Embodied Representation Alignment with Mirror Neurons

Supplementary Material

A. Theoretical Analysis

Theorem 1. *The mutual information between the action understanding representation $\boldsymbol{u}$ and the embodied execution representation $\boldsymbol{e}$ can be estimated by optimizing the transformations $\mathcal{T}_u$ and $\mathcal{T}_e$ to minimize the bidirectional alignment loss $\mathcal{L}_{align}$.*

Proof. The mutual information between $\boldsymbol{u}$ and $\boldsymbol{e}$ is defined as:

$$I(\boldsymbol{u};\boldsymbol{e}) = D_{\mathrm{KL}}(p(\boldsymbol{u},\boldsymbol{e}) \| p(\boldsymbol{u})p(\boldsymbol{e})). \tag{7}$$

Since direct computation is intractable, we introduce trainable transformations:

$$\boldsymbol{z}_u = \mathcal{T}_u(\boldsymbol{u}), \quad \boldsymbol{z}_e = \mathcal{T}_e(\boldsymbol{e}), \tag{8}$$

where $\mathcal{T}_u$ and $\mathcal{T}_e$ are optimized via a loss function. By the *Data Processing Inequality (DPI)*, these transformations satisfy:

$$I(\boldsymbol{z}_u;\boldsymbol{z}_e) \leq I(\boldsymbol{u};\boldsymbol{e}), \tag{9}$$

with equality if $\mathcal{T}_u$ and $\mathcal{T}_e$ preserve all relevant information. Thus, we estimate $I(\boldsymbol{u};\boldsymbol{e})$ indirectly via $I(\boldsymbol{z}_u;\boldsymbol{z}_e)$.

To estimate $I(\boldsymbol{z}_u;\boldsymbol{z}_e)$, we approximate the conditional distributions using contrastive learning. For a batch of size B, define:

$$\hat{p}(\boldsymbol{z}_u|\boldsymbol{z}_e) = \frac{\exp(\mathrm{sim}(\boldsymbol{z}_u,\boldsymbol{z}_e)/\tau)}{\sum_{j=1}^{B}\exp(\mathrm{sim}(\boldsymbol{z}_u,\boldsymbol{z}_e^{(j)})/\tau)}, \tag{10}$$

where $\boldsymbol{z}_e^{(j)}$ are batch samples, sim is a similarity function (*e.g.*, cosine similarity), and τ is a temperature parameter. This approximates the intractable sum over $p(\boldsymbol{z}_e)$.

Since KL divergence is non-negative:

$$D_{\mathrm{KL}}(p(\boldsymbol{z}_u|\boldsymbol{z}_e)\|\hat{p}(\boldsymbol{z}_u|\boldsymbol{z}_e)) \geq 0, \tag{11}$$

it follows that:

$$\mathbb{E}_{p(\boldsymbol{z}_u,\boldsymbol{z}_e)}[\log p(\boldsymbol{z}_u|\boldsymbol{z}_e)] \geq \mathbb{E}_{p(\boldsymbol{z}_u,\boldsymbol{z}_e)}[\log \hat{p}(\boldsymbol{z}_u|\boldsymbol{z}_e)]. \tag{12}$$

Similarly, for the reverse direction:

$$\hat{p}(\boldsymbol{z}_e|\boldsymbol{z}_u) = \frac{\exp(\mathrm{sim}(\boldsymbol{z}_e,\boldsymbol{z}_u)/\tau)}{\sum_{j=1}^{B}\exp(\mathrm{sim}(\boldsymbol{z}_e,\boldsymbol{z}_u^{(j)})/\tau)}, \tag{13}$$

and:

$$\mathbb{E}_{p(\boldsymbol{z}_u,\boldsymbol{z}_e)}[\log p(\boldsymbol{z}_e|\boldsymbol{z}_u)] \geq \mathbb{E}_{p(\boldsymbol{z}_u,\boldsymbol{z}_e)}[\log \hat{p}(\boldsymbol{z}_e|\boldsymbol{z}_u)]. \tag{14}$$

The mutual information $I(\boldsymbol{z}_u;\boldsymbol{z}_e)$ can be expressed as:

$$I(\boldsymbol{z}_u;\boldsymbol{z}_e) = \mathbb{E}_{p(\boldsymbol{z}_u,\boldsymbol{z}_e)}[\log p(\boldsymbol{z}_u|\boldsymbol{z}_e)] - \mathbb{E}_{p(\boldsymbol{z}_u)}[\log p(\boldsymbol{z}_u)]. \tag{15}$$

Assuming the batch approximates $p(\boldsymbol{z}_u)$ as uniform (a common heuristic in contrastive learning):

$$\mathbb{E}_{p(\boldsymbol{z}_u)}[\log p(\boldsymbol{z}_u)] \approx -\log B. \tag{16}$$

Define the single-direction InfoNCE loss:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(\mathrm{sim}(\boldsymbol{z}_u^{(i)},\boldsymbol{z}_e^{(i)})/\tau)}{\sum_{j=1}^{B}\exp(\mathrm{sim}(\boldsymbol{z}_u^{(i)},\boldsymbol{z}_e^{(j)})/\tau)}. \tag{17}$$

Substituting into the mutual information bound:

$$I(\boldsymbol{z}_u;\boldsymbol{z}_e) \geq \log B - \mathcal{L}_{\text{InfoNCE}}. \tag{18}$$

Given the bidirectional alignment loss defined earlier in Equation 2, we note that:

$$\mathcal{L}_{\text{align}} = \frac{1}{2}\left(\mathcal{L}_{u\to e} + \mathcal{L}_{e\to u}\right), \tag{19}$$

where $\mathcal{L}_{u\to e}$ is the InfoNCE loss from $\boldsymbol{z}_u$ to $\boldsymbol{z}_e$, and $\mathcal{L}_{e\to u}$ is from $\boldsymbol{z}_e$ to $\boldsymbol{z}_u$. Since each provides a bound:

$$I(\boldsymbol{z}_u;\boldsymbol{z}_e) \geq \log B - \mathcal{L}_{u\to e}, \quad I(\boldsymbol{z}_u;\boldsymbol{z}_e) \geq \log B - \mathcal{L}_{e\to u}, \tag{20}$$

substituting $\mathcal{L}_{\text{align}}$, the combined lower bound becomes:

$$I(\boldsymbol{z}_u;\boldsymbol{z}_e) \geq \log B - \mathcal{L}_{\text{align}}. \tag{21}$$

By optimizing $\mathcal{T}_u$ and $\mathcal{T}_e$ to minimize $\mathcal{L}_{\text{align}}$, the lower bound $\log B - \mathcal{L}_{\text{align}}$ is maximized. If $\mathcal{T}_u$ and $\mathcal{T}_e$ preserve sufficient information, $I(\boldsymbol{z}_u;\boldsymbol{z}_e)$ approximates $I(\boldsymbol{u};\boldsymbol{e})$, providing an indirect estimate of $I(\boldsymbol{u};\boldsymbol{e})$. This completes the proof. □

Theorem 2. *The mutual information $I(\boldsymbol{u};\boldsymbol{e})$ between the action understanding representation $\boldsymbol{u}$ generated by the model $\mathcal{U}(\cdot;\theta_u)$ and the embodied execution representation $\boldsymbol{e}$ generated by the model $\mathcal{E}(\cdot;\theta_e)$ can be maximized by simultaneously optimizing $\mathcal{U}(\cdot;\theta_u)$ and $\mathcal{E}(\cdot;\theta_e)$, along with linear transformations $\mathcal{T}_u(\cdot;\phi_u)$ and $\mathcal{T}_e(\cdot;\phi_e)$, to minimize the bidirectional alignment loss $\mathcal{L}_{align}$, provided $\mathcal{T}_u(\cdot;\phi_u)$ and $\mathcal{T}_e(\cdot;\phi_e)$ preserve sufficient information.*

Proof. Let $\boldsymbol{u}$ and $\boldsymbol{e}$ be representations generated by the action understanding model $\mathcal{U}$ and the embodied execution model $\mathcal{E}$, parameterized by θ_u and θ_e, respectively. Define linear transformations:

$$\boldsymbol{z}_u = \mathcal{T}_u(\boldsymbol{u};\phi_u), \quad \boldsymbol{z}_e = \mathcal{T}_e(\boldsymbol{e};\phi_e), \tag{22}$$

Task	Variation Type	# of Variations	Avg. Keyframes	Language Template
`open drawer`	placement	3	3.0	"open the ___ drawer"
`slide block`	color	4	4.7	"slide the block to ___ target"
`sweep to dustpan`	size	2	4.6	"sweep dirt to the ___ dustpan"
`meat off grill`	category	2	5.0	"take the ___ off the grill"
`turn tap`	placement	2	2.0	"turn ___ tap"
`put in drawer`	placement	3	12.0	"put the item in the ___ drawer"
`close jar`	color	20	6.0	"close the ___ jar"
`drag stick`	color	20	6.0	"use the stick to drag the cube onto the ___ target"
`stack blocks`	color, count	60	14.6	"stack ___ ___ blocks"
`screw bulb`	color	20	7.0	"screw in the ___ light bulb"
`put in safe`	placement	3	5.0	"put the money away in the safe on the ___ shelf"
`place wine`	placement	3	5.0	"stack the wine bottle to the ___ of the rack"
`put in cupboard`	category	9	5.0	"put the ___ in the cupboard"
`sort shape`	shape	5	5.0	"put the ___ in the shape sorter"
`push buttons`	color	50	3.8	"push the ___ button, [then the ___ button]"
`insert peg`	color	20	5.0	"put the ring on the ___ spoke"
`stack cups`	color	20	10.0	"stack the other cups on top of the ___ cup"
`place cups`	count	3	11.5	"place ___ cups on the cup holder"

Table 4. Language-conditioned tasks and variations in RLBench [18].

where ϕ_u and ϕ_e are the parameters of $\mathcal{T}_u$ and $\mathcal{T}_e$. The mutual information satisfies:

$$I(\boldsymbol{z}_u; \boldsymbol{z}_e) \leq I(\boldsymbol{u}; \boldsymbol{e}), \tag{23}$$

with equality if $\mathcal{T}_u$ and $\mathcal{T}_e$ are invertible.

Based on the bidirectional alignment loss as defined in Eq. (2), we optimize the parameters:

$$\{\theta_u^*, \theta_e^*, \phi_u^*, \phi_e^*\} = \arg\min_{\theta_u, \theta_e, \phi_u, \phi_e} \mathcal{L}_{\text{align}}(\theta_u, \theta_e, \phi_u, \phi_e), \tag{24}$$

augmenting the original objectives of $\mathcal{U}$ and $\mathcal{E}$. This minimizes $\mathcal{L}_{\text{align}}$, maximizing:

$$I(\boldsymbol{z}_u; \boldsymbol{z}_e) \geq \ logB - \mathcal{L}_{\text{align}}(\theta_u^*, \theta_e^*, \phi_u^*, \phi_e^*). \tag{25}$$

Optimizing θ_u and θ_e adjusts $\boldsymbol{u}$ and $\boldsymbol{e}$ to increase $I(\boldsymbol{u}; \boldsymbol{e})$, while optimizing ϕ_u and ϕ_e aligns $\boldsymbol{z}_u$ and $\boldsymbol{z}_e$ with the constraint $I(\boldsymbol{z}_u; \boldsymbol{z}_e) \leq I(\boldsymbol{u}; \boldsymbol{e})$. Assuming $\mathcal{T}_u$ and $\mathcal{T}_e$ preserve sufficient information, joint optimization reduces information loss between $\boldsymbol{u}, \boldsymbol{e}$ and $\boldsymbol{z}_u, \boldsymbol{z}_e$, allowing $I(\boldsymbol{z}_u; \boldsymbol{z}_e)$ to closely approximate $I(\boldsymbol{u}; \boldsymbol{e})$. Hence, maximizing $I(\boldsymbol{z}_u; \boldsymbol{z}_e)$ through $\mathcal{L}_{\text{align}}$ also maximizes $I(\boldsymbol{u}; \boldsymbol{e})$, completing the proof. □

B. Environment Details

Tasks Our *action recognition* and *embodied execution* tasks follow the multi-task definition from previous work [12, 13, 37, 44] based on RLBench [18]. Specifically, there are 18 tasks with 249 variations, defined through diverse language instructions. These tasks include non-prehensile actions such as *push buttons*, common pick-and-place tasks like *place wine*, and high-precision peg-in-hole tasks such as *insert peg*. Table 4 provides an overview of these tasks.

Variations Task variations include randomly sampled colors, sizes, shapes, counts, placements, and categories of objects. The set of colors include 20 instances: `colors` = {`red`, `maroon`, `lime`, `green`, `blue`, `navy`, `yellow`, `cyan`, `magenta`, `silver`, `gray`, `orange`, `olive`, `purple`, `teal`, `azure`, `violet`, `rose`, `black`, `white`}. The set of sizes include 2 instances: `sizes` = {`short`, `tall`}. The set of shapes include 5 instances: `shapes` = {`cube`, `cylinder`, `triangle`, `star`, `moon`}. The set of counts include 3 instances: `counts` = {`1`, `2`, `3`}. The placements and object categories are specific to each task. For instance, `open drawer` has 3 placement locations: `top`, `middle`, and `bottom`, and `put in cupboard` includes 9 YCB objects. In addition to these semantic variations, objects are placed on the tabletop at random poses. Some large objects like drawers have constrained pose variations [18] to ensure that manipulating them is kinematically feasible with the Franka arm.

C. Implementation Details

C.1. Action Recognition

We follow ViCLIP [41] to implement the action recognition module. Specifically, the video encoder uses a standard ViT with spatiotemporal attention [5]. Random patch masking is applied to the input videos during pretraining, which significantly alleviates the computational burden. We use the model weights pretrained on InternVid [41] and fine-tune on video-text pairs of object interactions simulated in RLBench [18]. The training objective is to align the corresponding video and text embeddings, similar to CLIP [31], using contrastive learning with a temperature parameter $\tau_{\text{viclip}} = 0.05$.

For action recognition evaluation, given an input video, we compute its video embedding and compare it with the text embeddings of all possible action classes using cosine similarity. The class with the highest similarity score is selected as the predicted label.

C.2. Embodied Execution

We follow ARP [44] to implement the action recognition module. The experimental settings are consistent with prior works [12, 13, 37, 44]. The input RGB-D images have a resolution of 128 × 128 and are captured by four noiseless cameras mounted at the front, left shoulder, right shoulder, and wrist of the robot.

We use the next key end-effector pose as the control interface, eliminating the need for high-frequency actions. Consequently, neither the horizon nor action steps are applicable. Instead, low-level robot movements are generated using RLBench's built-in RRT planner. We use a chunk size of 2 for binary gripper states and a chunk size of 1 for end-effector positions and rotations. For example, ARP first predicts the roll, followed by the pitch and yaw of the rotation Euler angles. Following the strategy of RVT-2 [13], we first predict coarse positions and then refine them by zooming into the images (with updated vision features) to obtain more accurate positions. The end-effector positions are initially predicted in 2D, and the corresponding 3D positions are derived from the 2D coordinates in each viewpoint. Table 5 presents the training parameters.

C.3. Joint Training

We perform end-to-end joint training of both tasks and representation alignment, as previously discussed. Since action recognition is easier to learn than embodied execution, we control the learning frequency of action recognition to 20% to balance the training pace. We use a batch size of 192 and train for 25 hours on 2 NVIDIA A100 80GB GPUs.

D. Additional Results

D.1. Embodied Execution

We present an additional visualization comparison that contains more failure patterns in Fig. 7.

D.2. Latent Representation

We present a comparative visualization of latent representations (via t-SNE [39]) over different training iterations and involving a total of 18 tasks in Fig. 8 to Fig. 16.

Table 5. Hyperparameters used for the embodied execution module on RLBench.

Hyperparameter	**Value**
Model	
number of layers	8
embedding size	128
mlp size	512
backbone	MVT [12]
Action Sequence	
chunk size	mix of 2 and 1
Train & Eval	
observation	RGBD $4 \times 128 \times 128 \times 4$
maximum evaluation steps	25
train iterations	80000
eval frequency	10000
batch size	96*2
learning rate	1.25e-5
learning rate scheduler	cosine
optimizer	LAMB

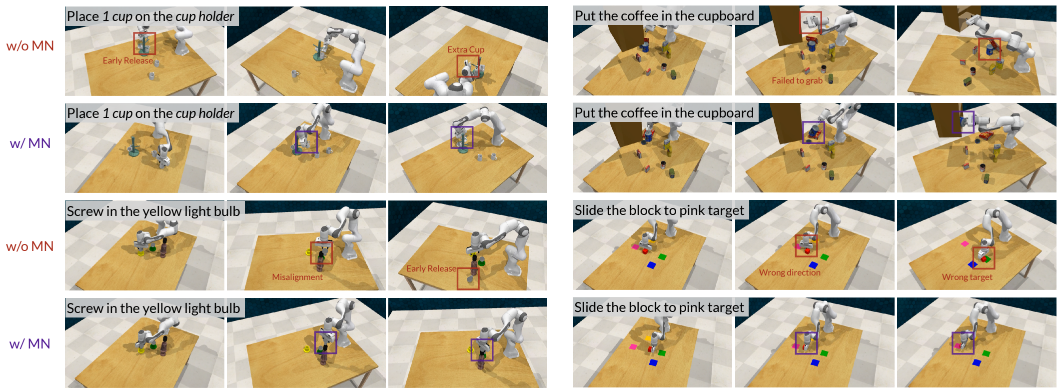

Figure 7. Visualization comparison of embodied execution results. Key details are highlighted in boxes.

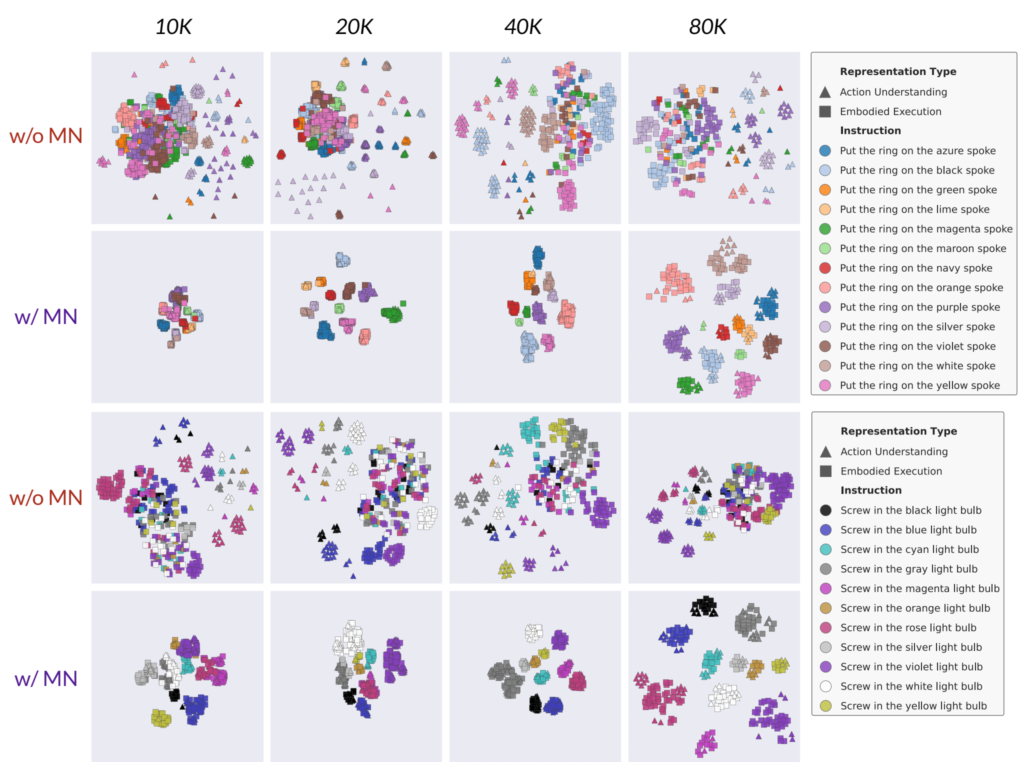

Figure 8. Comparative visualization of latent representations over iterations of task *Insert onto square spoke* and *Screw in the light bulb*.

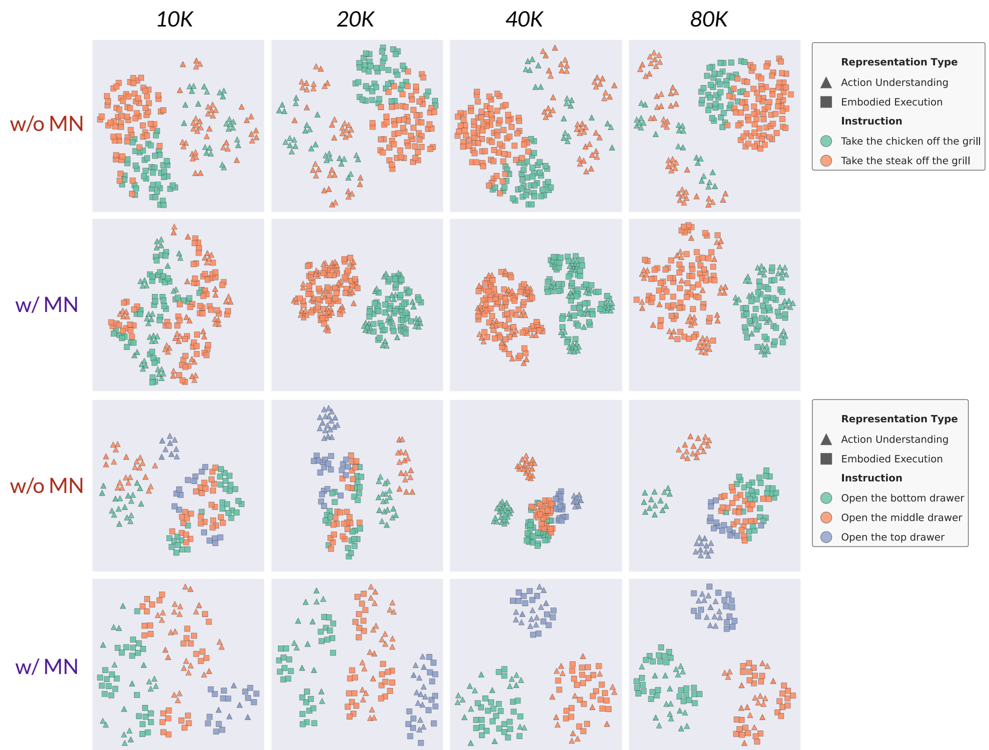

Figure 9. Comparative visualization of latent representations over iterations of task *Meat off grill* and *Open the drawer*.

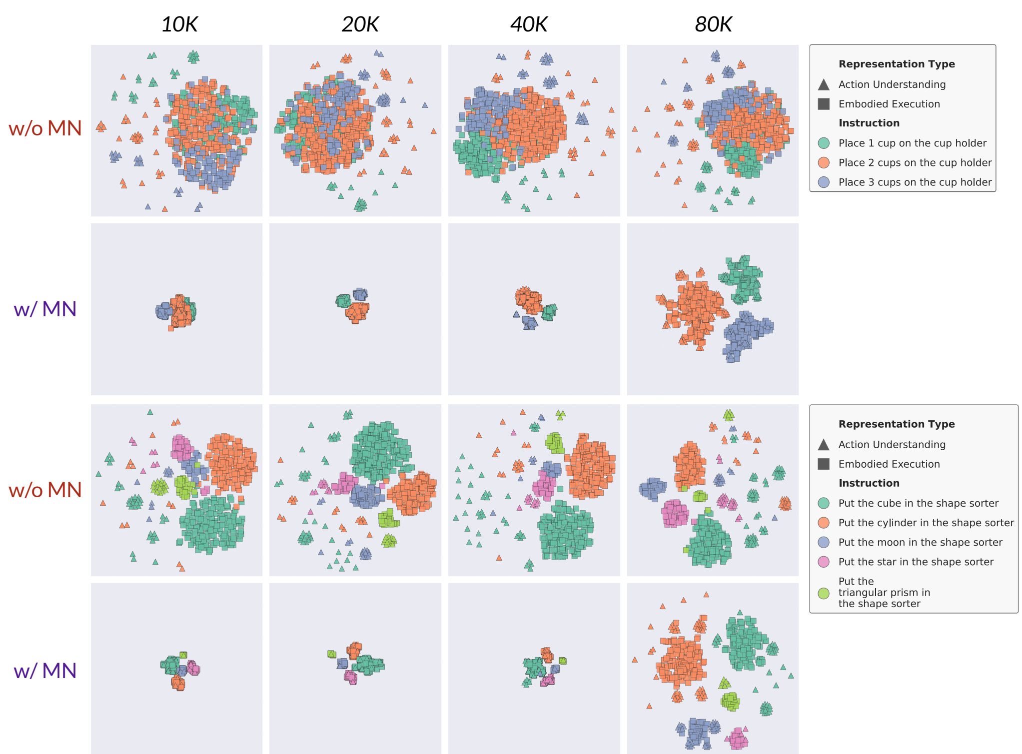

Figure 10. Comparative visualization of latent representations over iterations of task *Place cups* and *Sort shapes*.

Figure 11. Comparative visualization of latent representations over iterations of task *Place wine at rack* and *Push buttons*.

Figure 12. Comparative visualization of latent representations over iterations of task *Put groceries in cupboard* and *Put items in drawer*.

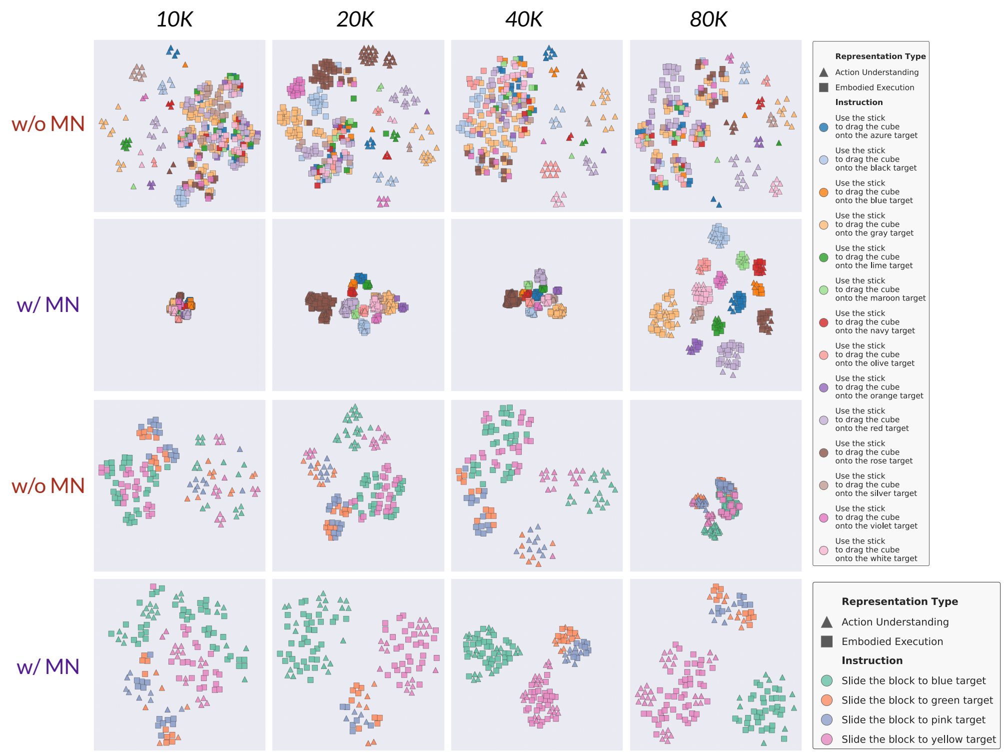

Figure 13. Comparative visualization of latent representations over iterations of task *Drag Stick* and *Slide block to color target*.

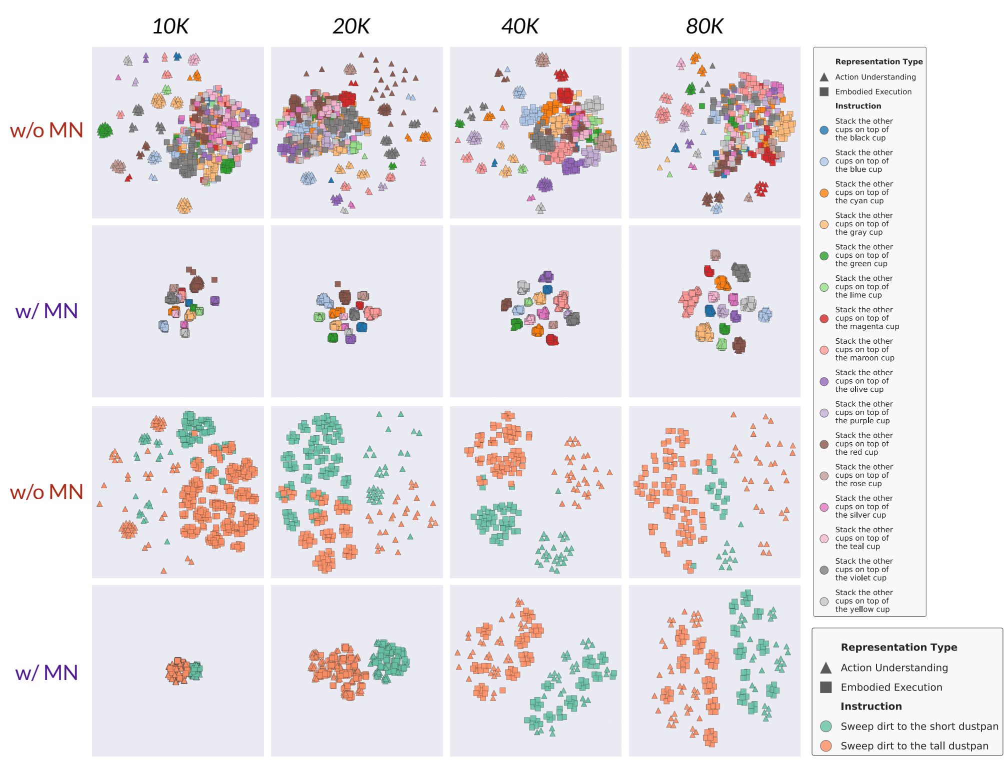

Figure 14. Comparative visualization of latent representations over iterations of task *Stack cups* and *Sweep dirt to dustpan*.

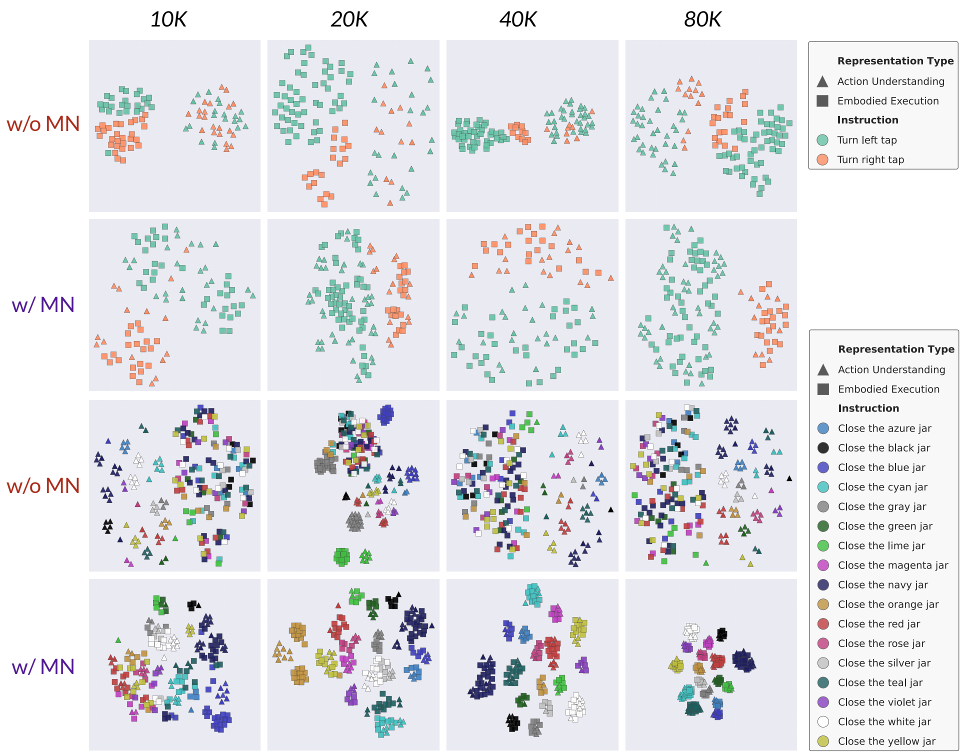

Figure 15. Comparative visualization of latent representations over iterations of task *Turn tap* and *Close jar*.

Figure 16. Comparative visualization of latent representations over iterations of task *Put money in safe* and *Stack blocks*.