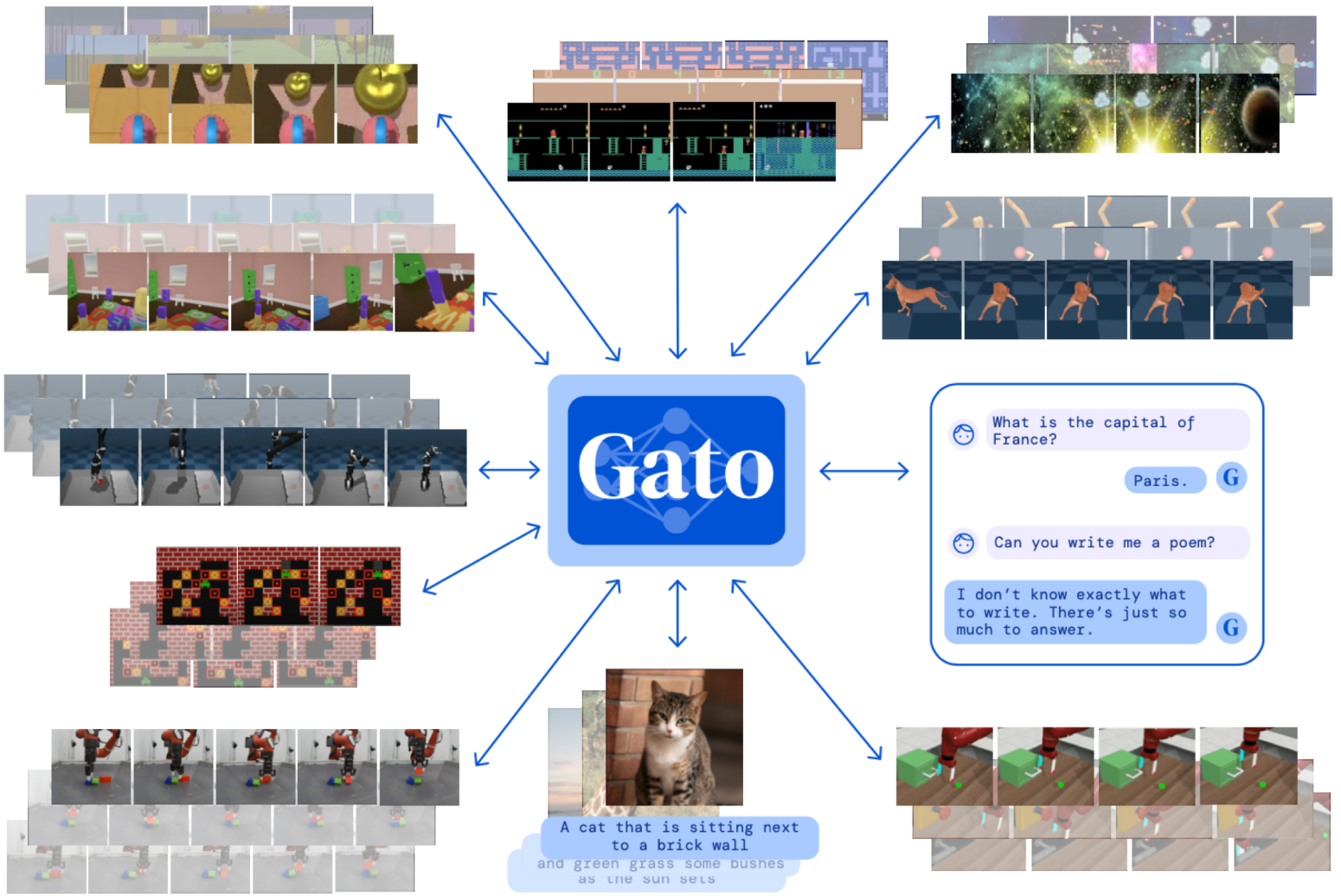
GATO: Towards General AI

John Tan Chong Min

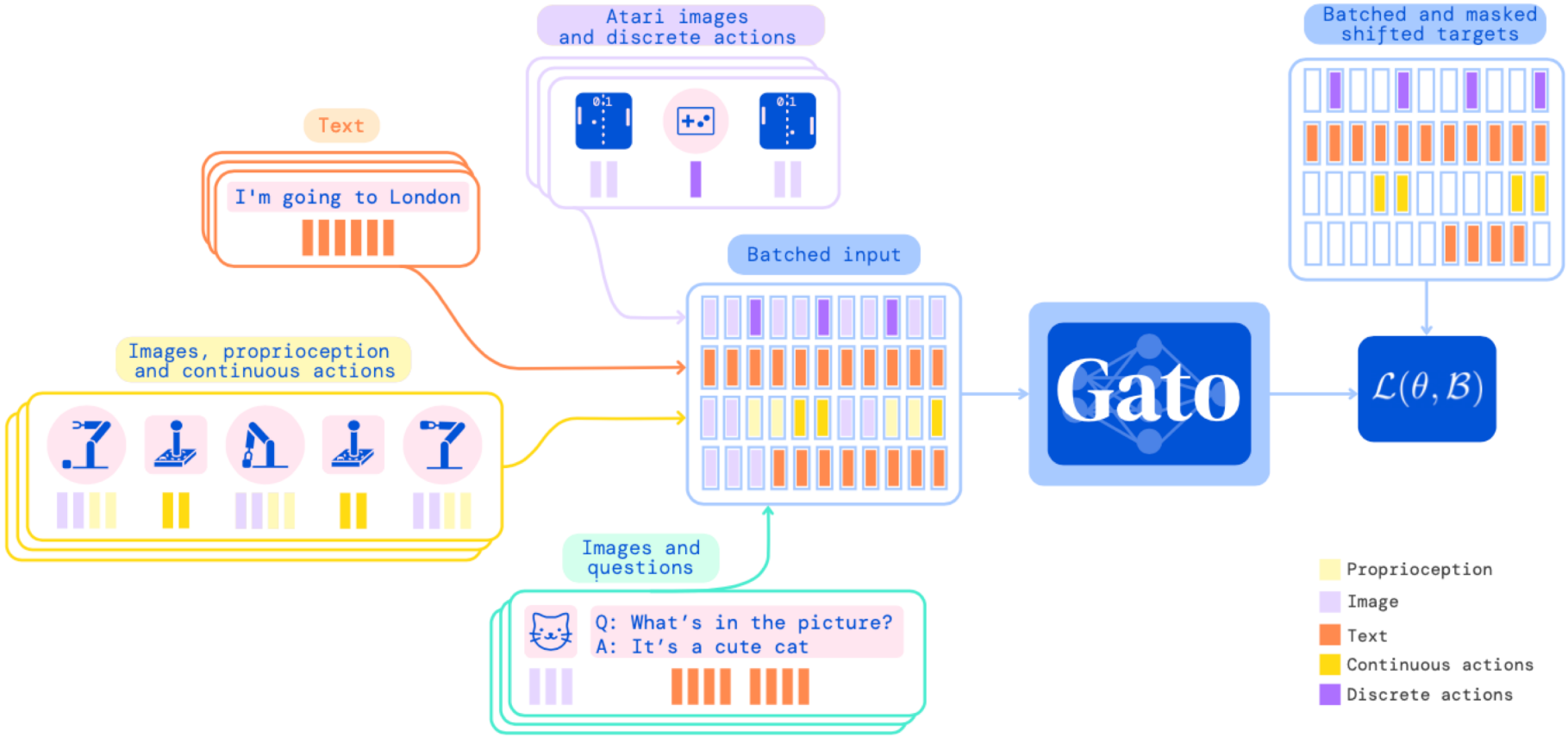
8 Aug 2022

Background



- Multi-modal, multi-task, multi-embodiment
- 1.2B parameters
- Same set of weights for all tasks

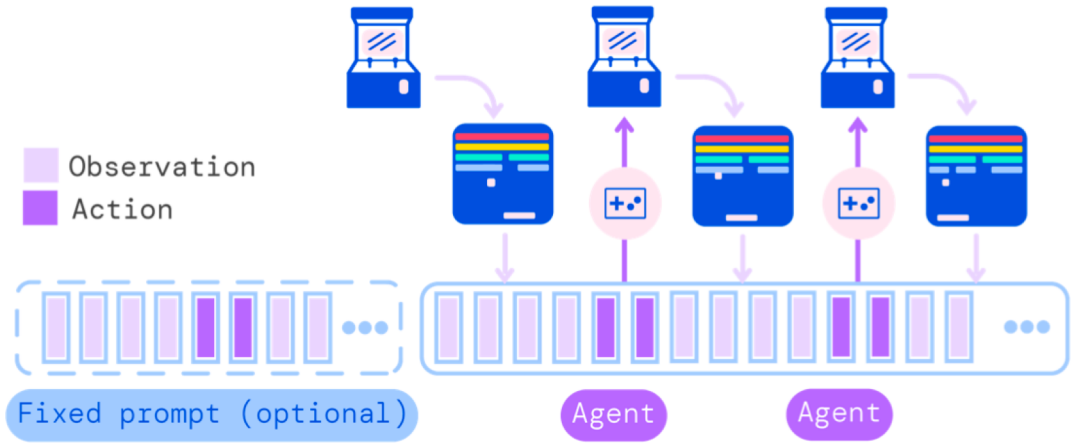
GATO Structure



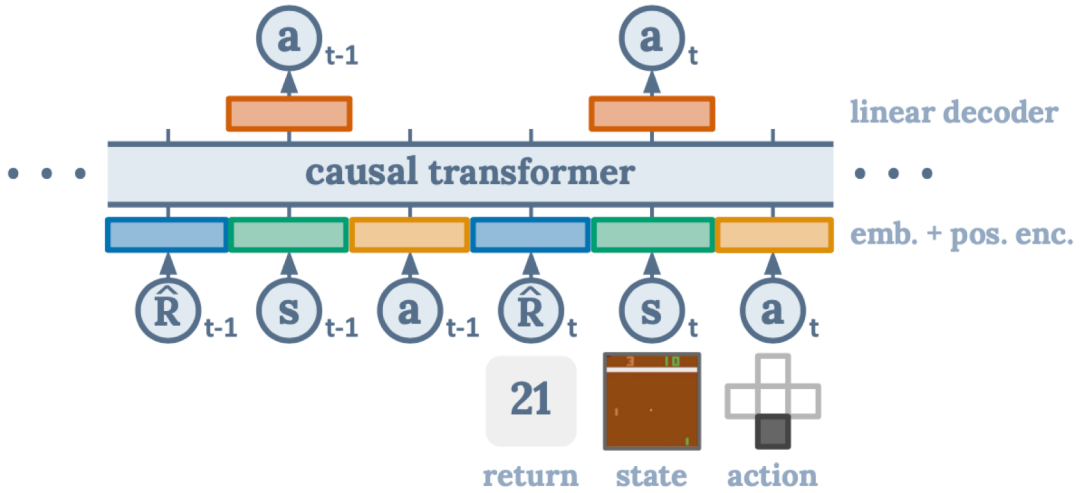
- Different input formats
- Masking to apply loss function only to target inputs

Prediction problem

- GATO does not predict observations, only the token of next action
  - Similar to decision transformers (only no rewards in sequence)
- Instead of one-hot task identifiers, prompt conditioning is used
  - Similar to T5 architecture: <prompt> + <sequence>
  - *Prompt: samples of episode, with 50% from end, and 50% uniformly sampled*



GATO



Decision Transformer

Prediction Problem (details)

Given a sequence of tokens $s_{1:L}$ and parameters θ, we model the data using the chain rule of probability:

$$\log p_\theta(s_1, \ldots, s_L) = \sum_{l=1}^{L} \log p_\theta(s_l | s_1, \ldots, s_{l-1}), \tag{1}$$

Let b index a training batch of sequences $\mathcal{B}$. We define a masking function m such that $m(b, l) = 1$ if the token at index l is either from text or from the logged action of an agent, and 0 otherwise. The training loss for a batch $\mathcal{B}$ can then be written as

$$\mathcal{L}(\theta, \mathcal{B}) = -\sum_{b=1}^{|\mathcal{B}|} \sum_{l=1}^{L} m\,(b, l) \log p_\theta \left(s_l^{(b)} | s_1^{(b)}, \ldots, s_{l-1}^{(b)} \right) \tag{2}$$

Tokenization

Data Type	Method	Ordering	Range
Text	SentencePiece with 32000 subwords	Text order	[0, 32000)
Images	Split into non-overlapping 16x16 patches and use ViT	Raster order	[-1, 1] for each pixel, divided by square root of patch size (i.e. 4)
Discrete values (e.g. Atari actions)	Flattened into sequences of integers	Row-major order	[0, 1024)
Continuous values (e.g. proprioceptive inputs)	Flattened into sequences of floating point values	Row-major order	Mu-law encoded to [-1,1], discretized to 1024 uniform bins. Then shifted to [32000,33024)

Tokenization (details)

- **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:

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

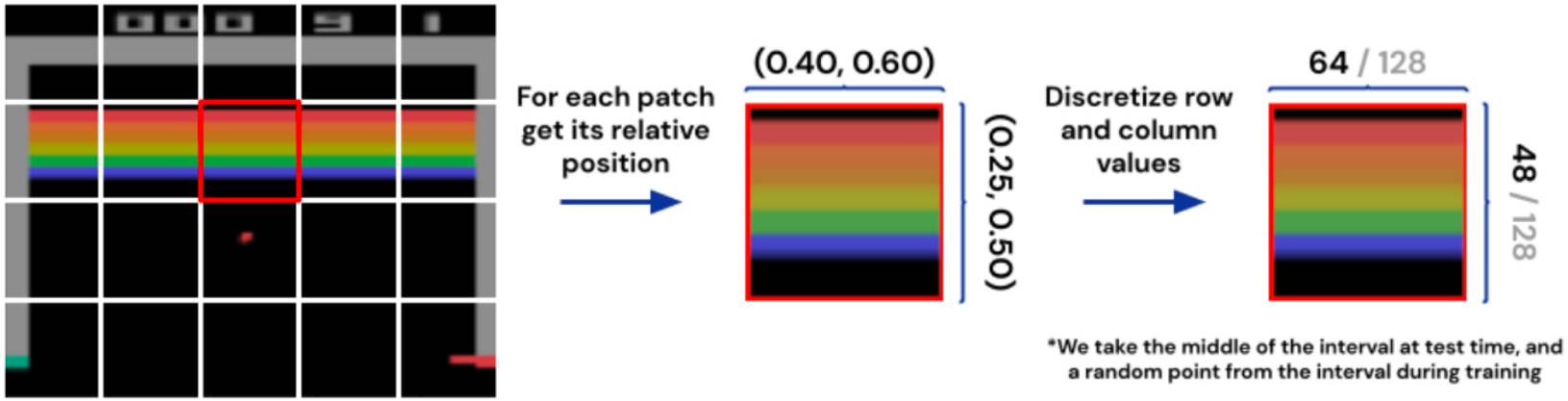
where $L = T(k + m + n + 1 + A)$ is the total number of tokens.

Embedding Inputs (details)

- Parameterized embedding function $f(\cdot; \theta e)$ to each token
  - Text, discrete or continuous valued observations or actions are embedded via lookup table into learned vector embedding space
  - Image patches embedded using a ResNet to obtain a vector per patch
- Learnable position encoding vector added to vector

Image Embedding

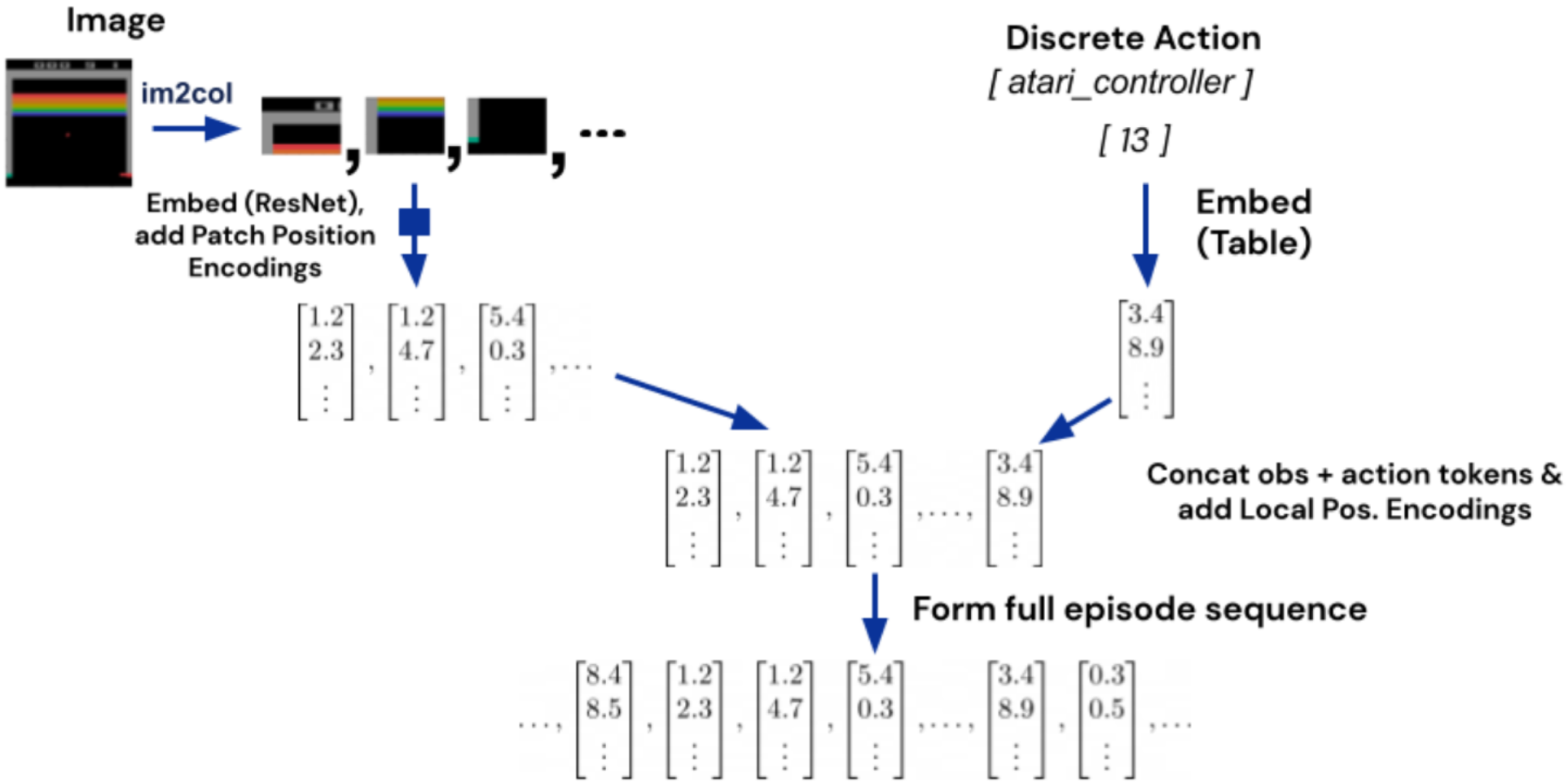
- Similar to ViT



Final patch embedding | ResNet-based embedding | Column position embedding | Row position embedding

$$\begin{bmatrix} 4.6 \\ 3.4 \\ \vdots \end{bmatrix} = \begin{bmatrix} 7.6 \\ 5.3 \\ \vdots \end{bmatrix} + col_{64} + row_{48}$$

Tokenization + Embedding Pipeline (Image + Discrete actions)



Tokenization + Embedding Pipeline (Proprioception + Continuous actions)

Proprioception

[cart_x, velocity_x, pole_x, pole_y]

[0.3, 1.5, 0.25, -0.4]

Mu-law Encode, Discretize & Embed (Table)

$\begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix}, \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix}, \begin{bmatrix}7.6\\5.3\\\vdots\end{bmatrix}, \begin{bmatrix}4.6\\3.4\\\vdots\end{bmatrix}$

Token Embedding

Continuous Action

[cart_move_x]

[-0.7]

Discretize & Embed (Table)

$\begin{bmatrix}9.0\\3.3\\\vdots\end{bmatrix}$

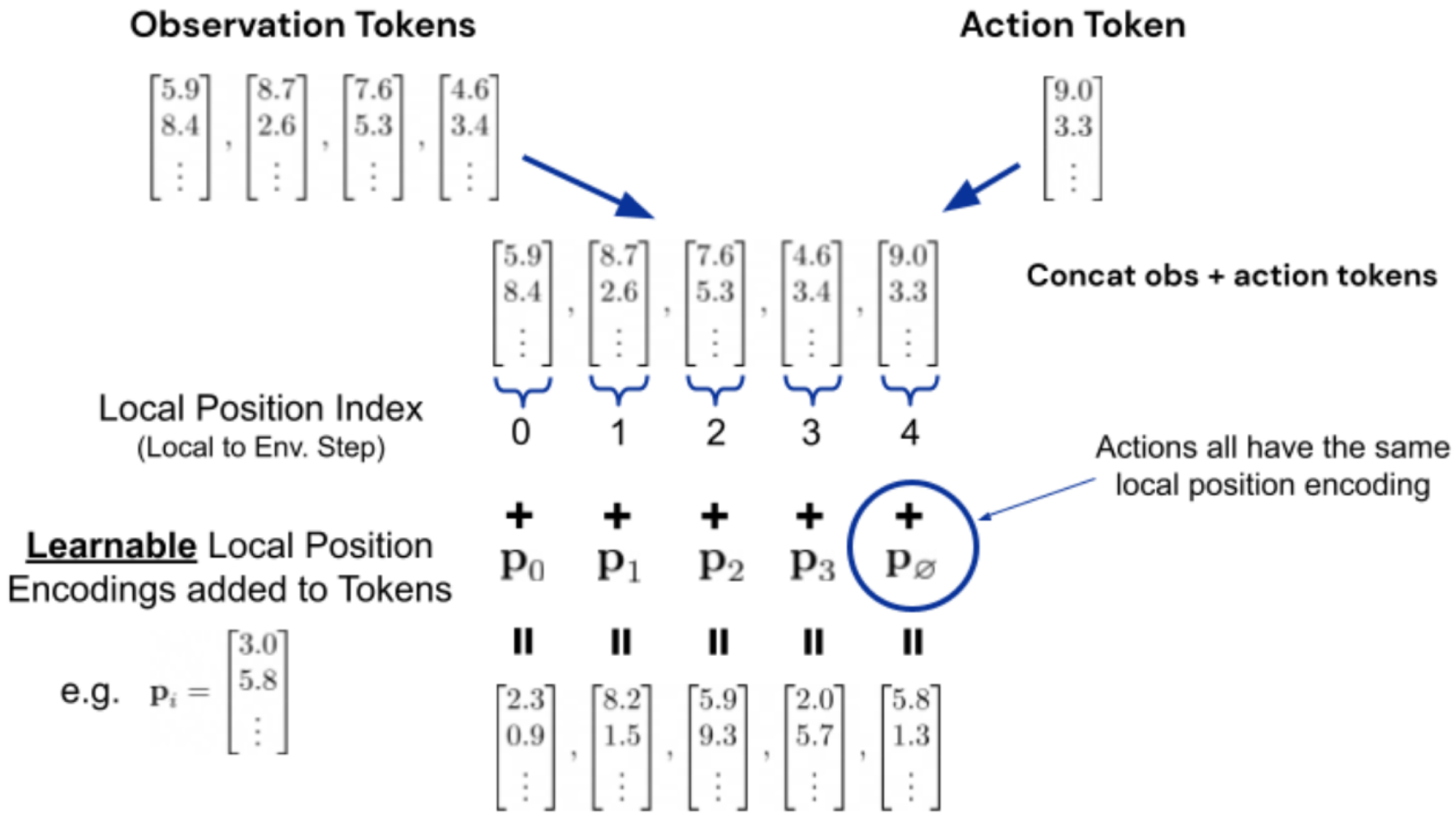
$\begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix}, \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix}, \begin{bmatrix}7.6\\5.3\\\vdots\end{bmatrix}, \begin{bmatrix}4.6\\3.4\\\vdots\end{bmatrix}, \begin{bmatrix}9.0\\3.3\\\vdots\end{bmatrix}$

Concat obs + action tokens & add Local Pos. Encodings

Form full episode sequence

$\ldots, \begin{bmatrix}2.3\\8.3\\\vdots\end{bmatrix}, \begin{bmatrix}5.9\\8.4\\\vdots\end{bmatrix}, \begin{bmatrix}8.7\\2.6\\\vdots\end{bmatrix}, \begin{bmatrix}7.6\\5.3\\\vdots\end{bmatrix}, \begin{bmatrix}4.6\\3.4\\\vdots\end{bmatrix}, \begin{bmatrix}9.0\\3.3\\\vdots\end{bmatrix}, \begin{bmatrix}2.3\\0.9\\\vdots\end{bmatrix}, \ldots$

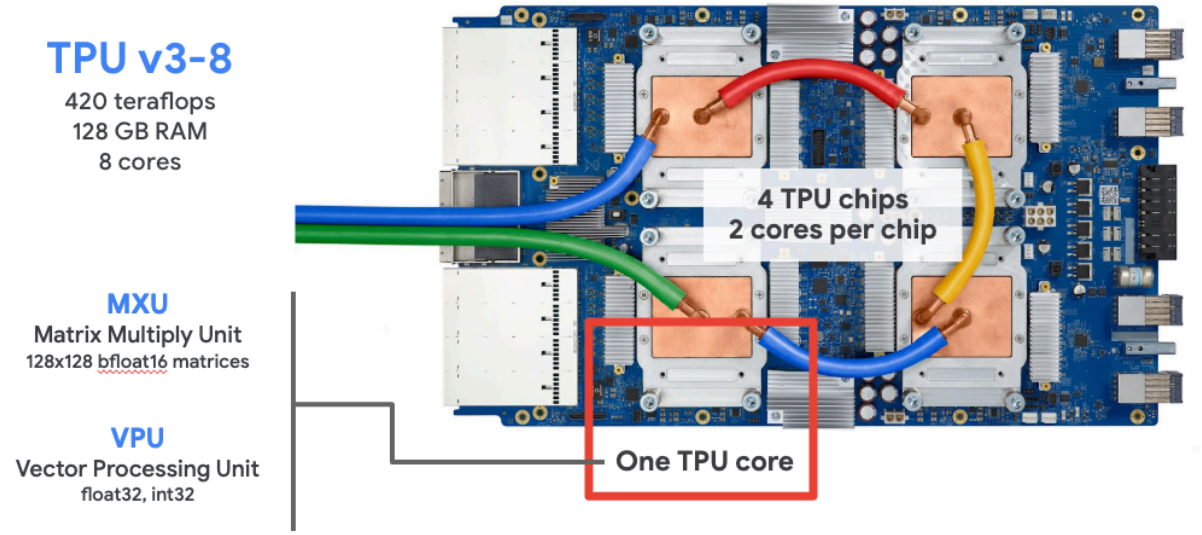
Local Position Embeddings



Training Details

- Hardware: 16x16 TPU v3 slice
- Timesteps: 1M
- Batch size: 512
- Token sequence length: 1024

- Training time: 4 days



TPU type (v3)	v3 cores	chips	Total memory	Evaluation price (USD)
v3-32	32	16	512 GiB	$32 / hour

Datasets

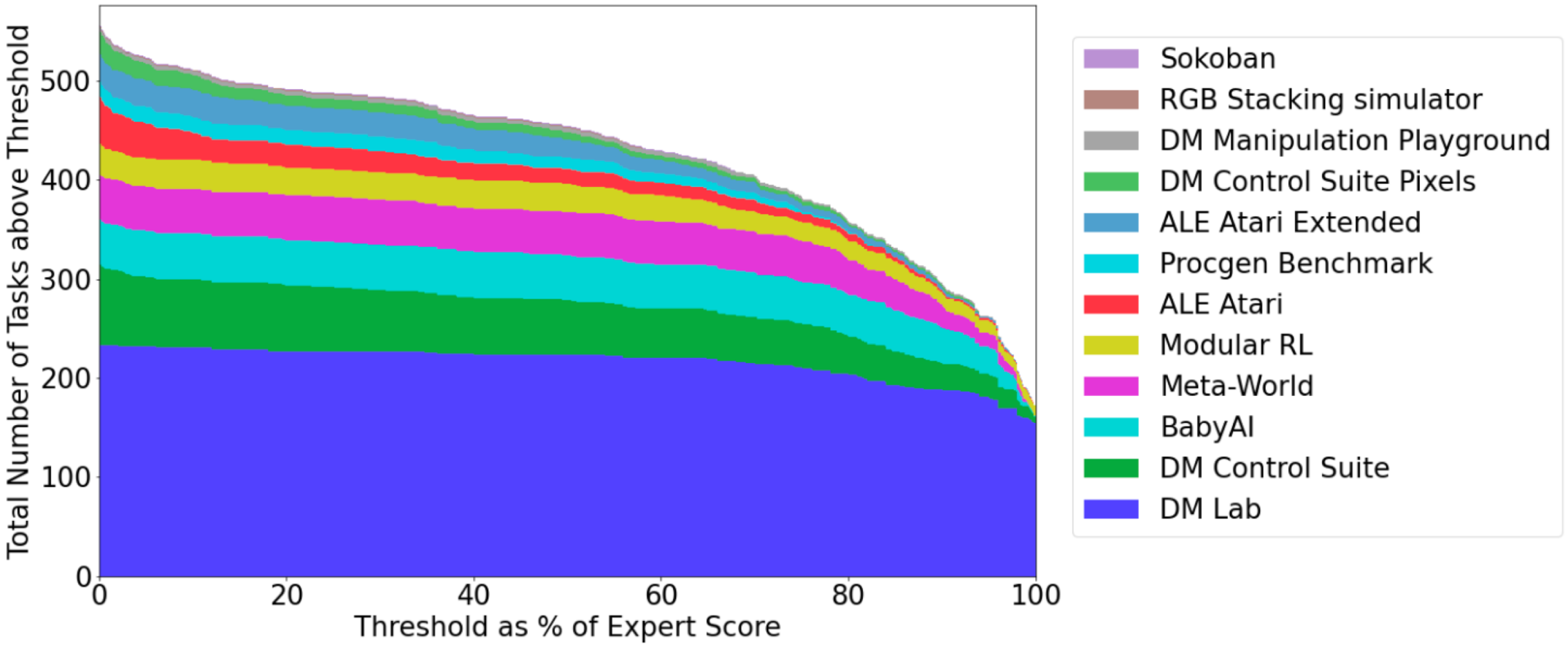
Control environment	Tasks	Episodes	Approx. Tokens	Sample Weight
DM Lab	254	16.4M	194B	9.35%
ALE Atari	51	63.4K	1.26B	9.5%
ALE Atari Extended	28	28.4K	565M	10.0%
Sokoban	1	27.2K	298M	1.33%
BabyAI	46	4.61M	22.8B	9.06%
DM Control Suite	30	395K	22.5B	4.62%
DM Control Suite Pixels	28	485K	35.5B	7.07%
DM Control 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%

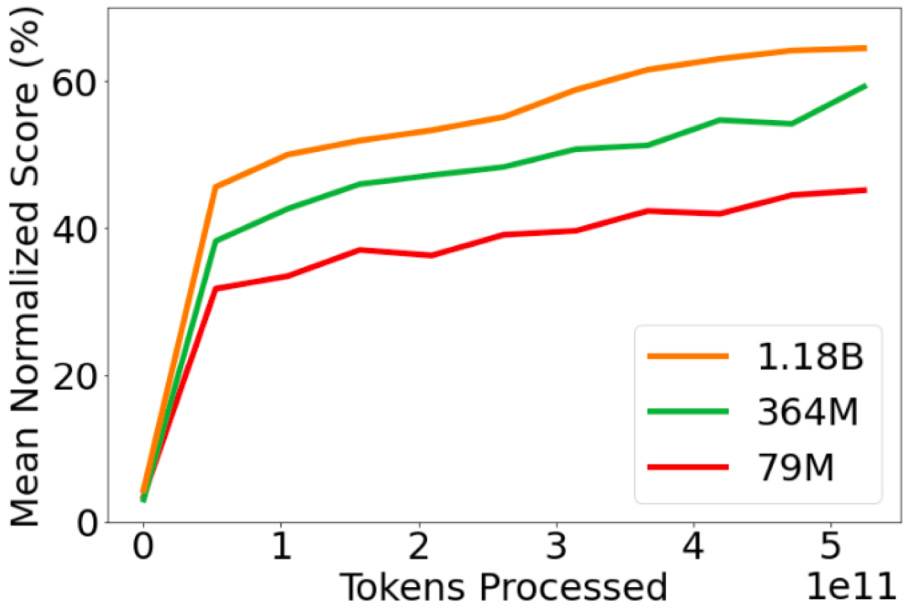
Training Procedure

- Mimic expert trajectories from SOTA or near-SOTA agents
- Train only with episodes returns of at least 80% of expert return
- Future work: Supposedly possible to learn via RL from scratch

Is the general agent as good as the expert?

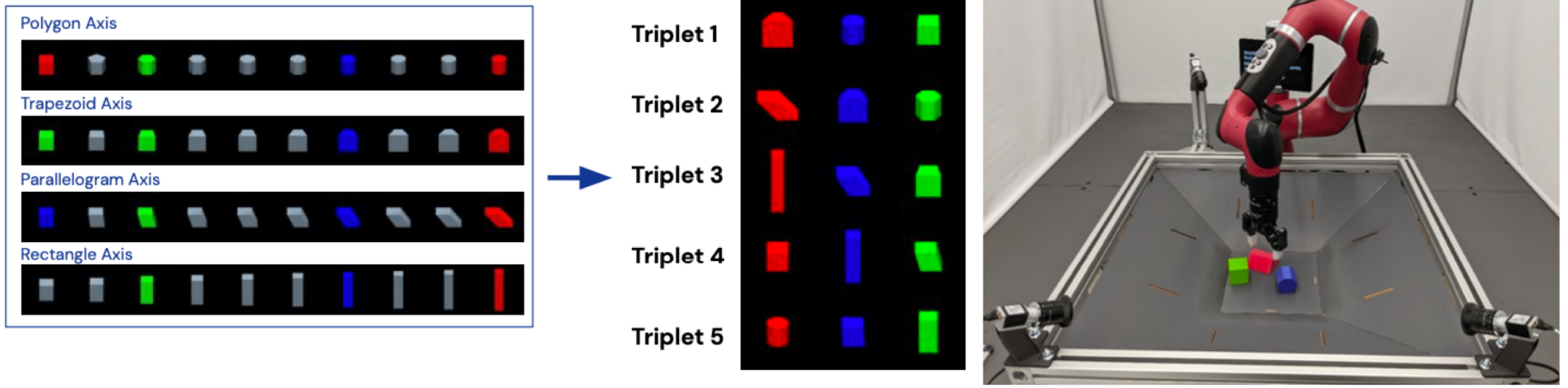


Is GATO scalable?



- Normalized return:
  - Each task calculate performance of model as percentage of expert score
  - Average percentage score across all tasks of a domain
  - Mean-aggregate percentage score across all domains
- Increasing tokens trained = increased performance
- Increasing model size = increased performance

Can GATO generalize (zero-shot)?



In-distribution

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%

Out-of-distribution

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%

- Not too well on held-out set, zero-shot transfer is a common problem with ML techniques in general

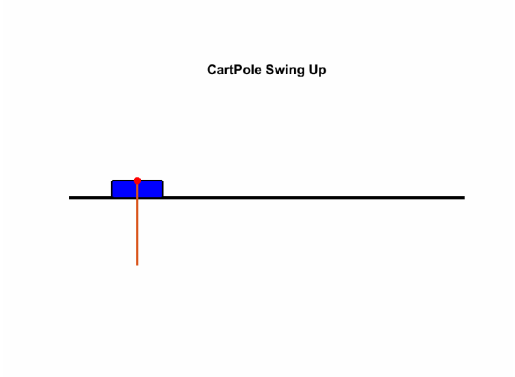
Generalizability experiments (few-shot)

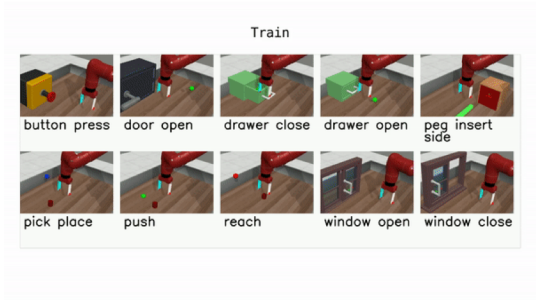
- Should ideally learn by conditioning on different prompts
  - Sequence lengths of tokenized demonstrations too long
  - Maximum content length insufficient to describe task

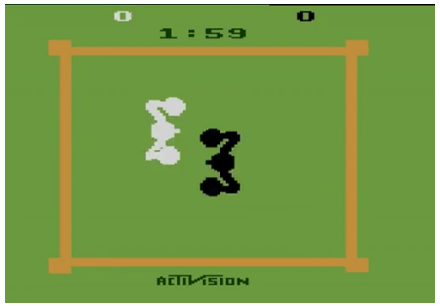
- Instead, fine-tune agent's parameters on new task, and evaluate fine-tuned model's performance on environment

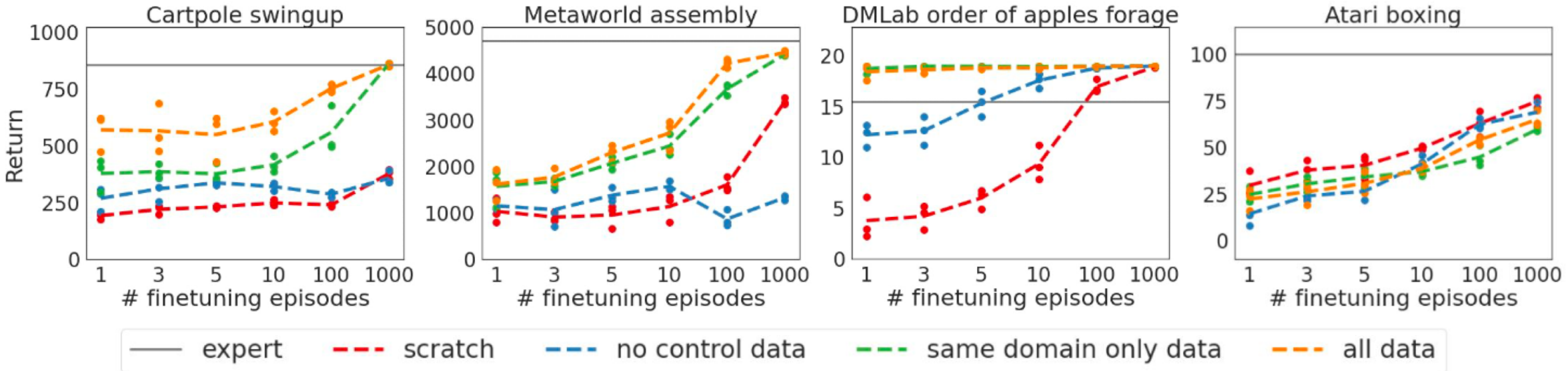
- 3 models
  - *same domain only data*: pretrained only from data from same domain as task to be fine-tuned on
  - *no control data*: pretrained only on non-control data
  - *scratch*: no pretraining at all

Results of generalizability experiments









Non-image data
(No benefit for control = no trf)

Image data
(Natural = +ve trf)
(Non-natural = no trf)

Discussion (Part 1)

- CNNs vs ViTs for image representation. Which is better?
- Learnable positional embeddings vs pre-defined positional embeddings (as in original Transformer). Which has better inductive bias and hence better for generalization?
- Difficulties in generalization without fine-tuning. How can we better encode the context in the prompt or network structure for better generalization?

Discussion (Part 2)

- GATO never seems to be able to beat the expert. Why?
- GATO is fundamentally model-free, like Decision Transformers. How can we incorporate a world model for planning?
- Can GATO learn from scratch like AlphaZero?

Discussion