A. Appendix

A.1. Network architectures and Training setup

A.1.1. Importance Weighted Actor-Learner Architecture

Agents were trained using the IMPALA framework (Espeholt et al., 2018). Briefly, there are N parallel 'actors' collecting experience from the environment in a replay buffer and one learner taking batches of trajectories and performing the learning updates. During one learning update the agent network is unrolled, all the losses (RL and auxiliary ones) are evaluated and the gradients computed.

A.1.2. Agents

Input encoder To process the frame input, all models in this work use a residual network (He et al., 2016) of 6 64-channel ResNet blocks with rectified linear activation functions and bottleneck channel of size 32. We use strides of (2, 1, 2, 1, 2, 1) and don't use batch-norm. Following the convnet we flatten the ouput and use a linear layer to reduce the size to 500 dimensions. Finally, we concatenate this encoding of the frame together with a one hot encoding of the previous action and the previous reward.

Core architecture The recurrent core of all agents is a 2-layer LSTM with 256 hidden units per layer. At each time step this core consumes the input embedding described above and updates its state. We then use a 200 units single layer MLP to compute a value baseline and an equivalent network to compute action logits, from where one discrete action is sampled.

Simulation Network Both predictive agents have a simulation network with the same architecture as the agent's core. This network is initialized with the agent state at some random time t from the trajectory and unrolled forward for a random number of steps up to 16, receiving only the actions of the agent as inputs. We then use the resulting LSTM hidden state as conditional input for the prediction loss (SimCore or CPC|A).

SimCore We use the same architecture and hyperparameters described in (Gregor et al., 2019). The output of the simulation network is used to condition a Convolutional DRAW (Gregor et al., 2016). This is a conditional deep variational auto-encoder with recurrent encoder and decoder using convolutional operations and a canvas that accumulates the results at each step to compute the distribution over inputs. It features a recurrent prior network that receives the conditioning vector and computes a prior over the latent variables. See more details in (Gregor et al., 2019).

Action-conditional CPC We replicate the architecture used in (Guo et al., 2018). CPC|A uses the output of the simulation network as input to an MLP that is trained to discriminate true versus false future frame embedding. Specifically, the simulation network outputs a conditioning vector after k simulation steps which is concatenated with the frame embedding z_{t+k} produced by the image encoder on the frame x_{t+k} and sent through the MLP discriminator. The discriminator has one hidden layer of 512 units, ReLU activations and a linear output of size 1 which is trained to binary classify true embeddings into one class and false embeddings into another. We take the negative examples from random time points in the same batch of trajectories.

A.1.3. QA network architecture

Question encoding The question string is first tokenized to words and then mapped to integers corresponding to vocabulary indices. These are then used to lookup 32-dimensional embeddings for each word. We then unroll a 64-units single-layer LSTM for a fixed number of 15 steps. The language representation is then computed by summing the hidden states for all time steps.

QA decoder. To decode answers from the internal state of the agents we use a second LSTM initialized with the internal state of the agent's LSTM and unroll it for a fixed number of steps, consuming the question embedding at each step. The results reported in the main section were computed using 12 decoding steps. The terminal state is sent through a two-layer MLP (sizes 256, 256) to compute a vector of answer logits with the size of the vocabulary and output the top-1 answer.

A.1.4. Hyper-parameters

The hyper-parameter values used in all the experiments are in Table 3.

A.1.5. Negative sampling strategies for CPC|A

We experimented with multiple sampling strategies for the CPC|A agent (whether or not negative examples are sampled from the same trajectory, the number of contrastive prediction steps, the number of negative examples). We report the best results in the main text. The CPC|A agent did provide better representations of the environment than the LSTM-based agent, as shown by the top-down view reconstruction loss (Figure 6a). However, none of the CPC|A agent variations that we tried led to better-than-chance question-answering accuracy. As an example, in Figure 6b we compare sampling negatives from the same trajectory or from any trajectory in the training batch.

A.2. Effect of QA network depth

To study the effect of the QA network capacity on the answer accuracy, we tested decoders of different depths applied to both the SimCore and the LSTM agent's internal represen-

Agent	
Learning rate	1e-4
Unroll length	50
Adam β_1	0.90
Adam β_2	0.95
Policy entropy regularization	0.0003
Discount factor	0.99
No. of ResNet blocks	6
No. of channel in ResNet block	64
Frame embedding size	500
No. of LSTM layers	2
No. of units per LSTM layer	256
No. of units in value MLP	200
No. of units in policy MLP	200
Simulation Network	
Overshoot length	16
No. of LSTM layers	2
No. of units per LSTM layer	256
No. of simulations per trajectory	6
No. of evaluations per overshoot	2
SimCore	
No. of ConvDRAW Steps	8
GECO kappa	0.0015
CPC|A	
MLP discriminator size	64
QA network	
Vocabulary size	1000
Maximum question length	15
No. of units in Text LSTM encoder	64
Question embedding size	32
No. of LSTM layers in question decoder	2
No. of units per LSTM layer	256
No. of units in question decoder MLP	200
No. of decoding steps	12

Table 3. Hyperparameters.

tations (7). The QA network is an LSTM initialized with the agent's internal state that we unroll for a fixed number of steps feeding the question as input at each step. We found that, indeed, the answering accuracy increased with the number of unroll steps from 1 to 12, while greater number of steps became detrimental. We performed the same analysis on the LSTM agent and found that regardless of the capacity of the QA network, we could not decode the correct answer from its internal state, suggesting that the limiting factor is not the capacity of the decoder but the lack of useful representations in the LSTM agent state.

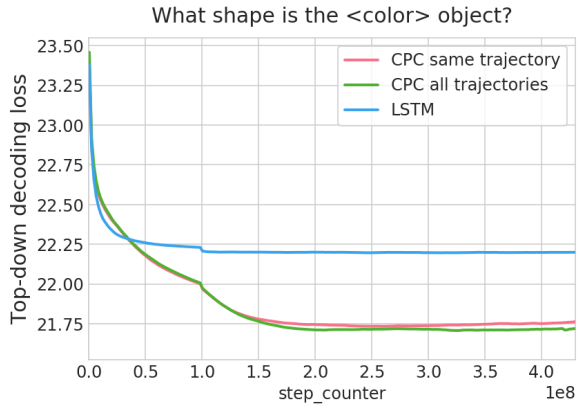
A.3. Answering accuracy during training for all questions

The QA accuracy over training for all questions is shown in Figure 8.

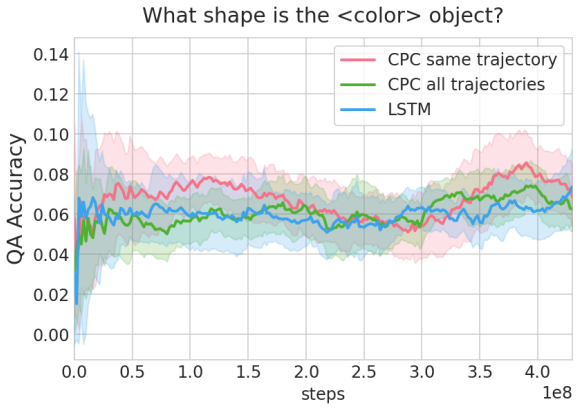
A.4. Environment

Our environment is a single L-shaped 3D room, procedurally populated with an assortment of objects.

Actions and Observations. The environment is episodic, and runs at 30 frames per second. Each episode takes 30 seconds (or 900 steps). At each step, the environment provides the agent with two observations: a 96x72 RGB image with the first-person view of the agent and the text containing the



(a) To test whether the CPC|A loss provided improved representations we reconstructed the environment top-down view, similar to (Gregor et al., 2019). Indeed the reconstruction loss is lower for CPC|A than for the LSTM agent.



(b) QA accuracy for the CPC|A agent is not better than the LSTM agent, for both sampling strategies of negatives.

Figure 6.

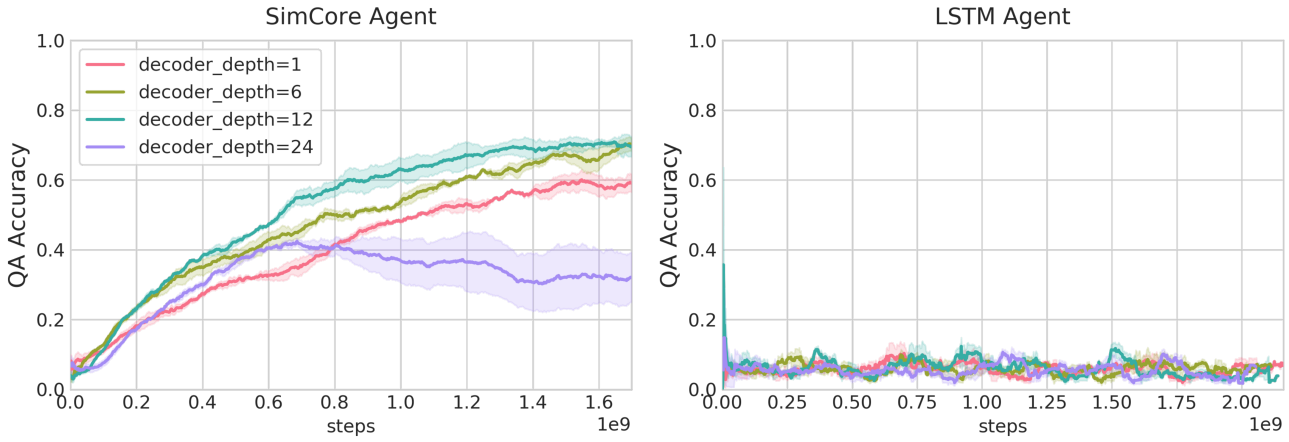


Figure 7. Answer accuracy over training for increasing QA decoder's depths. Left subplot shows the results for the SimCore agent and right subplot for the LSTM baseline. For SimCore, the QA accuracy increases with the decoder depth, up to 12 layers. For the LSTM agent, QA accuracy is not better than chance regardless of the capacity of the QA network.

question.

The agent can interact with the environment by providing multiple simultaneous actions to control movement (forward/back, left/right), looking (up/down, left/right), picking up and manipulating objects (4 degrees of freedom: yaw, pitch, roll + movement along the axis between agent and object).

Rewards. To allow training using cross-entropy, as described in Section 4, the environment provides the ground-truth answer instead of the reward to the agent.

Object creation and placement. We generate between 2 and 20 objects, depending on the task, with the type of the object, its color and size being uniformly sampled from the set described in Table 4.

Objects will be placed in a random location and random orientation. For some tasks, we required some additional constraints - for example, if the question is "What is the color of the cushion near the bed?", we need to ensure only one cushion is close to the bed. This was done by checking the constraints and regenerating the placement in case they were not satisfied.

Attribute	Options
Object	basketball, cushion, carriage, train, grinder, candle, teddy, chair, scissors, stool, book, football, rubber duck, glass, toothpaste, arm chair, robot, hairdryer, cube block, bathtub, TV, plane, cuboid block, car, tv cabinet, plate, soap, rocket, dining table, pillar block, potted plant, boat, tennisball, tape dispenser, pencil, wash basin, vase, picture frame, bottle, bed, helicopter, napkin, table lamp, wardrobe, racket, keyboard, chest, bus, roof block, toilet
Color	aquamarine, blue, green, magenta, orange, purple, pink, red, white, yellow
Size	small, medium, large

Table 4. Randomization of objects in the Unity room. 50 different types, 10 different colors and 3 different scales.

Body movement actions	**Movement and grip actions**	**Object manipulation**
NOOP	GRAB	GRAB + SPIN_OBJECT_RIGHT
MOVE_FORWARD	GRAB + MOVE_FORWARD	GRAB + SPIN_OBJECT_LEFT
MOVE_BACKWARD	GRAB + MOVE_BACKWARD	GRAB + SPIN_OBJECT_UP
MOVE_RIGHT	GRAB + MOVE_RIGHT	GRAB + SPIN_OBJECT_DOWN
MOVE_LEFT	GRAB + MOVE_BACKWARD	GRAB + SPIN_OBJECT_FORWARD
LOOK_RIGHT	GRAB + LOOK_RIGHT	GRAB + SPIN_OBJECT_BACKWARD
LOOK_LEFT	GRAB + LOOK_LEFT	GRAB + PUSH_OBJECT_AWAY
LOOK_UP	GRAB + LOOK_UP	GRAB + PULL_OBJECT_CLOSE
LOOK_DOWN	GRAB + LOOK_DOWN	

Table 5. Environment action set.

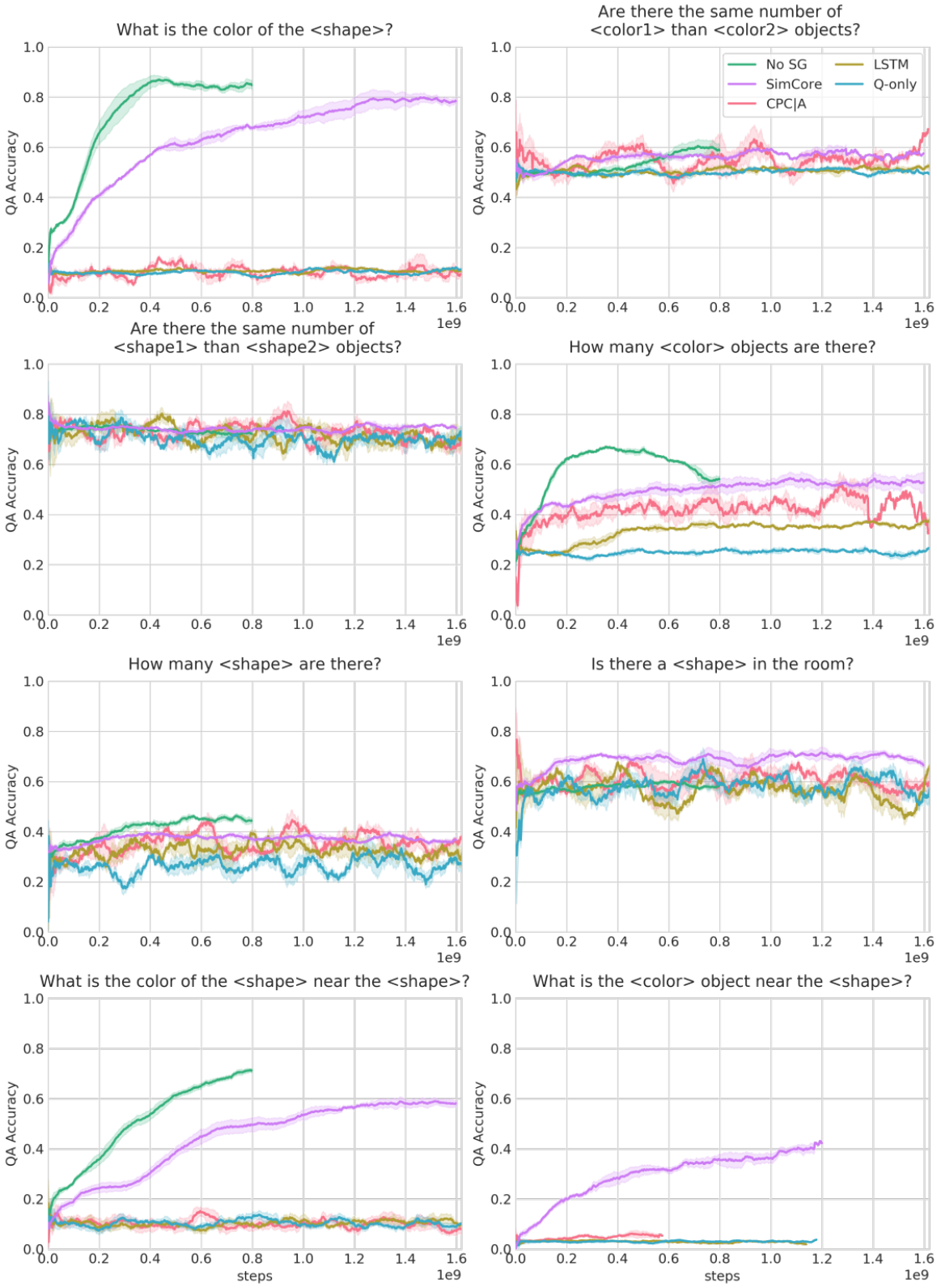


Figure 8. QA accuracy over training for all questions and all models.