions)



Attention in Transformer

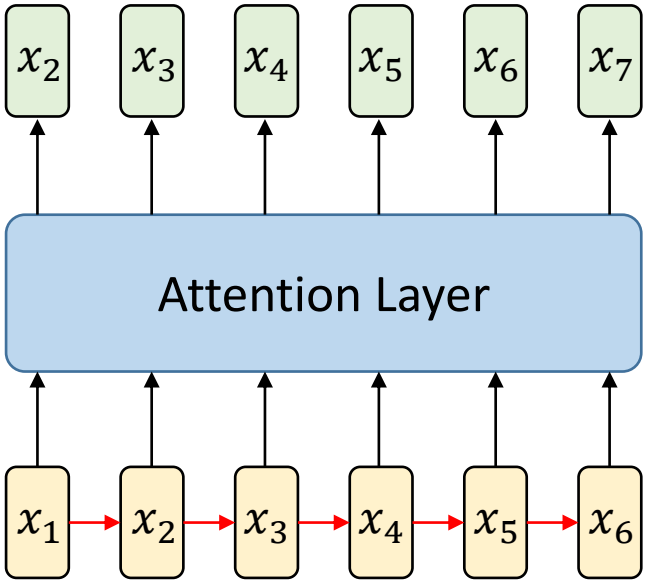




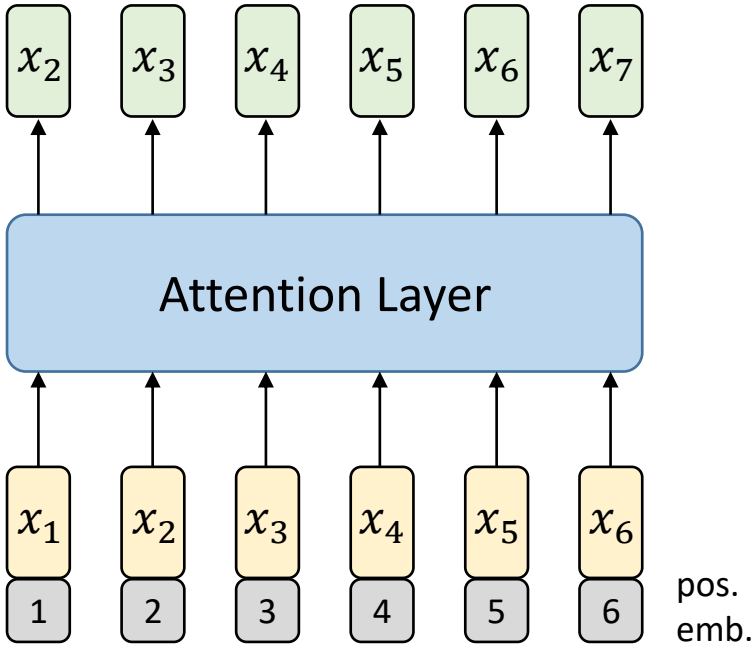

```python
class CausalSelfAttention(nn.Module):

    def __init__(self, config):
        super().__init__()
        assert config.n_embd % config.n_head == 0
        # key, query, value projections for all heads
        self.key = nn.Linear(config.n_embd, config.n_embd)
        self.query = nn.Linear(config.n_embd, config.n_embd)
        self.value = nn.Linear(config.n_embd, config.n_embd)
        # regularization
        self.attn_drop = nn.Dropout(config.attn_pdrop)
        self.resid_drop = nn.Dropout(config.resid_pdrop)
        # output projection
        self.proj = nn.Linear(config.n_embd, config.n_embd)
        # causal mask to ensure that attention is only applied to the left in the input sequence
        self.register_buffer("mask", torch.tril(torch.ones(config.block_size + 1, config.block_size + 1))
                                     .view(1, 1, config.block_size + 1, config.block_size + 1))
        self.n_head = config.n_head

    def forward(self, x, layer_past=None):
        # x: (batch, context_length, x_dim)
        B, T, C = x.size()

        # calculate query, key, values for all heads in batch and move head forward to be the batch dim
        k = self.key(x).view(B, T, self.n_head, C // self.n_head).transpose(1, 2)  # (B, nh, T, hs)
        q = self.query(x).view(B, T, self.n_head, C // self.n_head).transpose(1, 2)  # (B, nh, T, hs)
        v = self.value(x).view(B, T, self.n_head, C // self.n_head).transpose(1, 2)  # (B, nh, T, hs)

        # causal self-attention; Self-attend: (B, nh, T, hs) x (B, nh, hs, T) -> (B, nh, T, T)
        att = (q @ k.transpose(-2, -1)) * (1.0 / math.sqrt(k.size(-1)))
        att = att.masked_fill(self.mask[:, :, :T, :T] == 0, float('-inf'))
        att = F.softmax(att, dim=-1)
        att = self.attn_drop(att)
        y = att @ v  # (B, nh, T, T) x (B, nh, T, hs) -> (B, nh, T, hs)
        y = y.transpose(1, 2).contiguous().view(B, T, C)  # re-assemble all head outputs side by side

        # output projection
        y = self.resid_drop(self.proj(y))
        return y
```

Attention in RNN vs. Transformer

- Whether to use recent architecture or position embedding?



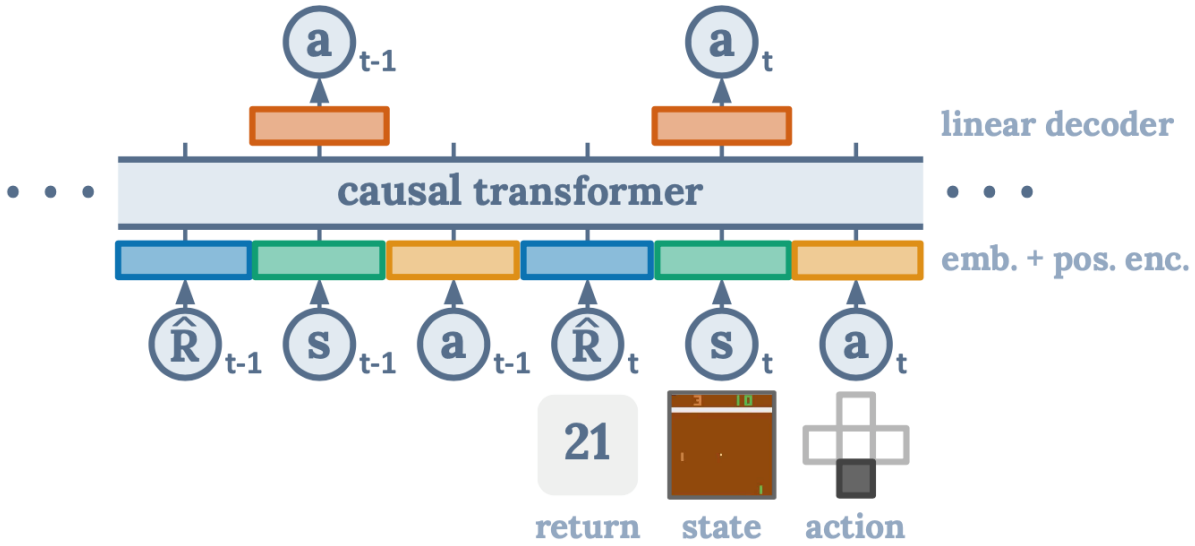
Here the same recent weights are used for multiple times of recent calculation, which introduce compounding error when decoding

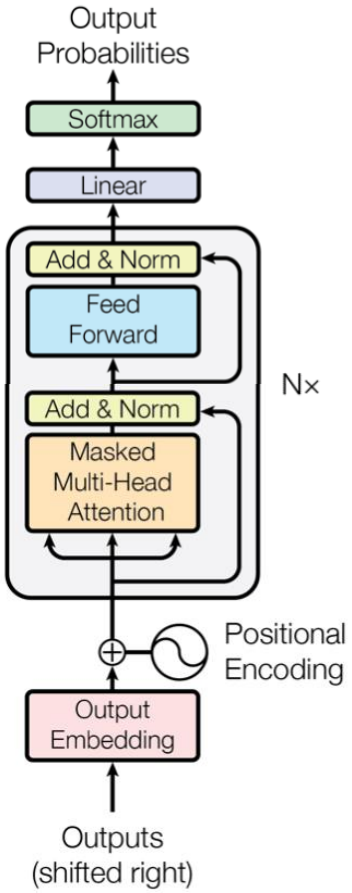


Position embedding does not introduce such a compounding error when decoding

Decision Transformer

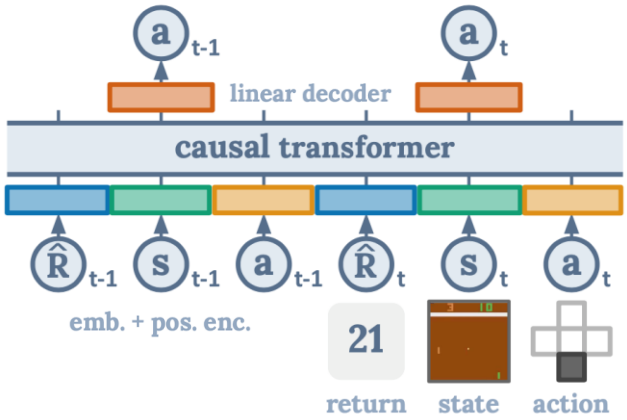
- Re-build RL task as a sequence prediction problem
- Causal transformer: GPT (i.e., decoder-only transformer)





Chen, Lili, et al. "Decision transformer: Reinforcement learning via sequence modeling." NeurIPS 2021.

Decision Transformer



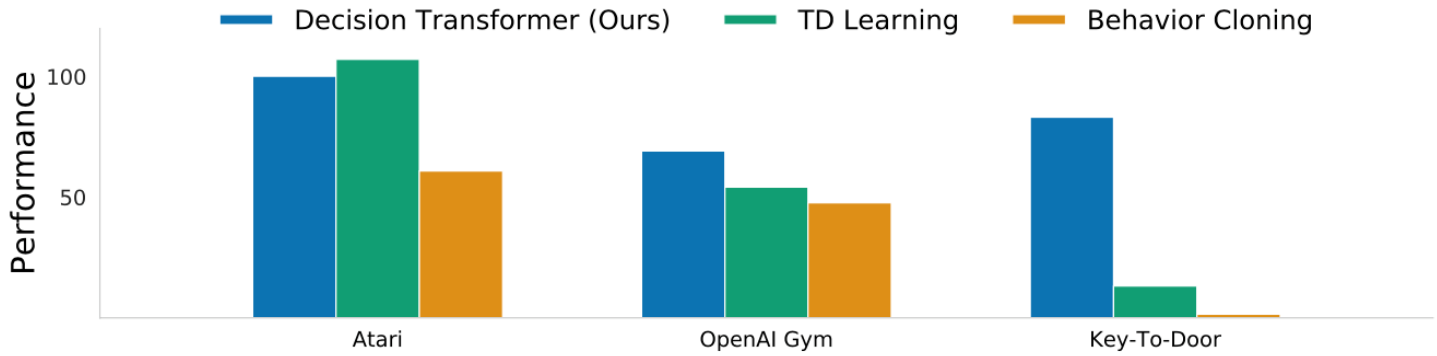
- States, actions, and returns are fed into modality-specific linear embeddings and a positional episodic timestep encoding is added.
- Tokens are fed into a GPT architecture which predicts actions autoregressively using a causal self-attention mask

Algorithm 1 Decision Transformer Pseudocode (for continuous actions)

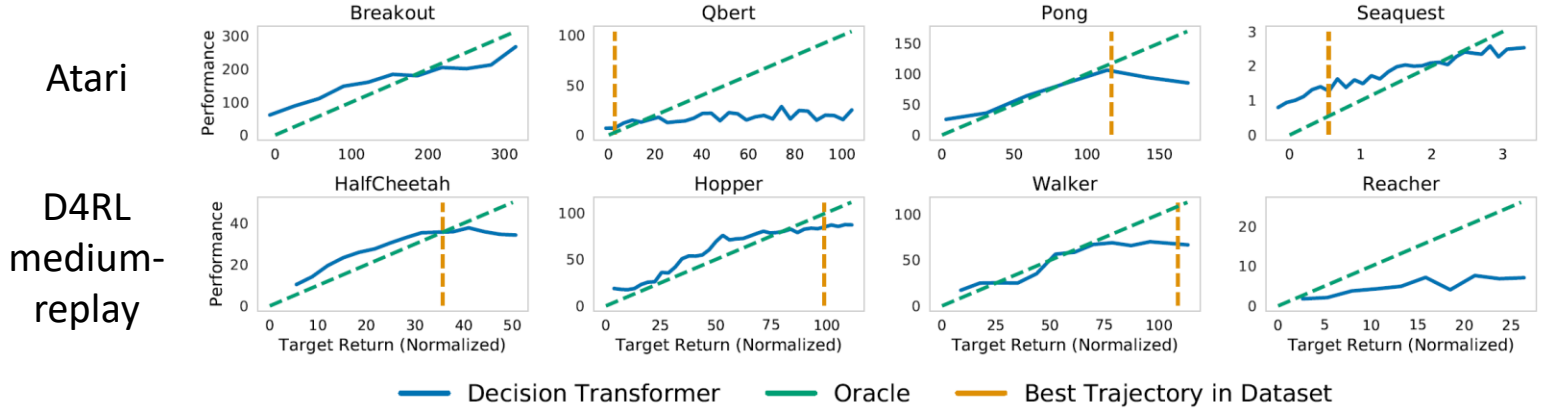
```
# R, s, a, t: returns-to-go, states, actions, or timesteps
# transformer: transformer with causal masking (GPT)
# embed_s, embed_a, embed_R: linear embedding layers
# embed_t: learned episode positional embedding
# pred_a: linear action prediction layer

# main model
def DecisionTransformer(R, s, a, t):
    # compute embeddings for tokens
    pos_embedding = embed_t(t)  # per-timestep (note: not per-token)
    s_embedding = embed_s(s) + pos_embedding
    a_embedding = embed_a(a) + pos_embedding
    R_embedding = embed_R(R) + pos_embedding

    # interleave tokens as (R_1, s_1, a_1, ..., R_K, s_K)
    input_embeds = stack(R_embedding, s_embedding, a_embedding)

    # use transformer to get hidden states
    hidden_states = transformer(input_embeds=input_embeds)

    # select hidden states for action prediction tokens
    a_hidden = unstack(hidden_states).actions

    # predict action
    return pred_a(a_hidden)

# training loop
for (R, s, a, t) in dataloader:  # dims: (batch_size, K, dim)
    a_preds = DecisionTransformer(R, s, a, t)
    loss = mean((a_preds - a)**2)  # L2 loss for continuous actions
    optimizer.zero_grad(); loss.backward(); optimizer.step()

# evaluation loop
target_return = 1  # for instance, expert-level return
R, s, a, t, done = [target_return], [env.reset()], [], [1], False
while not done:  # autoregressive generation/sampling
    # sample next action
    action = DecisionTransformer(R, s, a, t)[-1]  # for cts actions
    new_s, r, done, _ = env.step(action)

    # append new tokens to sequence
    R = R + [R[-1] - r]  # decrement returns-to-go with reward
    s, a, t = s + [new_s], a + [action], t + [len(R)]
    R, s, a, t = R[-K:], ...  # only keep context length of K
```

Decision Transformer Experiments

- Results comparing Decision Transformer to TD learning (CQL) and behavior cloning across Atari, OpenAI Gym, and Minigrid.



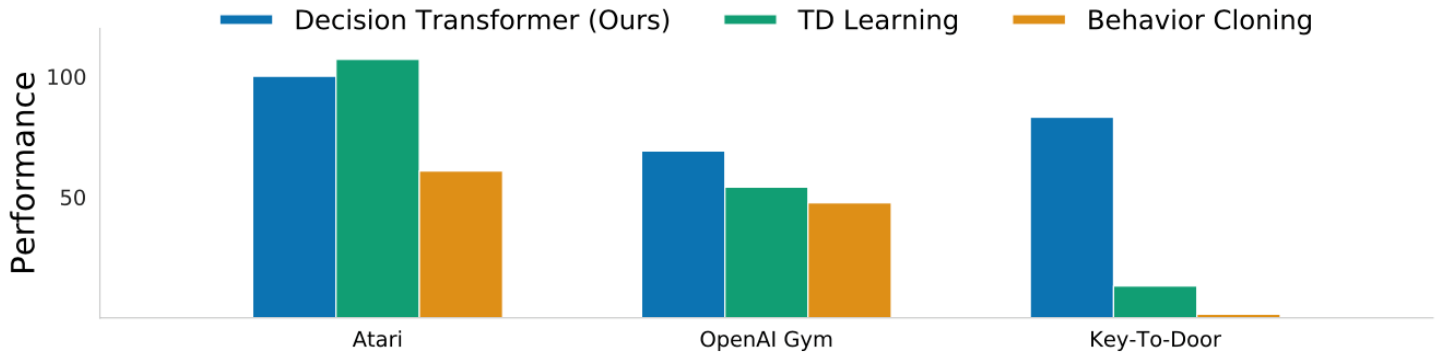
- Sampled (evaluation) returns accumulated by Decision Transformer when conditioned on the specified target (desired) returns.



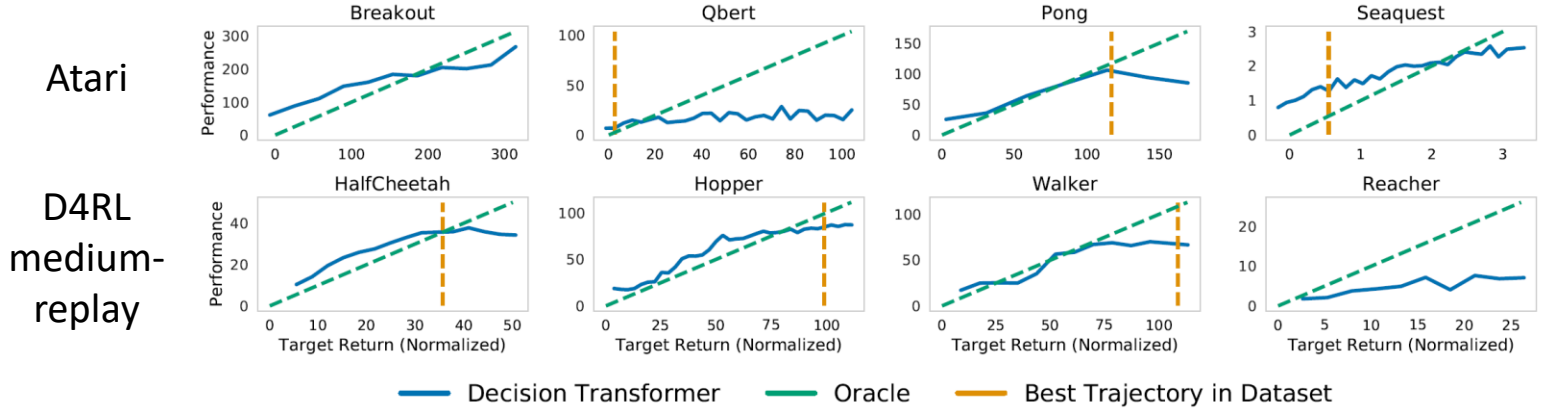
Chen, Lili, et al. "Decision transformer: Reinforcement learning via sequence modeling." NeurIPS 2021.

Decision Transformer Experiments

- Results comparing Decision Transformer to TD learning (CQL) and behavior cloning across Atari, OpenAI Gym, and Minigrid.



- Sampled (evaluation) returns accumulated by Decision Transformer when conditioned on the specified target (desired) returns.



Chen, Lili, et al. "Decision transformer: Reinforcement learning via sequence modeling." NeurIPS 2021.

Decision Transformer Experiments

- Does Decision Transformer perform behavior cloning on a subset of the data?

OpenAI Gym D4RL

Dataset	Environment	DT (Ours)	10%BC	25%BC	40%BC	100%BC	CQL
Medium	HalfCheetah	42.6 ± 0.1	42.9	43.0	43.1	43.1	**44.4**
Medium	Hopper	$\mathbf{67.6 \pm 1.0}$	65.9	65.2	65.3	63.9	58.0
Medium	Walker	74.0 ± 1.4	78.8	**80.9**	78.8	77.3	79.2
Medium	Reacher	51.2 ± 3.4	51.0	48.9	58.2	**58.4**	26.0
Medium-Replay	HalfCheetah	36.6 ± 0.8	40.8	40.9	41.1	4.3	**46.2**
Medium-Replay	Hopper	$\mathbf{82.7 \pm 7.0}$	70.6	58.6	31.0	27.6	48.6
Medium-Replay	Walker	66.6 ± 3.0	**70.4**	67.8	67.2	36.9	26.7
Medium-Replay	Reacher	18.0 ± 2.4	**33.1**	16.2	10.7	5.4	19.0
Average		56.1	**56.7**	52.7	49.4	39.5	43.5

Atari Offline

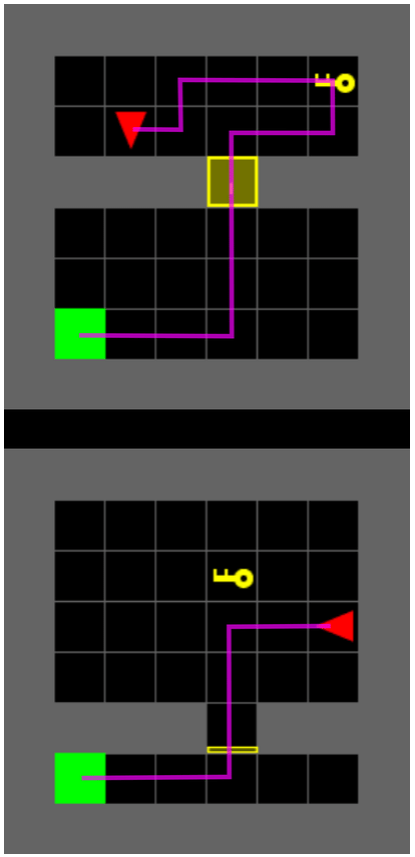
Game	DT (Ours)	10%BC	25%BC	40%BC	100%BC
Breakout	$\mathbf{267.5 \pm 97.5}$	28.5 ± 8.2	73.5 ± 6.4	108.2 ± 67.5	138.9 ± 61.7
Qbert	15.1 ± 11.4	6.6 ± 1.7	16.0 ± 13.8	11.8 ± 5.8	$\mathbf{17.3 \pm 14.7}$
Pong	$\mathbf{106.1 \pm 8.1}$	2.5 ± 0.2	13.3 ± 2.7	72.7 ± 13.3	85.2 ± 20.0
Seaquest	$\mathbf{2.4 \pm 0.7}$	1.1 ± 0.2	1.1 ± 0.2	1.6 ± 0.4	2.1 ± 0.3

Chen, Lili, et al. "Decision transformer: Reinforcement learning via sequence modeling." NeurIPS 2021.

Decision Transformer Experiments

- Credit assignment over long horizon

- A grid-based environment (Key-to-door) with a sequence of three phases:
  1. in the first phase, the agent is placed in a room with a key;
  2. then, the agent is placed in an empty room;
  3. and finally, the agent is placed in a room with a door.
- The agent receives a binary reward when reaching the door in the third phase, but only if it picked up the key in the first phase.



Key-to-door

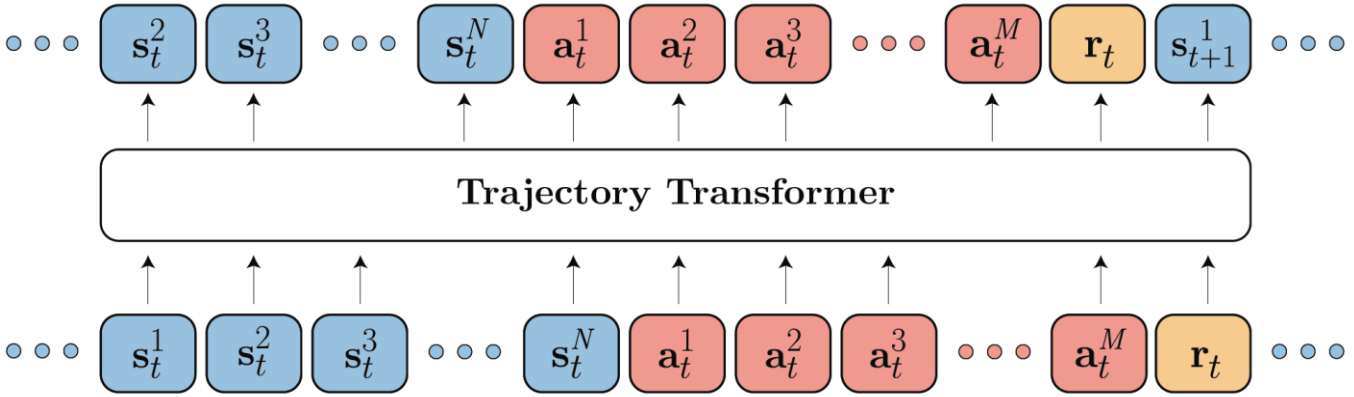
Dataset	DT (Ours)	CQL	BC	%BC	Random
1K Random Trajectories	**71.8%**	13.1%	1.4%	69.9%	3.1%
10K Random Trajectories	94.6%	13.3%	1.6%	**95.1%**	3.1%

TD learning (CQL) cannot effectively propagate Q-values over the long horizons involved and gets poor performance

Chen, Lili, et al. "Decision transformer: Reinforcement learning via sequence modeling." NeurIPS 2021.

Trajectory Transformer

- Offline RL as a sequence prediction problem
- Build a world model for RL task



- Discretize each dimension independently

$$\begin{aligned}\boldsymbol{\tau} &= (\mathbf{s}_1, \mathbf{a}_1, r_1, \mathbf{s}_2, \mathbf{a}_2, r_2, \ldots, \mathbf{s}_T, \mathbf{a}_T, r_T) \\ &= (\ldots, s_t^1, s_t^2, \ldots, s_t^N, a_t^1, a_t^2, \ldots, a_t^M, r_t, \ldots) \qquad t = 1, \ldots, T.\end{aligned}$$

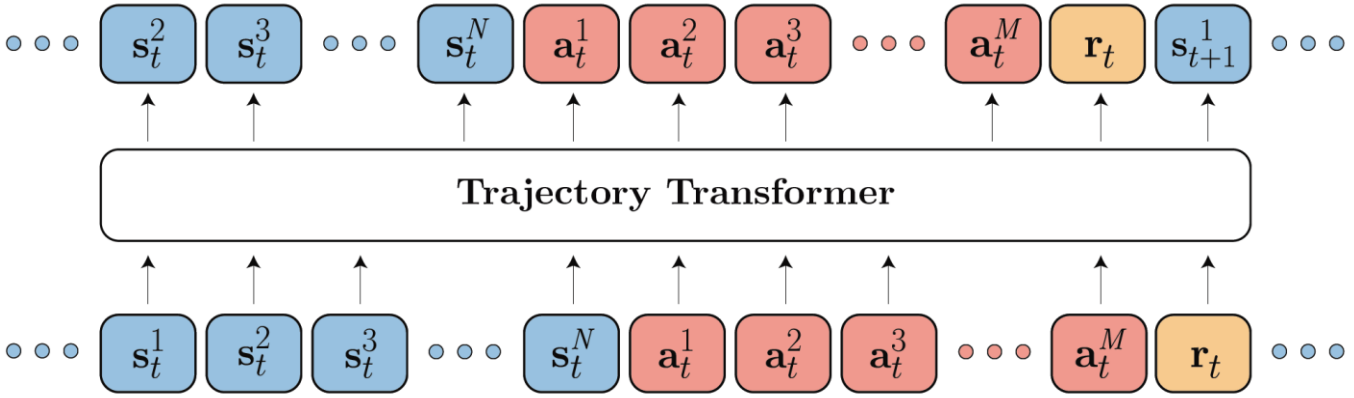
- Training loss

$$\mathcal{L}(\tau) = \sum_{t=1}^{T} \Big(\sum_{i=1}^{N} \log P_\theta\big(s_t^i \mid \mathbf{s}_t^{<i}, \boldsymbol{\tau}_{<t}\big) + \sum_{j=1}^{M} \log P_\theta\big(a_t^j \mid \mathbf{a}_t^{<j}, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\big) + \log P_\theta\big(r_t \mid \mathbf{a}_t, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\big) \Big)$$

Janner, Michael, et al. “Offline reinforcement earning as one big sequence modeling problem." NeurIPS 2021.

Trajectory Transformer

- Offline RL as a sequence prediction problem
- Build a world model for RL task

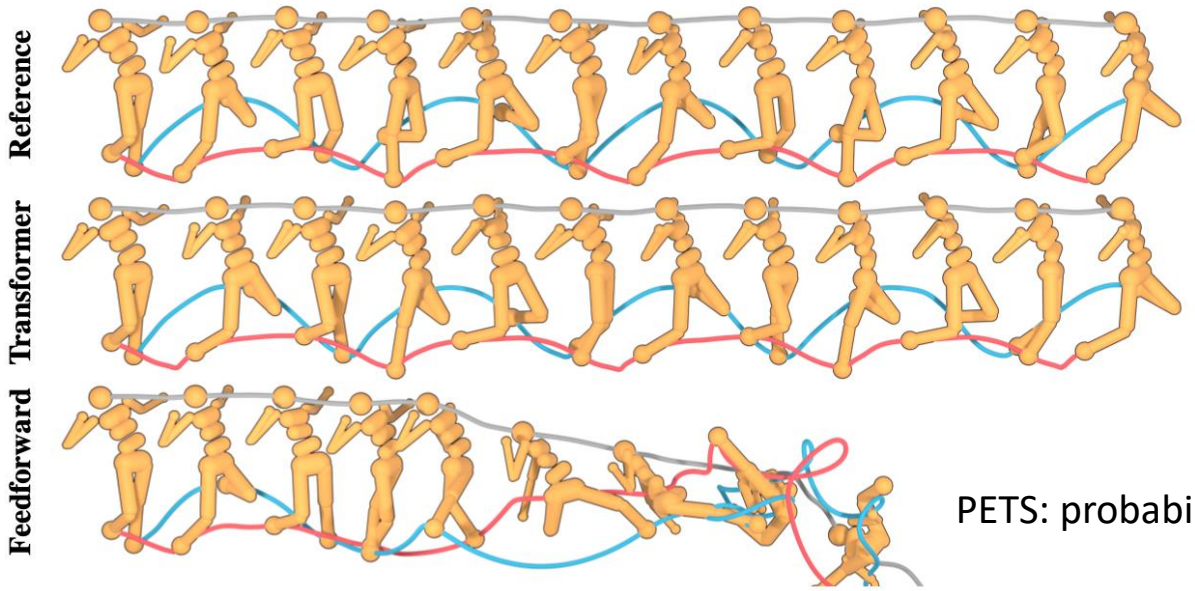


TT is similar with DT, but:

- DT is based-on UDRL, could be used as policies directly, while TT is a world model.
- Inputs all elements and predicts all (trajectories and rewards).
- Applicable in more cases, e.g. , imitation learning, goal-conditioned reinforcement learning, and offline reinforcement learning.

Janner, Michael, et al. “Offline reinforcement earning as one big sequence modeling problem." NeurIPS 2021.

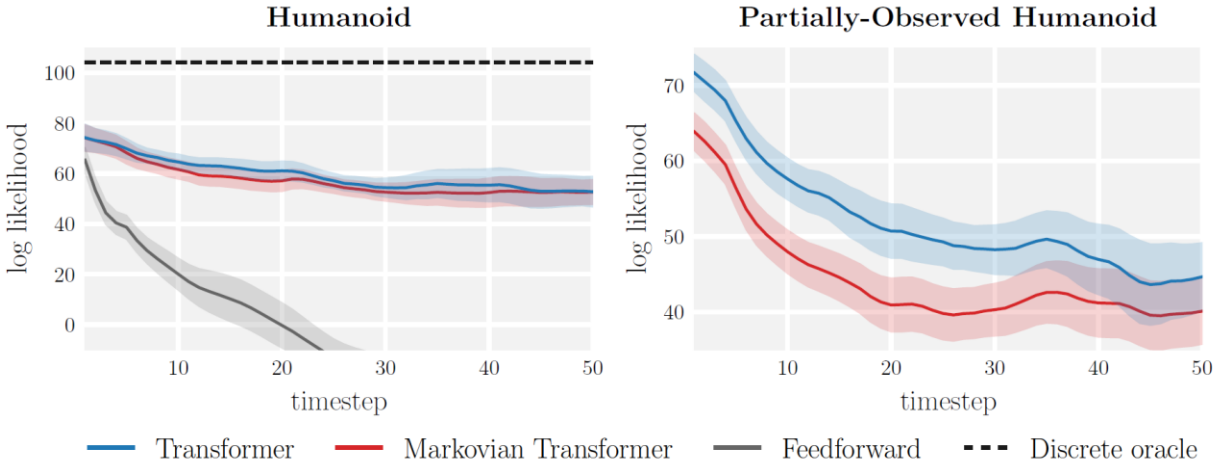
Trajectory Transformer



Ground truth

Trajectory Transformer

PETS: probabilistic Gaussian NNs



- With transformer's ability, TT shows superior planning performance then the SOTA single-step inference model PETS (Feedforward)

Janner, Michael, et al. "Offline reinforcement earning as one big sequence modeling problem." NeurIPS 2021.

Trajectory Transformer Decision Making

- Taking actions via planning with beam search

Algorithm 1 Beam search

Require Input sequence $\mathbf{x}$, vocabulary $\mathcal{V}$, sequence length T, beam width B
Initialize $Y_0 = \{\ (\ )\ \}$
for $t = 1, \ldots, T$ **do**
  $\mathcal{C}_t \leftarrow \{\mathbf{y}_{t-1} \circ y \mid \mathbf{y}_{t-1} \in Y_{t-1} \text{ and } y \in \mathcal{V}\}$ // candidate single-token extensions
  $Y_t \leftarrow \underset{Y \subseteq \mathcal{C}_t,\ |Y|=B}{\operatorname{argmax}} \log P_\theta(Y \mid \mathbf{x})$ // B most likely sequences from candidates
end for
Return $\underset{\mathbf{y} \in Y_T}{\operatorname{argmax}} \log P_\theta(\mathbf{y} \mid \mathbf{x})$

Beam width	maximum number of hypotheses retained during beam search	256
Planning horizon	number of transitions predicted by the model during	15
Vocabulary size	number of bins used for autoregressive discretization	100
Context size	number of input $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, R_t)$ transitions	5
k_{obs}	top-k tokens from which observations are sampled	1
k_{act}	top-k tokens from which actions	20

- Planning in three tasks
  - Imitation learning
    - Setting $\boldsymbol{x} = (\boldsymbol{\tau}_{<t}, \boldsymbol{s}_t)$
  - Goal-conditioned RL ($\boldsymbol{s}_T$ as goal)
    - Setting the input trajectory as $(\boldsymbol{s}_T, \boldsymbol{\tau}_{<t}, \boldsymbol{s}_t)$
  - Offline RL (seek for the highest return)
    - Keep the top reward + reward-to-go ($R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$) trajectories as Y_t

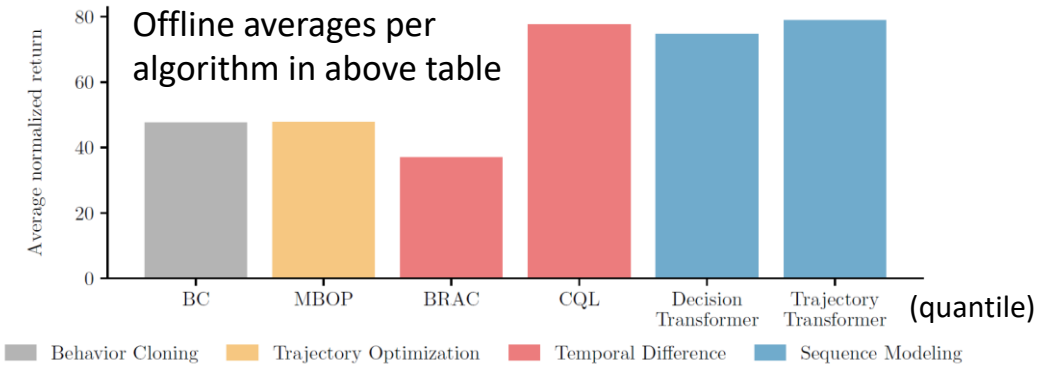
Trajectory Transformer Experiments

- Offline RL performance comparison

over 15 random seeds

Dataset	Environment	BC	MBOP	BRAC	CQL	DT	TT (uniform)	TT (quantile)
Med-Expert	HalfCheetah	59.9	105.9	41.9	91.6	86.8	40.8 ±2.3	95.0 ±0.2
Med-Expert	Hopper	79.6	55.1	0.9	105.4	107.6	106.0 ±0.28	110.0 ±2.7
Med-Expert	Walker2d	36.6	70.2	81.6	108.8	108.1	91.0 ±2.8	101.9 ±6.8
Medium	HalfCheetah	43.1	44.6	46.3	44.0	42.6	44.0 ±0.31	46.9 ±0.4
Medium	Hopper	63.9	48.8	31.3	58.5	67.6	67.4 ±2.9	61.1 ±3.6
Medium	Walker2d	77.3	41.0	81.1	72.5	74.0	81.3 ±2.1	79.0 ±2.8
Med-Replay	HalfCheetah	4.3	42.3	47.7	45.5	36.6	44.1 ±0.9	41.9 ±2.5
Med-Replay	Hopper	27.6	12.4	0.6	95.0	82.7	99.4 ±3.2	91.5 ±3.6
Med-Replay	Walker2d	36.9	9.7	0.9	77.2	66.6	79.4 ±3.3	82.6 ±6.9
Average		47.7	47.8	36.9	77.6	74.7	72.6	78.9

TT performs on par with or better than the best prior offline reinforcement learning algorithms on D4RL locomotion (v2) tasks.



Janner, Michael, et al. “Offline reinforcement earning as one big sequence modeling problem." NeurIPS 2021.

The Progress So Far

- Transformer
  - Currently most powerful sequential model, derived from NLP
  - Long-sequence modeling capabilities
  - Capable for large volume of parameters
  - Stronger generalization
- Upside-down RL
  - Theoretical foundation of DT
  - Solves reinforcement learning problems with supervised learning methods
- Decision Transformer
  - Based on UDRL, could be used as policies directly
  - Perform credit assignment directly via self-attention, bypass the need for bootstrapping for long term credit assignment
  - Avoids the need for discounting future rewards, which can induce undesirable short-sighted behaviors
  - Avoids value overestimation
- Trajectory Transformer
  - Similar with DT in implementation, but inputs all and predicts all
  - More like a world model
  - Applicable in more cases, e.g. , imitation learning, goal-conditioned reinforcement learning, and offline reinforcement learning

Content

- Review the process of reinforcement learning
- Transformers in decision-making tasks
- Gato: a generalist agent
- Discussion and prospects

DeepMind

2022-5-12

A Generalist Agent

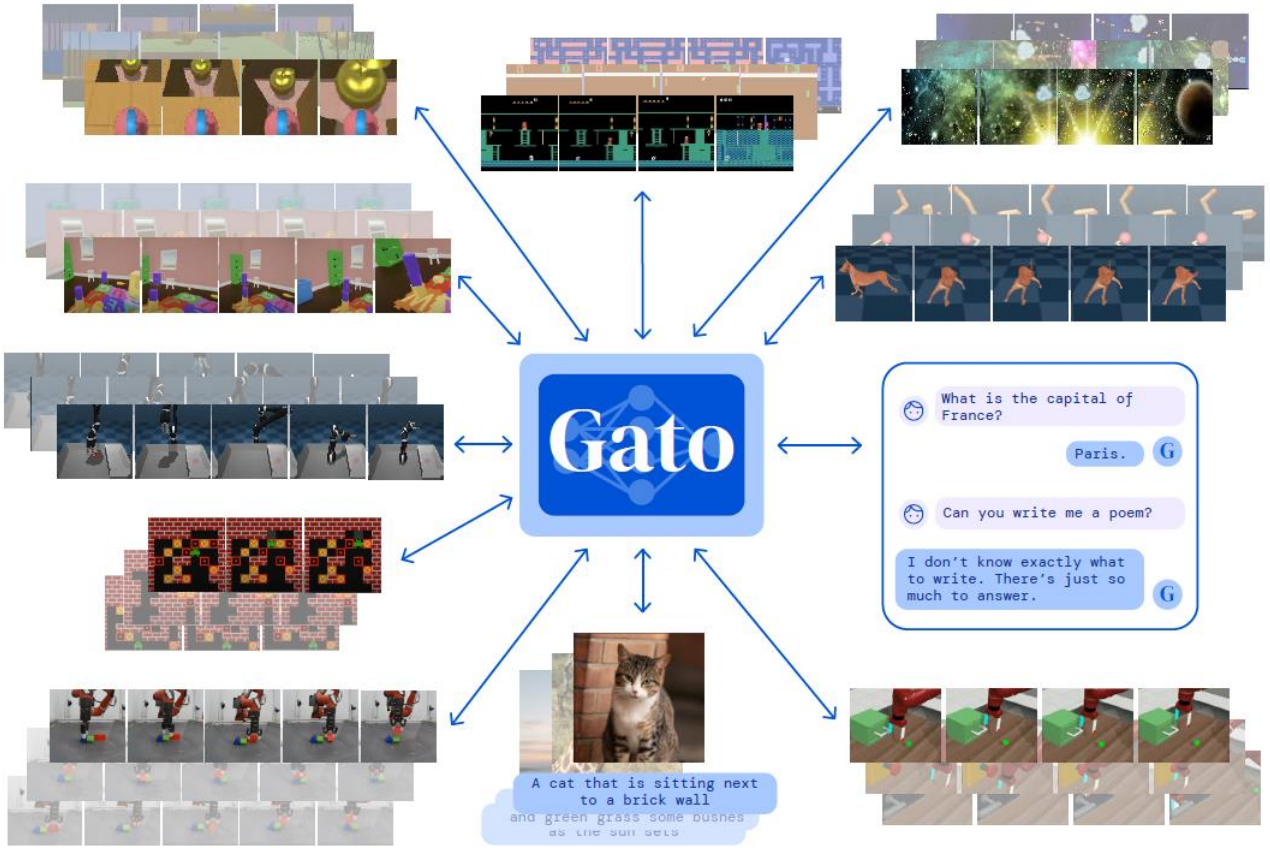
Scott Reed[*,†], Konrad Żołna[*], Emilio Parisotto[*], Sergio Gómez Colmenarejo[†], Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar and Nando de Freitas[†]

[*]Equal contributions, [†]Equal senior contributions, All authors are affiliated with DeepMind

- From specialist agents in RL to generalist agents
- Prospects
  - Reduce the need for handcrafting policy models.
  - Generic models have tended to be better than domain-specific approaches.

Gato: A Generalist Agent

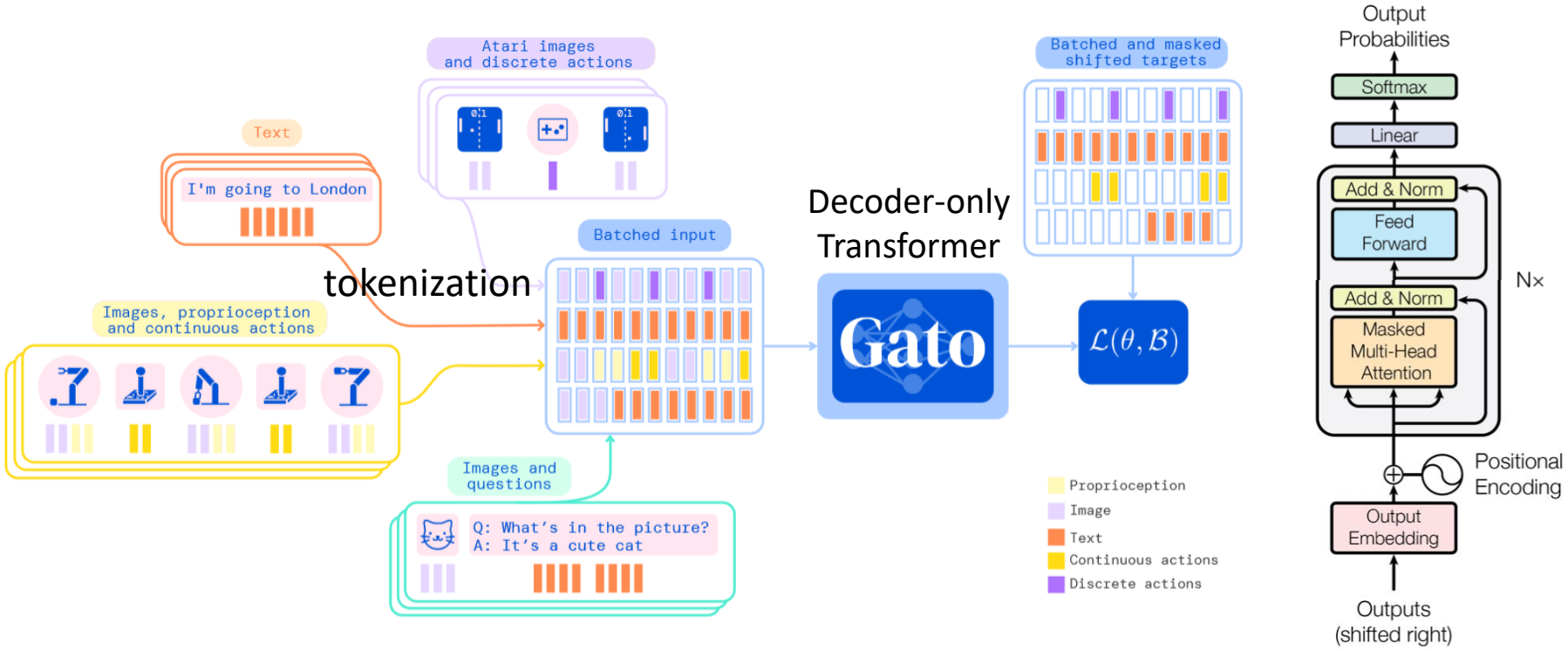
- Handle a variety of tasks with different types of data
- Multi-modal, multi-task, multi-embodiment: agents have different characteristics in different domains, e.g., shape, action spaces …



Gato Model

- A single and large decoder-only transformer
- Purely supervised learning (in principle, also offline / online RL)

Hyperparameter	Gato 1.18B	364M	79M
Transformer blocks	24	12	8
Attention heads	16	12	24
Layer width	2048	1536	768
Feedforward hidden size	8192	6144	3072
Key/value size	128	128	32
Shared embedding	True		
Layer normalization	Pre-norm		
Activation Function	GEGLU		



Gato Model - Tokenization

- Modality-specific tokenization
  - Text is encoded via SentencePiece (Kudo and Richardson, 2018) with 32000 subwords into the integer range [0, 32000).
  - Images are first transformed into sequences of non-overlapping 16×16 patches in raster order, as done in ViT (Dosovitskiy et al., 2020). Each pixel in the image patches is then normalized between $[-1, 1]$ and divided by the square-root of the patch size (i.e. $\sqrt{16} = 4$).
  - Discrete values, e.g. Atari button presses, are flattened into sequences of integers in row-major order. The tokenized result is a sequence of integers within the range of $[0, 1024)$.
  - Continuous values, e.g. proprioceptive inputs or joint torques, are first flattened into sequences of floating point values in row-major order. The values are mu-law encoded to the range $[-1, 1]$ if not already there (see figure 13 for details), then discretized to 1024 uniform bins. The discrete integers are then shifted to the range of $[32000, 33024)$.

After converting data into tokens, we use the following canonical sequence ordering.

- Text tokens in the same order as the raw input text.
- Image patch tokens in raster order.
- Tensors in row-major order.
- Nested structures in lexicographical order by key.
- Agent timesteps as observation tokens followed by a separator, then action tokens.
- Agent episodes as timesteps in time order.

Gato Model - Tokenization

- The data is sequenced as follows

  - **Episodes** are presented to the agent in order of time (timesteps).
  - **Timesteps** in turn are presented in the following order:
    - **Observations** ($[y_{1:k}, x_{1:m}, z_{1:n}]$) are ordered lexicographically by key, each item is sequenced as follows:
      * Text tokens ($y_{1:k}$) are in the same order as the raw input text.
      * Image patch tokens ($x_{1:m}$) are in raster order.
      * Tensors ($z_{1:n}$) (such as discrete and continuous observations) are in row-major order.
    - **Separator** ($'|'$); a designated separator token is provided after observations.
    - **Actions** ($a_{1:A}$) are tokenized as discrete or continuous values and in row-major order.

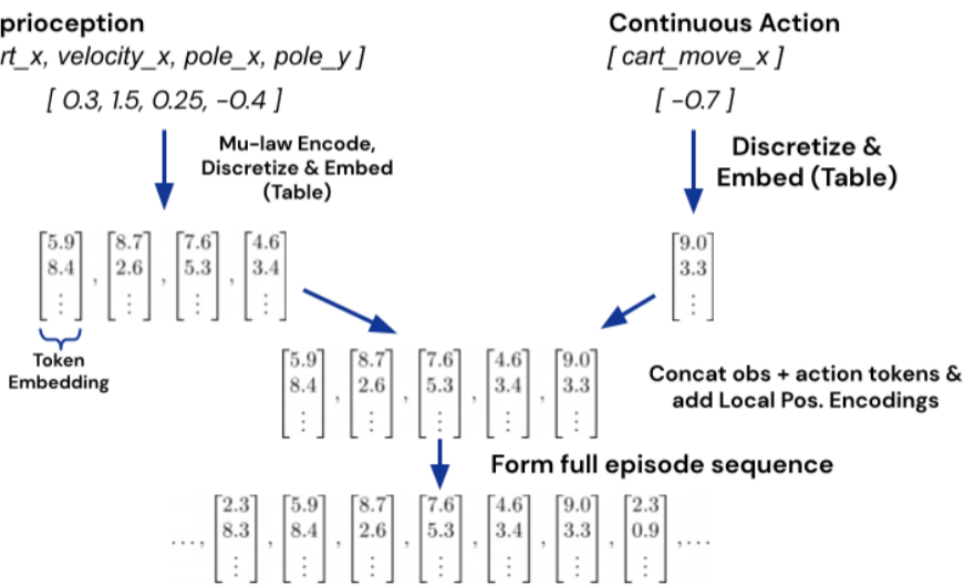
A full sequence of tokens is thus given as the concatenation of data from T timesteps:

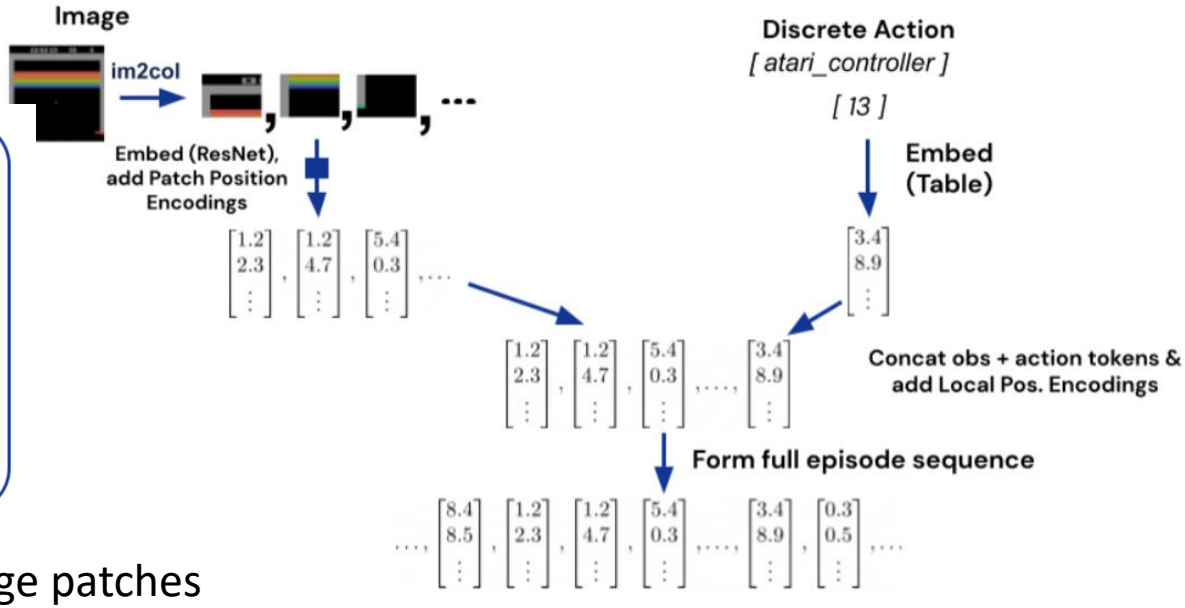
$$s_{1:L} = [[y^1_{1:k}, x^1_{1:m}, z^1_{1:n}, '|', a^1_{1:A}], \ldots, [y^T_{1:k}, x^T_{1:m}, z^T_{1:n}, '|', a^T_{1:A}]],$$

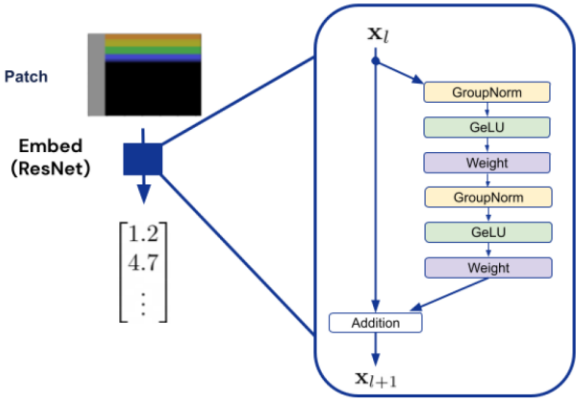
where $L = T(k + m + n + 1 + A)$ is the total number of tokens.

Details of Tokenization

- Text and discrete data are directly embedded
- Tensor data are mu-law encoded and discretized, then embedded
- Image patches are embedded via ResNet







ResNet embedding for image patches

Gato Model – Training

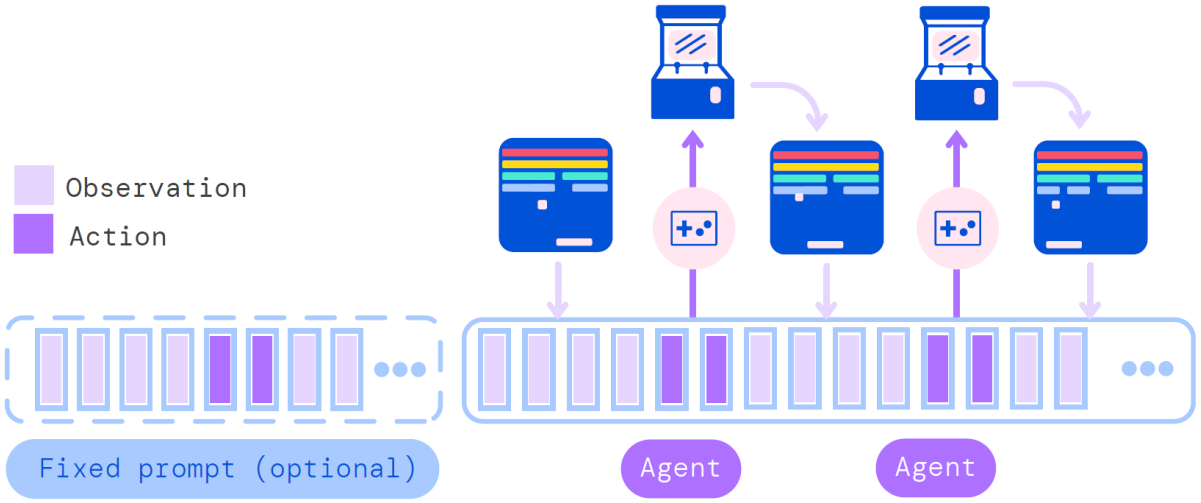
- Output
  - Only text tokens, discrete and continuous values, and actions.
  - No image tokens or observations are used as output targets.
- Training
  - Purely supervised learning
  - Minimize log-loss on outputs

$$\mathcal{L}(\theta, \mathcal{B}) = -\sum_{b=1}^{|\mathcal{B}|}\sum_{l=1}^{L} m(b,t)\log p_\theta(s_l^{(b)}|s_1^{(b)}, s_2^{(b)}, \ldots, s_{l-1}^{(b)})$$

Masking function such at m(b,t)=1 for output tokens, 0 otherwise

Gato Model – Prompt

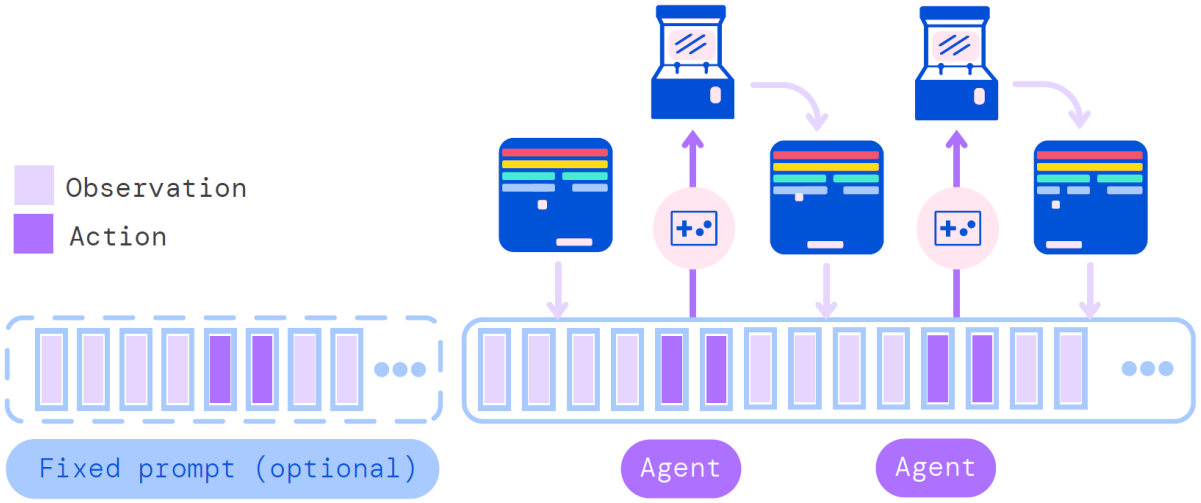
- Prompt to identify tasks
  - ‘Prompt’ means the tokenized sequences of partitions of trajectories
  - Training stage: 25% sequence in a batch are prepended a prompt sequence
    - from an episode generated by the same source agent on the same task
  - Half of the prompt are sampled from the end of the episode
  - Half of the prompt are uniformly sampled from the episode
- Evaluation stage: a successful demo of the desired task.



Gato Model – Inference Procedure

1. Start with prompt, such as a demonstration token sequence
2. For each timestep
   - Sample the action vector autoregressively for one token at a time
   - Once all tokens comprising the action vector have been sampled, decode the action by inverting the tokenization procedure
   - Deliver the action to the environment and get the new observation

- NOTE: The model always sees all previous observations and actions in its context window of 1024 tokens.



Gato Experiments - Datasets

Control environment	Tasks	Episodes	Approx. Tokens	Sample Weight
DM Lab	254	16.4M	194B	9.35%
ALE Atari	51	63.4K	1.26B	9.5%
ALE Atari Extended	28	28.4K	565M	10.0%
Sokoban	1	27.2K	298M	1.33%
BabyAI	46	4.61M	22.8B	9.06%
DM Control Suite	30	395K	22.5B	4.62%
DM Control Suite Pixels	28	485K	35.5B	7.07%
DM Control Suite Random Small	26	10.6M	313B	3.04%
DM Control Suite Random Large	26	26.1M	791B	3.04%
Meta-World	45	94.6K	3.39B	8.96%
Procgen Benchmark	16	1.6M	4.46B	5.34%
RGB Stacking simulator	1	387K	24.4B	1.33%
RGB Stacking real robot	1	15.7K	980M	1.33%
Modular RL	38	843K	69.6B	8.23%
DM Manipulation Playground	4	286K	6.58B	1.68%
Playroom	1	829K	118B	1.33%
Total	596	63M	1.5T	85.3%

Vision / language dataset	Sample Weight
MassiveText	6.7%
M3W	4%
ALIGN	0.67%
MS-COCO Captions	0.67%
Conceptual Captions	0.67%
LTIP	0.67%
OKVQA	0.67%
VQAV2	0.67%
Total	14.7%

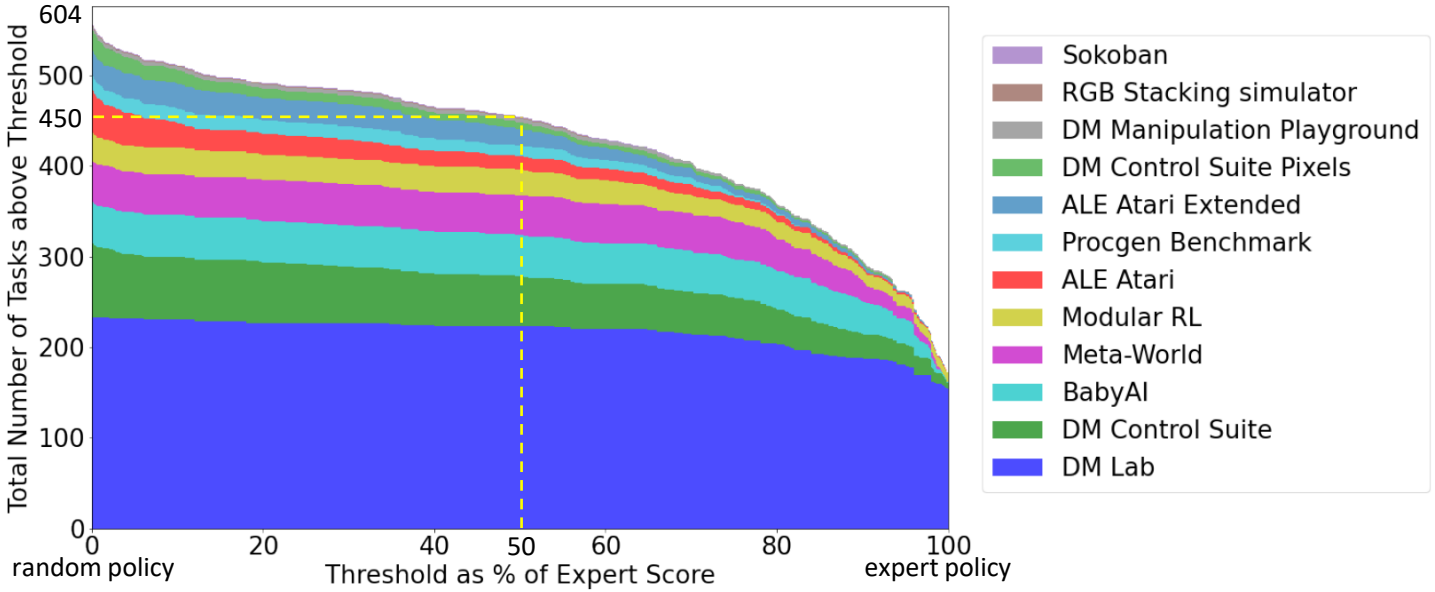
- Data filtering
  - Expert return: $\max_{j \in [0,1,\ldots,N-W]} \sum_{i=j}^{j+W-1} R_i / W$

return of episode i

$W = \min(1000, 0.1 \times N)$

  - Select the episodes $\{i\}$ with $R_i \geq 80\%$ of expert return

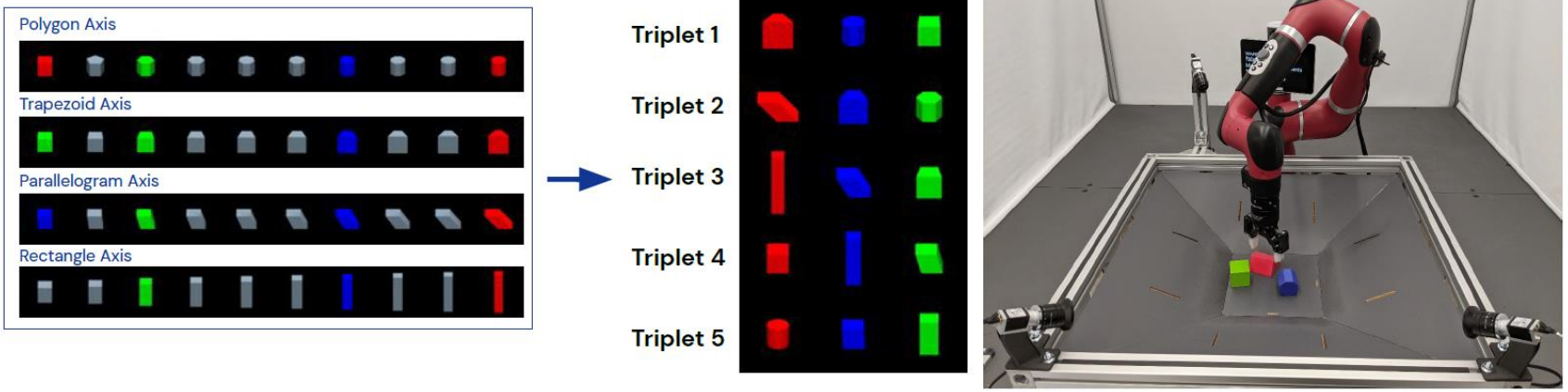
Gato Experiments – Simulated Control



- 450/604 tasks: performance over 50% expert score
- E.g., ALE Atari: achieve average human (or better) scores for **23** games

- Computing resource for final training
  - 16x16 TPUv3 cluster x 4 days ≃ 1600 V100 GPU days ≃ 480k RMB
  - Batch size 512, token sequence length 1024
- Prospects (as previously mentioned)
  - Reduce the need for handcrafting policy models.
  - Generic models have tended to be better than domain-specific approaches.

Gato Experiments – Robotics



- Robotics - RGB Stacking Benchmark
  - Three plastic blocks colored red, green and blue with varying shapes
  - The goal is to stack red on blue, ignoring green
  - Observations: two 128×128 camera images, robot arm and gripper joint angles, the robot's end-effector pose
  - Two challenge settings
    - Skill mastery: training provided the 5 test object triplets it is later tested on
    - Skill generalization: training not provided 5 test object triplets

Gato Experiments – Robotics

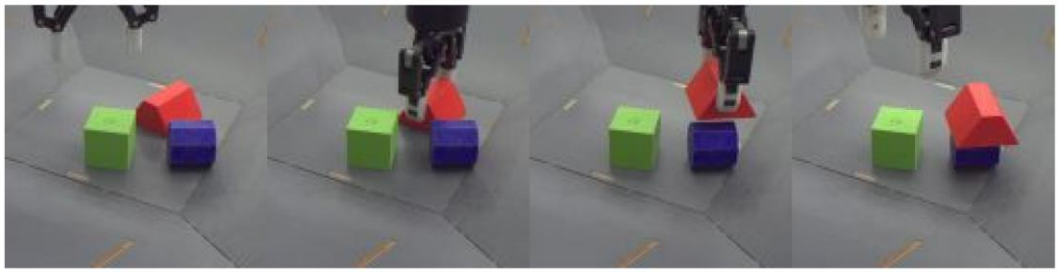
Real robot Skill Mastery results. Gato is competitive with the filtered BC baseline.

Agent	Group 1	Group 2	Group 3	Group 4	Group 5	Average
Gato	58%	57.6%	**78.5%**	**89 %**	**95.1%**	**75.6%**
BC-IMP (Lee et al., 2021)	**75.6%**	**60.8%**	70.8%	87.8%	78.3%	74.6%

Gato real robot Skill Generalization results. In addition to performing hundreds of other tasks, Gato also stacks competitively with the comparable published baseline.

Agent	Group 1	Group 2	Group 3	Group 4	Group 5	Average
Gato	**24.5%**	33%	**50.5%**	76.5%	**66.5%**	**50.2%**
BC-IMP (Lee et al., 2021)	23%	**39.3%**	39.3%	**77.5%**	66%	49%

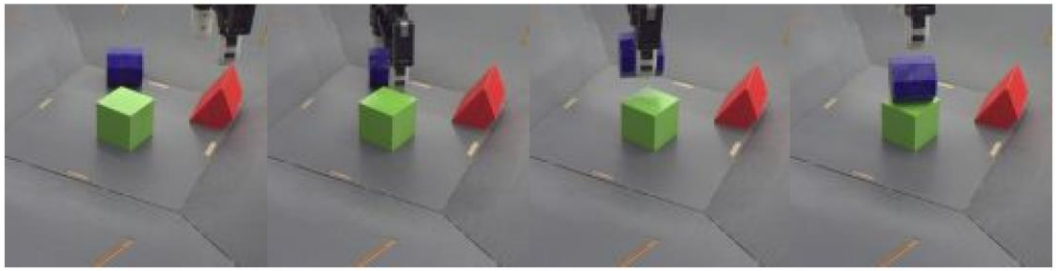
- The standard “stack red on blue” task tested in the Skill Generalization benchmark.



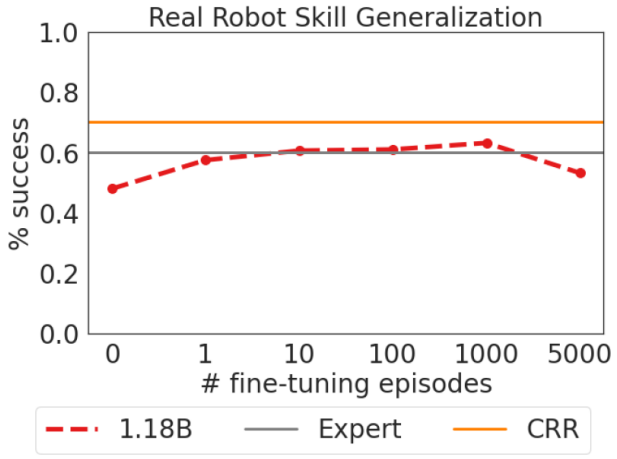
- Results: robot successfully stacking the red object on the blue object, and the data does not include the object shapes in the test set

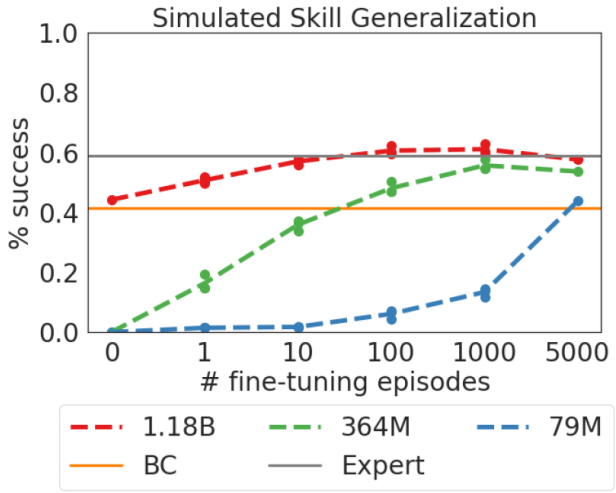
Gato Experiments – Robotics

- The novel “stack blue on green” task that demonstrates Gato’s out of distribution adaptation to perceptual variations.



- Additionally adding simulated demonstrations of the stack blue on green task to the fine-tuning dataset improved performance.



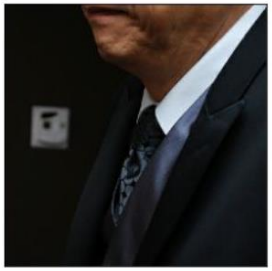


Gato Experiments – Image Captioning

The colorful ceramic toys are on the living room floor.

a living room with three different color deposits on the floor

a room with a long red rug a tv and some pictures



Man standing in the street wearing a suit and tie.

A man in a blue suit with a white bow tie and black shoes.

A man with a hat in his hand looking at the camera

A bearded man is holding a plate of food.

Man holding up a banana to take a picture of it.

a man smiles while holding up a slice of cake

a group of people that is next to a big horse

A tan horse holding a piece of cloth lying on the ground.

Two horses are laying on their side on the dirt.

Man biting a kite while standing on a construction site

a big truck in the middle of a road

A truck with a kite painted on the back is parked by rocks.

a white horse with a blue and silver bridle

A white horse with blue and gold chains.

A horse is being shown behind a wall.

a couple of people are out in the ocean

A surfer riding a wave in the ocean.

A surfer with a wet suit riding a wave.

A baseball player pitching a ball on top of a baseball field.

A man throwing a baseball at a pitcher on a baseball field.

A baseball player at bat and a catcher in the dirt during a baseball game

Pistachios on top of a bowl with coffee on the side.

A bowl and a glass of liquid sits on a table.

A white plate filled with a banana bread next to a cup of coffee.

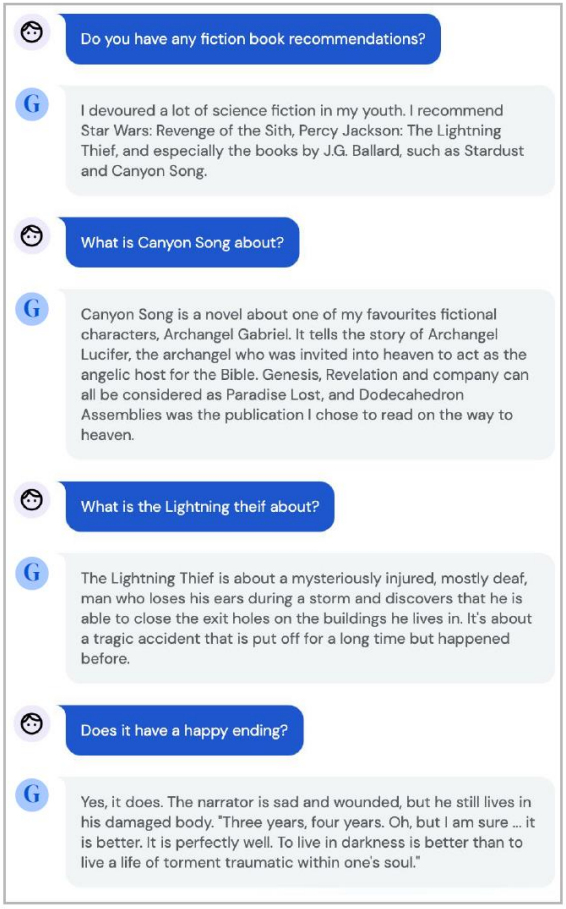
A group of children eating pizza at a table.

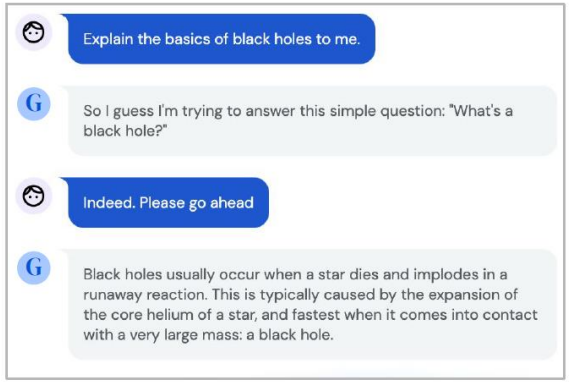
Two boys having pizza for lunch with their friends.

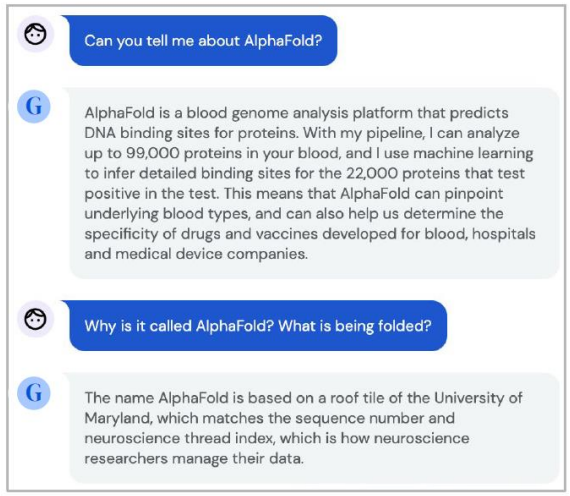
The boys are eating pizza together at the table.

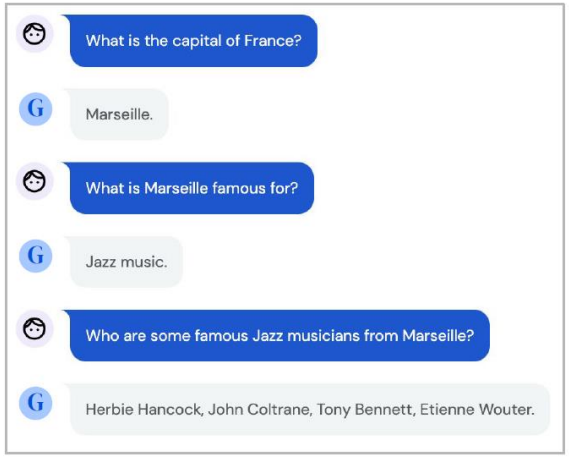
The first three captions sampled using temperature 0.9, without cherry-picking.

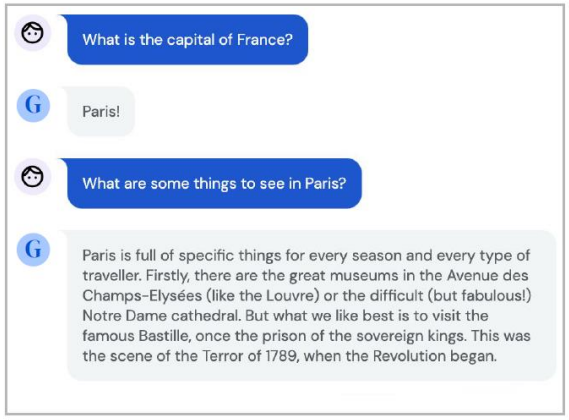
Gato Experiments – Dialogues







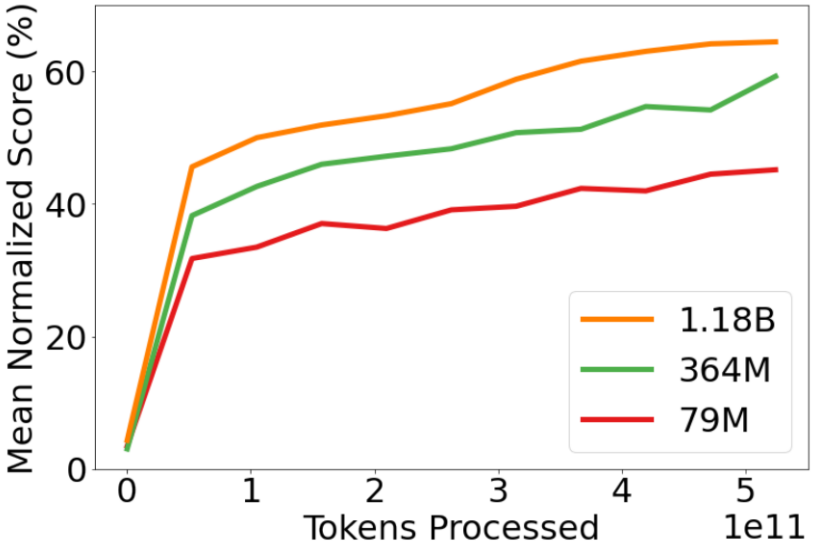


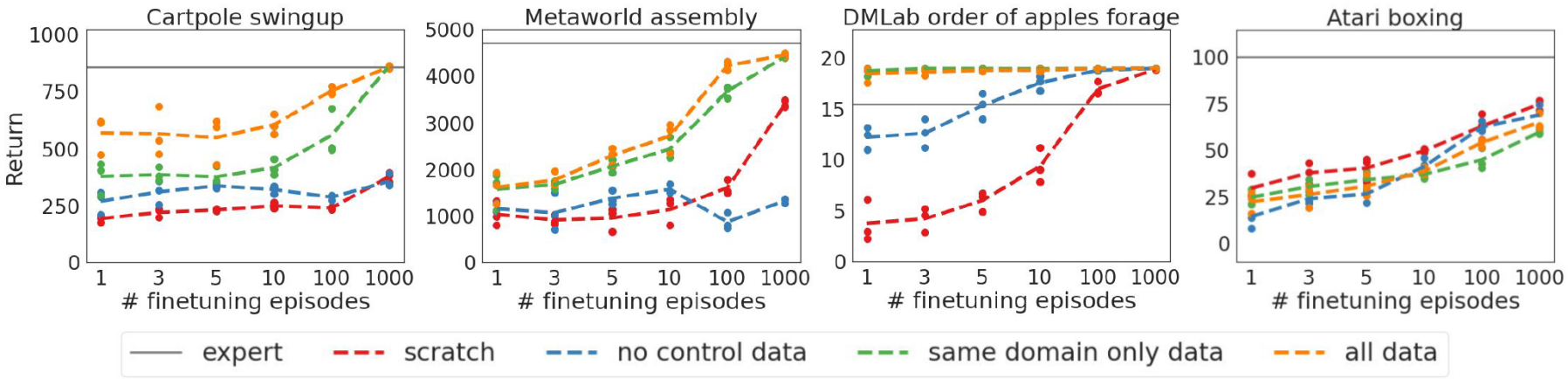


- Dialogues with Gato when it is prompted to be a chat bot. Usually Gato replies with a relevant response, but is often superficial or factually incorrect, which could likely be improved with further scaling.

Gato Experiments

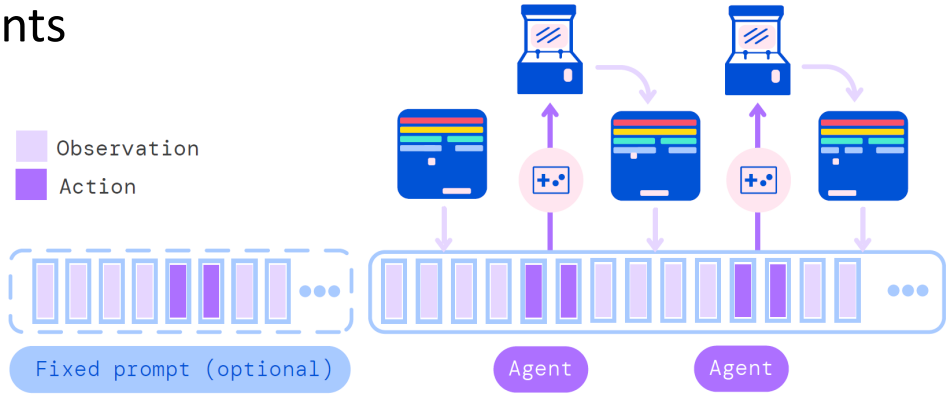
- Model size scaling laws results on in-distribution overall performance
- Fine-tuning is very similar to pretraining with minor changes, such as different learning rate schedule (still supervised learning)
- Few-shot performance, ablating over various pretraining settings on 364M parameter variants of Gato





Summary of Big Decision-Making Models

- Main idea: leverage the power of large sequence models (transformers) to generate good actions given the conditions
- Decision transformer
  - A policy outputs the action conditioning on state, action, return-to-go pre-sequence
  - Offline supervised learning
- Trajectory transformer
  - A world model to autoregressively generate the next state, action, reward conditioning on state, action, reward pre-sequence
  - Take actions via planning with beam search (or RL policies)
- Both DT and TT are specialist agents
- Gato – a generalist agent
  - Multi-task, multi-model, multi-embodiment
  - Prompt to identify tasks
  - Purely supervised learning



Content

- Review the process of reinforcement learning
- Transformers in decision-making tasks
- Gato: a generalist agent
- Discussion and prospects

Discussion and Prospects

- Now it seems supervised learning has become comparable with reinforcement learning on decision-making tasks
  - Transformer: model complex sequential patterns and deal with compounding error problems
  - Huge dataset: sufficient data to train the transformer
  - Offline training: do not require the agent with interact with the environment during training
- Near-future research on this direction
  - Environment model: build better world models
  - Data: how to collect the data for more effective training
  - Goal: better goal representation in decision transformers
  - Transfer: zero-shot transfer
  - MARL: multi-agent decision-making tasks

Discussion and Prospects

$$\hat{\pi}^* = \arg\max_{\pi} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim d_{\pi_D}} [m(\mathbf{s},\mathbf{a}) \log \pi(\mathbf{a}|\mathbf{s})]$$

- In traditional RL, $m(s,a)$ is the value function corresponds to $Q^{\pi}(s,a)$
  - Learned by TD propagation
  - Need the agent to interact with environment
- In big DM models, $m(s,a)$ is a masking (or weighting) function
  - Just predefined by rule
  - No need to interact with environment

Thank You!
Questions?

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Weinan Zhang

Associate Professor

APEX Data & Knowledge Management Lab

Department of Computer Science and Engineering

Shanghai Jiao Tong University

http://wnzhang.net