A HYPERPARAMTERS

All neural networks were build using the Jax library (Bradbury et al., 2018). In all experiments, training was carried out using a distributed A3C setup (Espeholt et al., 2018) with discrete actions. For 3D Unity Env experiments, we added an additional Pixel Control loss (Jaderberg et al., 2016) for all agents. We used a single learner and 256 actors. Important training hyper-parameters are shown in Table 2, along with the components of the agent's architecture that are shared between the different models. The parameter values used for each model presented in the main paper are shown below in Table 3.

A.1 HYPERPARAMTER SEARCH

We tuned all models using the BabyAI "Place X next to Y" task (Chevalier-Boisvert et al., 2019) (§B.1). For each architectures, we tuned using a random search. Additionally, we increased the size of the LSTM so that all architectures had approximately the same number of parameters.

Composable Perceptual Schemas. We found that we did not need to tune the model much beyond a random search over attention projection dims $W_i \in [16, 32]$ and Conv LSTM kernel size $[3, 5]$.

Attention Augmented agent. We used hyper-parameters from their paper put tuned the following: LSTM hidden size $[256, 512]$, Attention query MLP size $[\{\}, \{256\}, \{256, 256\}]$, number of attention heads $[4, 8]$. We consulted the authors about our implementation.

Recurrent Independent Mechanisms. We used hyper-parameters from their paper put tuned the following: LSTM hidden size $[100, 128, 256]$, Observation/communication head size $[32, 64, 128]$, number of observation/communication heads $[4, 5, 6]$, number of RIMs $[4, 6, 9, 12]$. We consulted the authors about our implementation and used their source code for replication `https://github.com/anirudh9119/RIMs`. We note that we did not use their "top-k" feature where only k RIMs are active because that **doubled** run-time. Since our experiments typically required ≥ 1 billion frames, using "top-k" led individual runs to take > 2 days.

Long Short-Term Memory. We searched an LSTM hidden size $[128, 256, 512]$. We consistently found that a larger memory had better results.

Table 2: Training hyper-parameters and shared network components used in experiments.

Loss Hyper-parameters	**3D Unity Env**	**Gridworlds**
V-trace baseline cost	1.0	0.5
V-trace entropy cost	10^{-4}	0.01
V-trace γ	0.95	1.0
V-trace loss scaling	0.1	1.0
Pixel Control loss scaling	0.1	–
Pixel Control loss cell size	4	–
Pixel Control discount factor	0.9	–
Pixel Control de-convolution sizes	(6, 9, 32), (8, 11, 32)	–
Pixel Control kernel shape	(3, 4)	–
Pixel Control de-convolution output shape	(18, 42)	–
Optimizer	clipped Adam	clipped Adam
Learning rate	2×10^{-4}	10^{-4}
Max gradient Norm	40.0	40.0
Optimizer epsilon	5×10^{-8}	10^{-8}
Adam β_1	0.0	0.9
Adam β_2	0.95	0.999
Shared Network Components		
Language encoder	GRU	GUR
Language encoder hidden sizes	128	128
Language word embedding size	128	128
Image encoder	Res-Net	Res-Net
Res-Net channels	(16, 32, 32)	(16, 32, 32)
Res-Net residual blocks	(2, 2, 2)	(2, 2, 2)
Res-Net stride	2	2
Res-Net kernel size	3	3
Res-Net padding	SAME	SAME
Previous action encoding	Identity	Identity
Reward encoding	Identity	Identity
Image-language-reward-action combination	Concatenation	Concatenation
Input Convolutional dims	(9, 12, 32)	(7, 7, 32)
Policy Head MLP shapes	[512, 46]	[200, 7]
Value Head MLP shapes	[512, 1]	[200, 1]

Table 3: Model specific parameters.

Model parameter	**3D Unity Env**	**Gridworld Ballet**	**Gridworld Keybox**
Observation Dims	72×96	99×99	56×56
Composable Perceptual Schemas			
Parameters (millions)	5.1	7.1	7.6
Input Convolutional dims	(9, 12, 32)	(12, 12, 32)	(7, 7, 32)
Policy-state size	512	512	1024
Number of subschemas	4	4	8
Schema dimension	128	128	128
Relation heads	2	2	4
Projection dims W_1, W_2	16	16	16
ConvLSTM kernel size	3	3	3
ConvLSTM hidden size	32	32	32
LSTM			
Parameters (millions)	5.6	7.2	7.6
LSTM Hidden size	896	768	1024
Attention Augmented Agent			
Parameters (millions)	5.1	6.9	7.5
ConvLSTM kernel size	3	3	3
ConvLSTM output size	128	128	128
LSTM hidden size	704	512	960
Number of attention heads	4	4	4
Attention query MLP size	(256, 256)	(256, 256)	(256, 256)
Positional basis dim	4	4	4
RIMs			
Parameters (millions)	5	6.6	7.6
Number of RIMs	12	9	9
Individual RIM size	128	128	128
Observation heads	6	6	6
Communication heads	6	6	6
Observation head size	32	32	32
Communication head size	32	32	32
Basis size (learned)	4	4	4
Dropout	0.2	0.2	0.2

B ADDITIONAL EXPERIMENTS

B.1 GENERALIZING TO AN UNSEEN NUMBER OF DISTRACTORS

We study this with the "Place X next to Y" task in the BabyAI gridworld (Chevalier-Boisvert et al., 2019) (Figure 11a). The agent is a red triangle. Other objects can be squares, boxes or circles and they can take on 7 colors. The agent receives a partial, egocentric observation of the environment (Figure 11a, right) and is given a synthetic language instruction. The agent gets a reward of 1 if chooses the correct dancer, and 0 otherwise. During training the agent sees either 0 or 2 distractors. During testing, the agent sees 11 distractors. As the number of distractors increases, the likelihood a distractor is either (a) confounding with the task objects or (b) blocks/confuses the agent also increases.

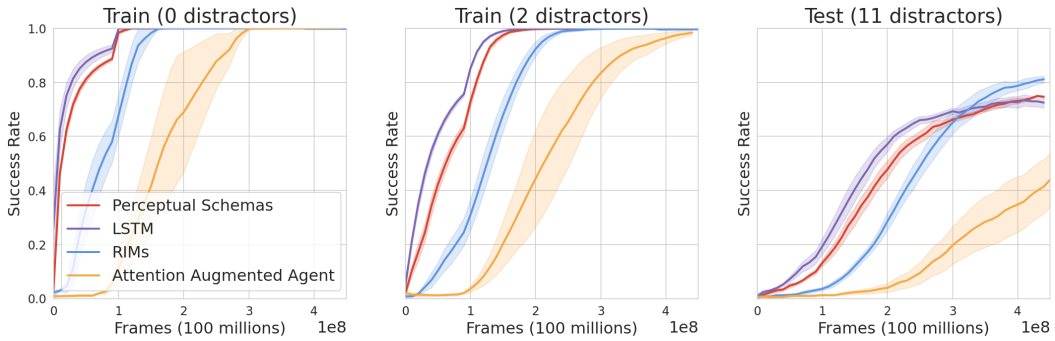


Figure 7: **RIMs, which uses spatial attention, better generalizes to more distractors.** We show train and test success rate performance for "Place X next to Y" in the BabyAI environment (10 runs).

We present results in Figure 7. On the left two panels, we present training results for $\{0, 2\}$ distractors. All architectures can learn this task. On the right-most panel, we present test results for 11 distractors. CPSand an LSTM get comparable performance ($\approx 70\%$). RIMs has the best generalization success rate ($\approx 80\%$).

B.2 ANALYSIS OF REPRESENTATIONS LEARNED FOR KEYBOX-LIKE ENVIRONMENT

We found no way to programmatically categorize the agent's experience with object-configurations. Thus, we found no way to study the representations learned for subsections in the KeyBox environment. In order to study this question, we created a toy "Abstract MDP" environment (Figure 11b). Importantly, each episode consists of performing a task in 3×3 environment. This is similar to the KeyBox task since its a sequence of 3×3 subsections. In this task, there are a fixed number of abstract MDPs which have their own unique object-placement, which mimics the fixed number of object-configurations that can be sampled for the KeyBox task. For an example of these differences, see Figure 11b). The object-placements of an abstract MDP are identical but the actual objects in the positions are completely random (i.e. both shape and color are random). We sample 1000 episodes from 20 abstract MDPs. By training CPS on this environment, we can see how it uses different subschemas to represent different categories. In this experiment, CPS had 4 RNNs of dimensionality 20, requiring that they all be used. In order to see how CPSrepresents these MDPS, we study the time-series of sum of each subschema LSTM-state: $\sum_{d_h} h_t^{(i)}$. **We present results in Figure 8**. We find that Subschemas respond to object-configurations, not to object types (e.g. key, ball, or box), nor to colors (green or blue).

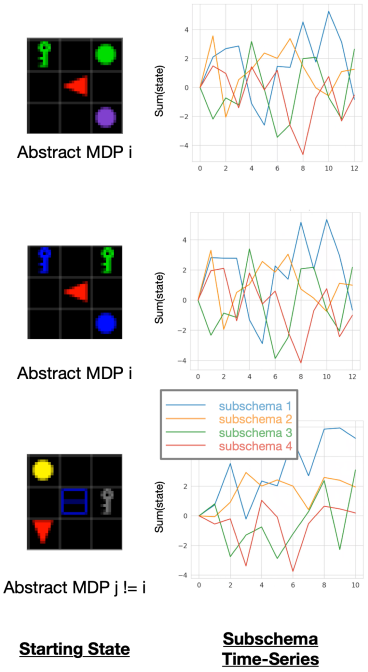


Figure 8: **Subschema respond to object-configuration not to individual object colors or shapes.** On the left, we plot the starting state of the episode. On the right, we plot the sum of each subschema LSTM-state.

C FULL CORRELATIONS FOR ANALYSIS

In this section, we show full plots for the analysis in §4.3.1. We show

1. Average L2 norm of all subschema-states (Figure 9).
2. Average pair-wise correlation between L2 norm of all subschema-states (Figure 10).

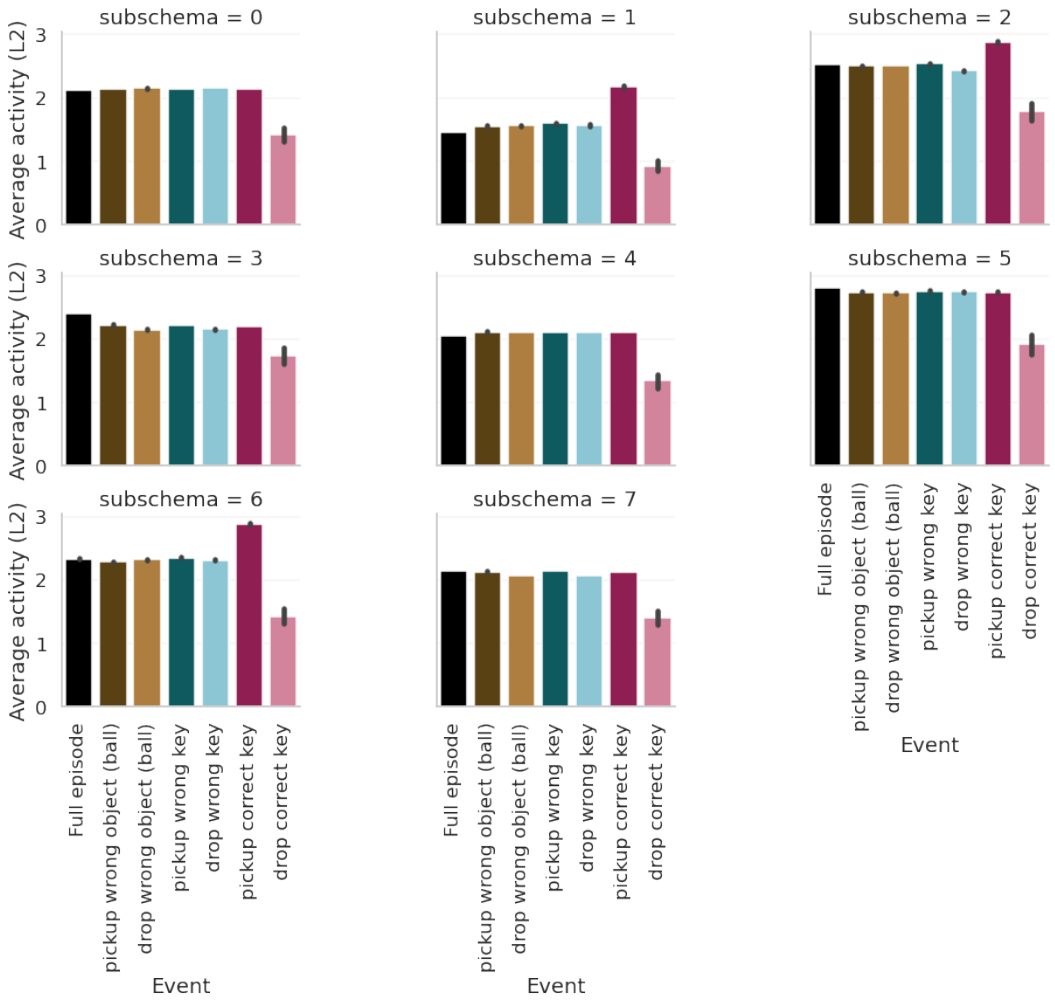
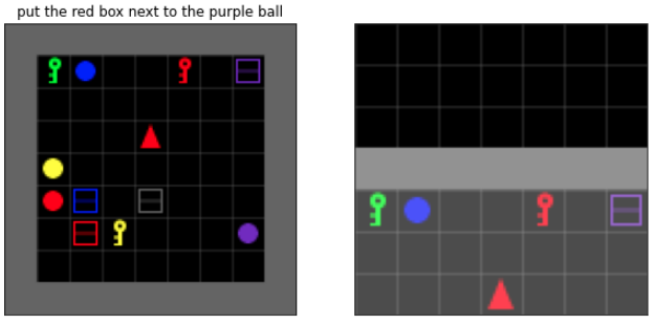


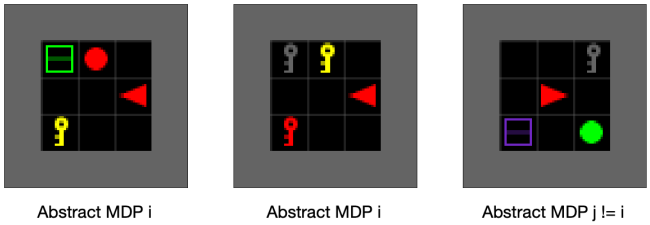
Figure 9: Average L2 norm of subschema-states.

Figure 10: Average pair-wise correlation between L2 norm of subschema-states.

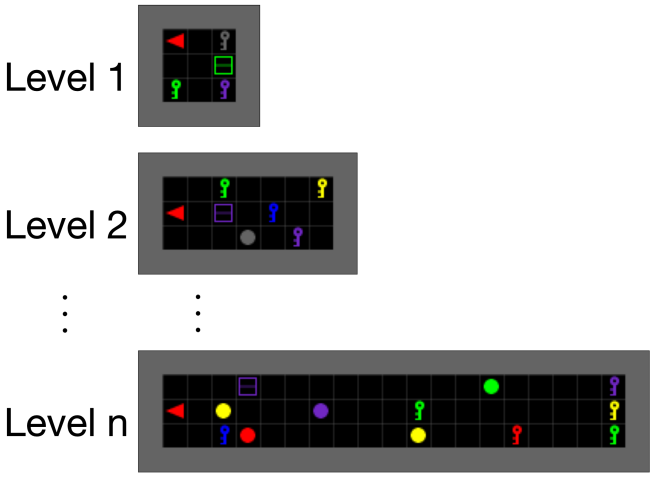
D ENVIRONMENTS



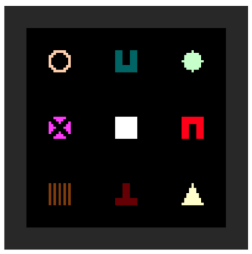
(a) Place X on Y task in BabyAI environment.



(b) Abstract MDP Environment based on BabyAI.



(c) KeyBox task.



(d) Ballet task.

Figure 11: Additional Environment Images.

D.1 BALLET

Please refer to Lampinen et al. (2021) for details on this task. Our only difference was to use tasks with $\{2, 4\}$ dancers during training and tasks with 8 dancers for testing.

D.2 KEYBOX

Observation Space. The agent receives a 56×56 partially observable, egocentric image of the environment as in Figure 11a, right.

Action Space. The action space is composed of the 7 discrete actions *turn left*, *turn right*, *go forward*, *pickup object*, *drop object*, *toggle*, and *done/no-op*.

Reward function. When the agent completes level n, it gets a reward of $n/n_{\texttt{max}}$ where $n_{\texttt{max}}$ is the maximum level the agent can complete. We set $n_{\texttt{max}} = 10$ during training. The agent has $50n$ time-steps to complete a level.

Table 4: Object and colors available for objects in the KeyBox task.

Set	Contains
Shapes	ball, key, box
Colors	red, green, blue, purple, pink, yellow, white

D.3 3D UNITY ENVIRONMENT

For the "place X on Y" experiments in 3D, all pickupable objects were split into two sets $O_1 = A \cup B$ and all object to place something on into another two sets $O_2 = C \cup D$, as shown in Table 5. Given the challenging nature of the 3D environment (huge number of possible states, partial observability, language commands, long credit assignment), we had to employ a set of curriculum tasks in order for the agents to make any progress on the actual task of interest "Put X on Y". The agent co-trained on the full set of tasks. This was possible since we used a distributed A3C setup for our training (Espeholt et al., 2018), where each of the actors generating the experience was running on one of the possible training levels. The different training tasks used during training and evaluation are shown in Table 6.

All episodes lasted for a maximum of 120 seconds and an action repeat of 4 was used. The images observations were rendered at $96 \times 72 \times 3$ and given to the agent along with a text language instruction, where each word in the instruction was mapped into a continuous vector of size 128 using a fixed vocabulary of maximum size 1000.

Reward function. An agent get's a reward of 1 if it completed the task and 0 otherwise.

Action Space. The action space for the experiments in 3d Unity Environment was 46 discrete actions that allow the agent to move its body and change its head direction, to grab objects while moving and manipulate the held objects by rotating, pulling or pushing the held object. The object is while as long as the agent is emitting a GRAB action, and dropped in the first instance that a GRAB action is not emitted. The full list of possible actions in the 3d Unity Environment environment is presented in Table 7.

Table 5: Object and color set splits for the 3d Unity Environment "Put X on Y" experiments.

Set	Contains
Set A (pickupable objects)	toilet roll, toothbrush, toothpaste
Set B (pickupable objects)	bus, car, carriage, helicopter, keyboard
Set C (support object)	stool, tv cabinet, wardrobe, wash basin
Set D (support object)	bed, book case, chest, dining, table
Colors	red, green, blue, aquamarine, magenta, orange, purple, pink, yellow, white

Table 6: Descriptions of all the tasks used during training and evaluation. D refers to number of distractors and S to the room size.

Task name	S	D	Description
Find X (Set A or B)	4×4	5	The agent is spawned randomly. Room has 3 objects from Set A (or B) and 3 from $C \cup D$ and instructed to go to an object from Set A (or B). The purpose of these training tasks is to associate objects from Set A and B with their names and the "find" instruction with finding them.
Find Y (Set $C \cup D$)	4×4	5	The agent is spawned randomly. Room has 3 objects from Set A (or B) and 3 from $C \cup D$ and instructed to go to an object from Set $C \cup D$. The purpose of these training tasks is to associate objects from Set $C \cup D$ with their names and the "find" instruction with finding them.
Lift X (Set A or B)	4×4	5	The agent is spawned randomly. Room has 3 objects from Set A (or B) and 3 from $C \cup D$ and instructed to lift an object from Set A (or B). The purpose of these training tasks is to associate the "lift" instruction with lifting the said object.
Put X near Y (X = Set A or B, Y = Set $C \cup D$)	3×3	0	The agent is spawned randomly. Room has 1 object from Set A (or B) and 1 from $C \cup D$ and instructed to put the object from Set A (or B) near the other. The purpose of these training tasks is to learn to move one object near another before putting it on it.
Put X on Y (X = Set A or B, Y = Set $C \cup D$)	3×3	0	The agent is spawned randomly. Room has 1 object from Set A (or B) and 1 from $C \cup D$ and instructed to put the object from Set A (or B) on top of the other. The purpose of these training tasks is to learn to move one object and place it on top of another.
Put X on Y (X = A, Y = D or X = B, Y = C)	4×4	4	The agent is spawned randomly. Room has 3 objects from Set A (or B) and 3 from Set D (or C) and instructed to put the object from Set A (or B) on top of the other. This is the training task most similar to the test task and requires mastering all the other ones.
Put X on Y **(test)** (X = A, Y = C or X = B, Y = D)	4×4	4	The agent is spawned randomly. Room has 3 objects from Set A (or B) and 3 from Set C (or D) and instructed to put the object from Set A (or B) on top of the other. This is the test task.

Table 7: 3d Unity Environment action space.

General body movement	**Fine grain movement**
NOOP	MOVE_RIGHT_SLIGHTLY
MOVE_FORWARD_FULL	MOVE_LEFT_SLIGHTLY
MOVE_BACKWARD_FULL	LOOK_RIGHT_MID
MOVE_RIGHT_FULL	LOOK_LEFT_MID
MOVE_LEFT_FULL	LOOK_DOWN_MID
LOOK_RIGHT_FULL	LOOK_UP_MID
LOOK_LEFT_FULL	LOOK_RIGHT_SLIGHTLY
LOOK_DOWN_FULL	LOOK_LEFT_SLIGHTLY
LOOK_UP_FULL	
Fine grained movement with grip	**General body movement with grip**
GRAB + MOVE_RIGHT_MID	GRAB
GRAB + MOVE_LEFT_MID	GRAB + MOVE_FORWARD_FULL
GRAB + LOOK_RIGHT_MID	GRAB + MOVE_BACKWARD_FULL
GRAB + LOOK_LEFT_MID	GRAB + MOVE_RIGHT_FULL
GRAB + LOOK_DOWN_MID	GRAB + MOVE_LEFT_FULL
GRAB + LOOK_UP_MID	GRAB + LOOK_RIGHT_FULL
GRAB + LOOK_RIGHT_SLIGHTLY	GRAB + LOOK_LEFT_FULL
GRAB + LOOK_LEFT_SLIGHTLY	GRAB + LOOK_DOWN_FULL
GRAB + PULL_CLOSER_MID	GRAB + LOOK_UP_FULL
GRAB + PUSH_AWAY_MID	
Object manipulation	
GRAB + SPIN_RIGHT	
GRAB + SPIN_LEFT	
GRAB + SPIN_UP	
GRAB + SPIN_DOWN	
GRAB + SPIN_FORWARD	
GRAB + SPIN_BACKWARD	
GRAB + PULL_CLOSER_FULL	
GRAB + PUSH_AWAY_FULL	
PULL_CLOSER_MID	
PUSH_AWAY_MID	