Entity-Centric Reinforcement Learning for Object Manipulation from Pixels

Dan Haramati
haramati@campus.technion.ac.il
Technion - Israel Institute of Technology

Tal Daniel
taldanielm@campus.technion.ac.il
Technion - Israel Institute of Technology

Aviv Tamar
avivt@technion.ac.il
Technion - Israel Institute of Technology

Abstract

Manipulating objects is a hallmark of human intelligence, and an important task in domains such as robotics. In principle, Reinforcement Learning (RL) offers a general approach to learn object manipulation. In practice, however, domains with more than a few objects are difficult for RL agents due to the curse of dimensionality, especially when learning from raw image observations. In this work we propose a structured approach for visual RL that is suitable for representing multiple objects and their interaction, and use it to learn goal-conditioned manipulation of several objects. Key to our method is the ability to handle goals with dependencies between the objects (e.g., moving objects in a certain order). We further relate our architecture to the generalization capability of the trained agent, and demonstrate agents that learn with 3 objects but generalize to similar tasks with over 10 objects. Rollout videos are available on our website: https://sites.google.com/view/entity-centric-rl

Introduction: While some studies focus on developing general agents that can solve a wide range of tasks [27], the difficulty of particular problems has motivated studying agents that incorporate *structure* into the learning algorithm [20]. Object manipulation – our focus in this work – is a clear example for the necessity of structure: as the number of degrees of freedom of the system grows exponentially with the number of objects, the curse of dimensionality inhibits standard approaches from learning. For problems such as robotic manipulation with image inputs, however, how to factor the state (images of robot and objects) into individual entities and their attributes (positions, orientation, etc.) is not trivial. Our main contribution in this work is a goal-conditioned RL framework for multi-object manipulation from pixels. Our approach consists of two components: (1) an unsupervised object-centric image representation (OCR), which extracts entities and their attributes from image data; and (2) a transformer-based architecture for the policy and Q-function neural networks that we name *Entity Interaction Transformer* (EIT). Combined with our choice of deep latent particles (DLP) [4] image representations, we demonstrate what is, to the best of our knowledge, the most accurate object manipulation from pixels involving more than two objects, or with goals that require interactions between objects. Finally, we demonstrate that an EIT trained on manipulating up to 3 objects can perform well on tasks with up to 6 objects, and in certain tasks we show generalization to new goals containing over 10 objects.

Related Work: Previous work has proposed unsupervised single-vector latent representations of images as goal and state spaces for solving goal-conditioned manipulation tasks with RL [21, 22]. In tasks involving multiple objects, this approach falls short when compared to models based on structured representations such as SMORL [34], which employs an object-centric image representation. The key assumption in SMORL is that multi-object goals can be addressed sequentially (object by

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

object) and independently, overlooking potential interactions among objects that might influence reaching goals. Our model, on the other hand, considers interaction between entities, leading to improved performance and generalization. Further related work is discussed in Appendix A.

Deep Latent Particles (DLP): DLP [4, 5] is an unsupervised object-centric model for images. DLP provides a disentangled latent space structured as a *set* of foreground particles $z = \{(z_p, z_s, z_d, z_t, z_f)_i\}_{i=0}^{K-1} \in \mathbb{R}^{K\times(6+l)}$, where K is the number of particles, $z_p \in \mathbb{R}^2$ is a *keypoint*, the position of the particle as (x, y) coordinates in Euclidean pixel space, $z_s \in \mathbb{R}^2$ is a scale attribute, $z_d \in \mathbb{R}$ is a pixel space "depth" attribute, $z_t \in \mathbb{R}$ is a transparency attribute and $z_f \in \mathbb{R}^l$ are the latent features that encode the visual appearance in the particle region, where l is the dimension of learned visual features. DLP also assigns an abstract particle for the background, located in the center of the image, and described by m_{bg} latent background visual features, $z_{\text{bg}} \in \mathbb{R}^{m_{\text{bg}}}$.

Method: Our approach is *entity-centric* – it is structured to decompose the input into individual entities, each represented by a latent feature vector, and learn the relationships between them. Our method consists of the following 2 components: (1) **Object centric representation (OCR) of Images** – we extract a representation of state and goal images consisting of a set of latent vectors using a pretrained model; (2) **Entity interaction transformer (EIT)** – we feed the sets of latent vectors, extracted from multiple viewpoints, to a transformer-based architecture for the RL policy and Q-function neural networks.

Given a tuple of image observations from K different viewpoints of the state $(I_1^s, \ldots, I_K^s)$ and goal $(I_1^g, \ldots, I_K^g)$, we process each image *separately* using a pretrained DLP model, extracting a set of M vectors $\{p_m^k\}_{m=1}^M$, k indexing the viewpoint of the source image, which we will refer to as (latent) particles. We denote particle m of state image I_k^s by p_m^k and of goal image I_k^g by q_m^k. The input to the goal-conditioned policy is a tuple of the $2K$ sets: $\big(\{p_m^1\}_{m=1}^M, \ldots, \{p_m^K\}_{m=1}^M, \{q_m^1\}_{m=1}^M \ \ldots, \{q_m^K\}_{m=1}^M\big)$. For the Q-function, $Q(s, a, g)$, an action particle $p_a \in \mathbb{R}^{6+l}$ is added to the input, obtained by learning a projection from the action dimension d_a to the latent particle dimension. An outline of the transformer-based architecture is presented in Figure 2. Compared to previous goal-conditioned methods' use of the attention mechanism, we use it to explicitly model relationships between entities from both state and goal across multiple viewpoints and do not assume privileged entity-entity matching information. See further method details in Appendix C.

Experiments: We separate our investigation into 2 main parts: (1) we focus on the design of our OCR and EIT, and how it handles complex interactions between objects, and (2) we study compositional generalization. We employ our method with TD3 [7] and HER [1] in a tabletop cube manipulation environment. Further implementation details as well as details about the environment, metrics and baselines can be found in Appendix D.

Multi-object Manipulation Results are presented in Figure 4. We observe that with a single object, all methods succeed, yet the unstructured baselines are less sample efficient. With more than 1 object, the image-based unstructured baseline is not able to learn at all, while the unstructured state-based baseline is significantly outperformed by the structured methods. Notably, on 3 objects, *our image-based method surpasses the unstructured state-based* method. In environments that **require interaction between objects**, our method outperforms SMORL using state input. Moreover, with image inputs our method outperforms SMORL with state inputs, demonstrating that SMORL is fundamentally limited in performing these more complex tasks.

Compositional Generalization We investigate our method's ability to achieve zero-shot compositional generalization. *Different number of cubes than in training* - visual results on 6 cubes are presented in Figure 5 (left) and evaluation metrics in Table 4. Our agent generalizes to a changing number of objects with some decay in performance as the number of objects grows. *Cube sorting* - we provide a goal image containing $X \leq 3$ cubes of different colors and then deploy a policy of an agent trained on 3 cubes in an environment containing $4X$ cubes, 4 of each color. The agent sorts the cubes around each goal cube position with matching color, and is also able to perform the task with cube colors unseen during RL training. Visual results on 12 cubes in 3 colors are presented in Figure 6. Further generalization capabilities of our agent are detailed in the Appendix F and demonstrated on our website.

Conclusion: We proposed an RL framework for object manipulation from images, composed of an off-the-shelf object-centric image representation, and a novel transformer-based policy architecture that can account for multiple viewpoints and interactions between entities. We have conducted

experiments that demonstrate non-trivial generalization behavior of our trained policies. Future work can explore multi-modal goal specification (e.g., language), and more expressive sensing (depth cameras, contact/force sensors, etc.), which could be integrated as additional input entities to the EIT.

Acknowledgements

The research was Funded by the European Union (ERC, Bayes-RL, 101041250). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency (ERCEA). Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. In *International Conference on Machine Learning*, pages 4644–4665. PMLR, 2022.

[5] Tal Daniel and Aviv Tamar. Ddlp: Unsupervised object-centric video prediction with deep dynamic latent particles. *arXiv preprint arXiv:2306.05957*, 2023.

[6] Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: Object-centric world models for robotics manipulation. *arXiv preprint arXiv:2307.02427*, 2023.

[7] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[8] Kevin Gmelin, Shikhar Bahl, Russell Mendonca, and Deepak Pathak. Efficient RL via disentangled environment and agent representations. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 11525–11545. PMLR, 23–29 Jul 2023.

[9] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019.

[10] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[11] Negin Heravi, Ayzaan Wahid, Corey Lynch, Pete Florence, Travis Armstrong, Jonathan Tompson, Pierre Sermanet, Jeannette Bohg, and Debidatta Dwibedi. Visuomotor control in multi-object scenes using object-aware representations. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9515–9522. IEEE, 2023.

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[13] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *Proceedings of ICLR 2020*. OpenReview.net, 2020. URL `https://openreview.net/pdf?id=SJxrKgStDH`.

[14] Andrej Karpathy. `https://github.com/karpathy/minGPT`, 2021.

[15] Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 ieee international conference on robotics and automation (icra)*, pages 4051–4058. IEEE, 2020.

[16] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

[17] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[18] Davide Mambelli, Frederik Träuble, Stefan Bauer, Bernhard Schölkopf, and Francesco Locatello. Compositional multi-object reinforcement learning with linear relation networks. *arXiv preprint arXiv:2201.13388*, 2022.

[19] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

[20] Aditya Mohan, Amy Zhang, and Marius Lindauer. Structure in reinforcement learning: A survey and open problems. *arXiv preprint arXiv:2306.16021*, 2023.

[21] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

[22] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[23] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[24] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[26] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[28] Gautam Singh, Sungjin Ahn, and Fei Deng. Illiterate dall-e learns to compose. In *The Tenth International Conference on Learning Representations, ICLR2022*. The International Conference on Learning Representations (ICLR), 2022.

[29] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[31] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. In *In Advances in Neural Information Processing Systems (NeurIPS), 2021*, 2021.

[32] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021.

[33] Jaesik Yoon, Yi-Fu Wu, Heechul Bae, and Sungjin Ahn. An investigation into pre-training object-centric representations for reinforcement learning. *arXiv preprint arXiv:2302.04419*, 2023.

[34] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised visual reinforcement learning with object-centric representations. *arXiv preprint arXiv:2011.14381*, 2020.

[35] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[36] Allan Zhou, Vikash Kumar, Chelsea Finn, and Aravind Rajeswaran. Policy architectures for compositional generalization in control. *arXiv preprint arXiv:2203.05960*, 2022.

Figure 1: *The environment we used for our experiments (Left) and how the agent perceives it (middle, right), colored keypoints are the position attribute (z_p) of particles from DLP.*

A Extended Related Work

Latent-based Visual RL: Unstructured latent representations, where each scene is represented by a *single* latent vector, have been widely employed, both in model-free [21, 22, 32] and model-based [10, 19] settings. However, in manipulation tasks involving multiple objects, this approach falls short when compared to models based on structured representations [8, 34].
Object-centric RL: Several recent works explored policy architectures for a structured representation of the state: Li et al. [15] employ relational networks [26, 3], Zhou et al. [36] explore attention [2, 30] and deep sets [35], while Mambelli et al. [18] strive to reduce architecture complexity with linear relation networks. Compared to our approach, the aforementioned methods assume access to ground-truth structured state representation, while we learn representations from images.
Object-centric RL from Pixels: Several works have explored adopting ideas from state-based structured representation methods to learning from visual inputs. FOCUS [6] learns an object-centric world model, demonstrating manipulation of up to 2 objects. In contrast, our method is trained in a model-free setting and we demonstrate learning to manipulate many more objects. OCRL [33] and Heravi et al. [11] have investigated slot-based representations [9, 16, 28] for manipulation tasks, the first in a model-free learning and the latter in an imitation learning setting. Both have demonstrated the clear advantages of these representations over alternative methods. In this work, we utilize particle-based image representations [4] and extend these findings by tackling more complex manipulation tasks and showcasing generalization capabilities. Closely related to our work, SMORL [34] employs SCALOR [13], a patch-based image representation for goal-conditioned manipulation tasks. The key assumption in SMORL is that goal-conditioned multi-object tasks can be addressed sequentially (object by object) and independently, overlooking potential interactions among objects that might influence reaching goals. Our model, on the other hand, considers interaction between entities, leading to improved performance and generalization, as we demonstrate in our experiments.

B Extended Background

Goal-Conditioned Reinforcement Learning (GCRL): RL considers a Markov Decision Process (MDP) [23] defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, P the environment transition dynamics, r the reward function, and γ the discount factor. GCRL additionally includes a goal space $\mathcal{G}$, and the agent seeks to learn a policy $\pi^* : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ that maximizes the expected return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $r_t = r\left(s_t, g\right) : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$ represents immediate rewards at time step t when the state is s_t and the goal is g.
A Q-function $Q^\pi\left(s, a, g\right) : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ parameterized by a deep neural network $Q_\theta\left(s, a, g\right)$ with parameters θ, approximates the expected return given goal g when taking action a at state s and then following the policy π. $Q_\theta\left(s, a, g\right)$ is learned via minimization of the temporal difference (TD) [29] objective given a state, action, next state, goal tuple (s, a, s', g): $\mathcal{L}_{TD}(\theta) = \left[r(s, g) + \gamma Q_{\bar{\theta}}\left(s', \pi(s', g), g\right) - Q_\theta\left(s, a, g\right)\right]^2$, where $Q_{\bar{\theta}}\left(s', a', g\right)$ is a target network with parameters $\bar{\theta}$ which are constant under the TD objective. Specifically in off-policy actor-critic algorithms [7], a policy network $\pi_\phi\left(s, g\right)$ (actor) with parameters ϕ is learned concurrently with the Q-function network (critic) with the objective of maximizing it with respect to the action: $\mathcal{L}_\pi(\phi) = -Q_\theta\left(s, \pi_\phi\left(s, g\right), g\right)$.

The Attention Mechanism: Attention [2] denoted $A(\cdot,\cdot)$ is an operator between two sets of vectors, $X = \{x_i\}_{i=1}^N$ and $Y = \{y_j\}_{j=1}^M$, producing a third set of vectors $Z = \{z_i\}_{i=1}^N$. For simplicity, we describe the case were all input, output and intermediate vectors are in $\mathbb{R}^d$. Denote the key, query and value projection functions $q(\cdot),\ k(\cdot),\ v(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ respectively. The attention operator is defined as $A(X,Y) = Z$ where:

$$z_i = \sum_{j=1}^{N} \alpha\left(x_i, y_j\right) v\left(y_j\right), \quad \alpha\left(x_i, y_j\right) = \mathrm{softmax}_j\left(\frac{q\left(x_i\right) \cdot k\left(y_j\right)}{\sqrt{d}}\right) \in \mathbb{R}.$$

An important property of $A(X,Y)$ is that it is *equivariant* to permutations of X (permutation of elements in X results in the same permutation in the output elements in Z, with no change in individual elements' values) and *invariant* to permutations of Y (permutation of elements in Y does not change the output Z). In the special case where $X = Y$, the operation is termed *self-attention* (SA), and otherwise, *cross-attention* (CA).

Deep Latent Particles (DLP): DLP [4, 5] is an unsupervised object-centric model for images. DLP provides a disentangled latent space structured as a *set* of foreground particles $z = \{(z_p, z_s, z_d, z_t, z_f)_i\}_{i=0}^{K-1} \in \mathbb{R}^{K\times(6+l)}$, where K is the number of particles, $z_p \in \mathbb{R}^2$ is a *keypoint*, the position of the particle as (x,y) coordinates in Euclidean pixel space, $z_s \in \mathbb{R}^2$ is a scale attribute containing the (x,y) dimensions of the bounding-box around the particle, $z_d \in \mathbb{R}$ is a pixel space "depth" attribute used to signify which particle is in front of the other in case there is an overlap, $z_t \in \mathbb{R}$ is a transparency attribute and $z_f \in \mathbb{R}^l$ are the latent features that encode the visual appearance in the particle region, where l is the dimension of learned visual features. In addition, DLP assigns an abstract particle for the background, located in the center of the image, and described by m_{bg} latent background visual features, $z_{\text{bg}} \in \mathbb{R}^{m_{\text{bg}}}$.

C Method

We propose an approach to solving goal-conditioned multi-object manipulation tasks from images. Our approach is *entity-centric* – it is structured to decompose the input into individual entities, each represented by a latent feature vector, and learn the relationships between them. Our method consists of the following 2 components: (1) **Object-Centric Representation (OCR) of Images** – We extract a representation of state and goal images consisting of a set of latent vectors using a pretrained model; (2) **Entity Interaction Transformer (EIT)** – We feed the sets of latent vectors, extracted from multiple viewpoints, to a transformer-based architecture for the RL policy and Q-function neural networks. These two components can be used with standard RL algorithms to optimize a given reward function. During training, the reward function can depend on either the true state of the system (in simulation), or on the image observation itself; during inference, the learned policy depends only on images. As designing a reward function from image inputs can be difficult, we additionally propose a novel reward function that is based on the OCR, and corresponds to moving objects to goal configurations. We term this **Chamfer Reward**, and it enables learning entirely from pixels. We elaborate on the Chamfer reward in Section E. We begin with a general reasoning that underlies our approach.

The complexity tradeoff between representation learning and decision making It is important to understand that the representation learning (OCR) and decision making (EIT) problems are dependent. Consider for example the task of moving objects with a robot arm, as in our experiments. Ideally, the OCR should output the physical state of the robot and each object, which is sufficient for optimal control. However, identifying that the robot is a single entity with several degrees of freedom, while the objects are separate entities, is difficult to learn just from image data, as it pertains to the *dynamics* of the robot. The relevant properties of each object can also be task dependent – for example, the color of the objects may only matter if the task's goal depends on it. Alternatively, one may settle for a much leaner OCR component that does not understand the dynamics of the objects nor their relevance to the task, and delegate the learning of this information to the EIT. Our design choice in this paper is the latter, i.e., *a lean OCR and an expressive EIT*. We posit that this design allows to (1) easily acquire an OCR, and (2) handle multiple views and mismatches between the visible objects in the current state and the goal seamlessly. We next detail our design.

C.1 Object-Centric Representation of Images

The first step in our method requires extracting a compact disentangled OCR from raw pixel observations. Given a tuple of image observations from K different viewpoints of the state $(I_1^s, \dots, I_K^s)$ and goal $(I_1^g, \dots, I_K^g)$, we process each image *separately* using a pretrained Deep Latent Particles (DLP) [4, 5] model, extracting a set of M vectors $\{p_m^k\}_{m=1}^M$, k indexing the viewpoint of the source image, which we will refer to as (latent) particles. We denote particle m of state image I_k^s by p_m^k and of goal image I_k^g by q_m^k. We emphasize that there is *no alignment* between particles from different views (e.g., p_m^1, p_m^2 can correspond to different objects) or between state and goal of the same view (e.g., between p_m^1 and q_m^1). The vectors $p_m^k, q_m^k \in \mathbb{R}^{6+l}$ contain different attributes detailed in the DLP section of the Background B. In contrast to previous object-centric approaches that utilize patch-based [34] or slot-based [33] representations, we adopt DLP, which has recently demonstrated state-of-the-art performance in single-image decomposition and various downstream tasks.

Pretraining the DLP: In this work, we pretrain the DLP from a dataset of images collected by a random policy. We found that in all our experiments, this simple pretraining was sufficient to obtain well performing policies even though the image trajectories in the pretraining data are different from trajectories collected by an optimal policy. We attribute this to the tradeoff described above – the lean single-image based DLP OCR is complemented by a strong EIT that can account for dynamics necessary to solve the task.

C.2 Entity-Centric Architecture

We next describe the EIT, which processes the OCR entities into an action or Q-value. As mentioned above, our choice of a lean OCR requires the EIT to account for the dynamics of the entities and their relation to the task. In particular, we have the following desiderata from the EIT: (1) **Permutation Invariance**; (2) **Handle goal-based RL**; (3) **Handle multiple views**; (4) **Compositional generalization**. Proper use of the attention mechanism provides us with (1). The main difficulty in (2) and (3) is that particles in different views and the goal are not necessarily matched. Thus, we designed our EIT to seamlessly handle unmatched entities. For (4), we compose the EIT using transformer [30] blocks. An outline of the architecture is presented in Figure 2. Compared to previous goal-conditioned methods' use of the attention mechanism, we use it to explicitly model relationships between entities from both state and goal across multiple viewpoints and do not assume privileged entity-entity matching information. A more detailed comparison can be found in Section H. We now describe the EIT in detail:

Input - The EIT policy receives the latent particles extracted from both the current state images and the goal images as input. We inject information on the source viewpoint of each state and goal particle with an additive encoding which is learned concurrently with the rest of the network parameters.
Forward - State particles $\left(\{p_j^1\}_{j=1}^M, \dots, \{p_j^N\}_{j=1}^M\right)$ are processed by a sequence of Transformer blocks: $SA \rightarrow CA \rightarrow SA \rightarrow AA$, denoting *Self-Attention* SA, *Cross-Attention* CA, and *Aggregation-Attention* AA, followed by an MLP.
Goal-conditioning - We condition on the goal particles $\left(\{q_k^1\}_{k=1}^M, \dots, \{q_k^N\}_{k=1}^M\right)$ using the CA block between the state (provide the queries) and goal (provide the keys and values) particles.
Permutation Invariant Output - The AA block reduces the set to a single-vector output and is implemented with a CA block between a single particle with learned features (provides the query) and the output particles from the previous block (provide the keys and values). This permutation invariant operation on the processed state particles, preceded by permutation equivariant transformer blocks, makes the output of the network invariant to permutations of the particles in each set. The aggregated particle is input to an MLP, producing the final output (action/value).
Action Entity - The EIT Q-function, in addition to the state and goal particles, receives an action as input. The action is projected to the dimension of the particles and added to the input set. Treating the action as an individual entity proved to be a significant design choice, see Section F.3.

C.3 Training and Implementation Details

We developed our method with the off-policy algorithm TD3 [7] along with hindsight experience replay (HER, Andrychowicz et al. 1). In principal, our approach is not limited to actor-critic algorithms or to the online setting, and can be used with any deep Q-learning algorithm, online or offline. We pre-train a single DLP model on images from multiple viewpoints of rollouts collected

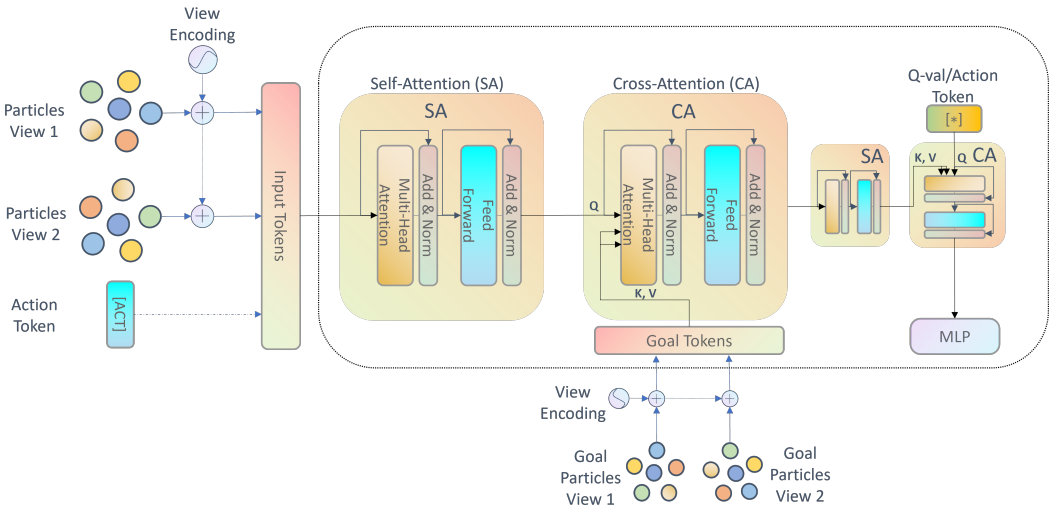


Figure 2: ***Outline of the Entity Interaction Transformer (EIT)*** *Sets of state and goal particles from multiple views with an additive view encoding are input to a sequence of transformer blocks. For the Q-function, an action particle is added. We condition on goals with cross-attention. Attention-based aggregation reduces the set to a single vector, followed by an MLP to produce the final output.*

with a random policy. In the off-policy RL training loop, during the data collection phase, we convert state and goal images to object-centric latent representations with the DLP encoder before inserting them to the replay buffer. We use our EIT architecture for all policy and Q-function neural networks. Further details and hyperparameters can be found in Section G. Our code will be made publicly available.

D Experiments

We design our experimental setup to address the following aspects: (1) benchmarking our method on multi-object manipulation tasks from pixels; (2) assessing the significance of accounting for interactions between entities for the RL agent's performance; (3) evaluating the scalability of our approach to increasing number of objects; (4) analyzing the generalization capabilities of our method.

Environments We evaluate our method on several simulated tabletop robotic object manipulation environments implemented with IsaacGym [17]. The environment includes a robotic arm set in front of a table with a varying number of cubes in different colors. The agent observes the state of the system through a number of cameras in fixed locations, and performs actions in the form of deltas in the end effector coordinates $a = (\Delta x_{ee}, \Delta y_{ee}, \Delta z_{ee})$. At the beginning of each episode, the cube positions are randomly initialized on the table, and a goal configuration is sampled similarly. The goal of the agent is to push the blocks to match the goal configuration. We categorize a suite of tasks as follows (see Figure 3):
`N-Cubes`: Push N different-colored cubes to their goal location.
`Adjacent-Goals`: A `3-Cubes` setting where goals are sampled randomly on the table such that all cubes are adjacent. This task requires accounting for interactions between objects.
`Small-Table`: A `3-Cubes` setting where the table is substantially smaller. This task requires to accurately account for all objects in the scene at all times, to avoid pushing blocks off the table.
`Ordered-Push`: A `2-Cubes` setting where a narrow corridor is set on top of the table such that its width can only fit a single cube. We consider two possible goal configurations: red cube in the rear of the corridor and green cube in the front, or vice versa. This tasks requires to fulfill the goals in a certain order, otherwise the agent fails (pulling a block out of the corridor is not possible).

Reward The reward calculated from the ground-truth state of the system, which we refer to as the ground-truth (GT) reward, is the mean negative L_2 distance between each cube and its desired goal position on the table. The image-based reward calculated from the DLP OCR for our method is the

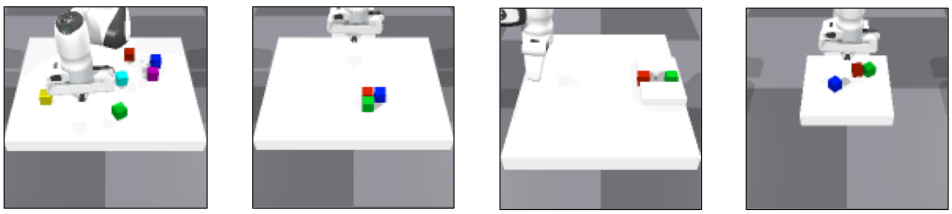


Figure 3: *The simulated environments used for experiments in this work.*

negative GDAC distance (see Eq. 1) between state and goal sets of particles, averaged over viewpoints. Further reward details can be found in the Section G.1.

Evaluation Metrics We evaluate the performance of the agents on several metrics. In this environment, we define an episode a *success* if all N objects are at a threshold distance from their desired goal. The metric most closely captures task success, but does not capture intermediate success or timestep efficiency. To this end, we additionally evaluate based on *success fraction*, *maximum object distance*, *average object distance* and *average return*. For a formal definition of these metrics see Section G.1. All results show means and standard deviations across 3 random seeds.

Baselines We compare our method with the following baselines:

`Unstructured` – a single-vector latent representation of images using a pre-trained VAE is extracted from multiple viewpoints from both state and goal images and then concatenated and fed to an MLP architecture for the policy and Q-function neural networks. This baseline corresponds to methods such as Nair et al. [21], with the additional multi-view data as in our method.

`SMORL` – We re-implement SMORL [34], extending it to multiview inputs and tune its hyper-parameters for the environments in this work. Re-implementation details are available in Section G.4. We use DLP as the pre-trained OCR for this method for a fair comparison. Note that image-based SMORL cannot utilize GT reward since it requires matching the particle selected from the goal image at the beginning of each training episode to the corresponding object in the environment. We therefore do not present such experiments.

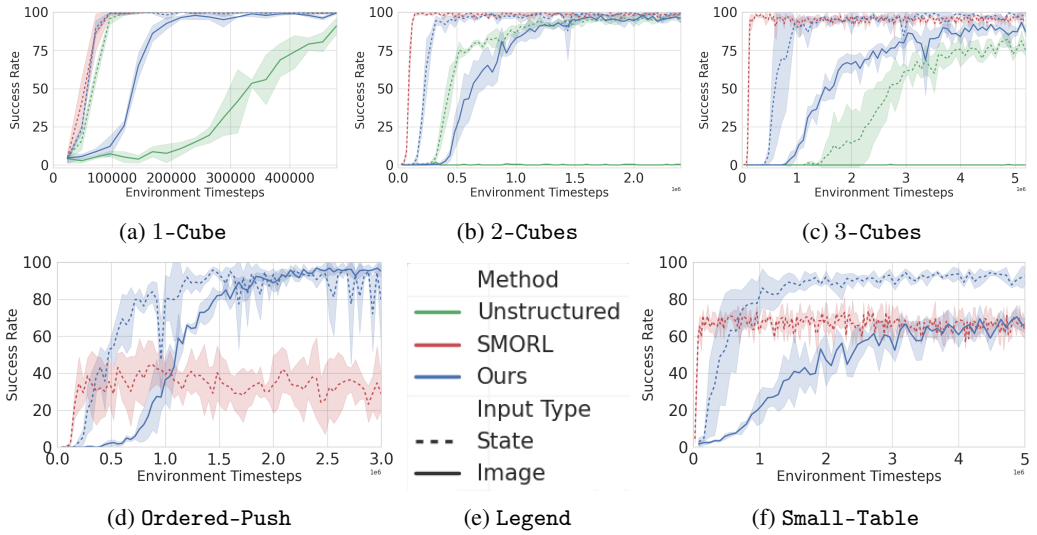
Object-centric Pretraining All image-based methods in this work utilize pre-trained unsupervised image representations trained on data collected with a random policy. For the object-centric methods, we train a single DLP model on data collected from the `6-Cubes` environment. We found it generalizes well to fewer objects and changing backgrounds (e.g. smaller table). For the non-OCR baselines, we use a mixture of data from the `1/2/3-Cubes` environments to learn a latent representation with a β-VAE [12]. More information on the architectures and hyper-parameters is available in Section G.3.

Experiment Outline We separate our investigation into 3 parts. In the first part, we focus on the design of our OCR and EIT, and how it handles complex interactions between objects. We study this using the GT reward, to concentrate on the representation learning question. When comparing to baselines, we experiment with both the OCR, and a ground truth state representation[1]. To prove that our method indeed handles complex interactions, we shall show that our method with the OCR outperforms baselines with GT state on complex tasks. In the second part, we study compositional generalization. In this case we also use the GT reward, with similar motivation as above. Finally, in the third part we evaluate our method using the Chamfer reward. This part is reported in Section E.2.

D.1 Multi-object Manipulation

We evaluate the different methods with GT rewards on the environments detailed above. Results are presented in Figure 4. We observe that with a single object, all methods succeed, yet the unstructured baselines are less sample efficient. With more than 1 object, the image-based unstructured baseline is not able to learn at all, while the unstructured state-based baseline is significantly outperformed by the structured methods. Our method and SMORL reach similar performance in the state-based setting, SMORL being more sample efficient. This is expected as SMORL essentially learns single

[1]The ground truth state is $s_i = (x_i, y_i)$, $g_i = (x_i^g, y_i^g)$ the (x, y) coordinates of the state and goal of entity i respectively. We detail how this state is input to the networks in Section G.1



(a) 1-Cube (b) 2-Cubes (c) 3-Cubes

(d) Ordered-Push (e) Legend (f) Small-Table

Figure 4: ***Success Rate vs. Environment Timesteps*** *– Values calculated on* 96 *randomly sampled goals. Methods with input type 'State' are presented in dashed lines and learn from GT state observations, otherwise, from images. Our method performs better than or equivalently to the best performing baseline in each category (state/image-based). In the environments requiring object interaction ((d), (f)), our method achieves significantly better performance than SMORL. Notably, our image-based method matches/surpasses state-based SMORL.*

object manipulation, regardless of the number of cubes in the environment. Notably, on 3-Cubes, *our image-based method surpasses the unstructured state-based* method.

In environments that require interaction between objects – Adjacent-Goals, Small-Table and Ordered-Push – our method outperforms SMORL using state input. Moreover, with image inputs our method outperforms SMORL with state inputs (significantly on Ordered-Push, yet marginally on Small-Table, Adjacent-Goals), demonstrating that SMORL is fundamentally limited in performing these more complex tasks.

Method	Success Rate	Success Fraction	Max Obj Dist	Avg Obj Dist	Avg Return
Ours (State)	0.963 ± 0.005	0.982 ± 0.005	0.022 ± 0.002	0.014 ± 0.002	-0.140 ± 0.008
SMORL (State)	0.716 ± 0.006	0.863 ± 0.005	0.063 ± 0.003	0.031 ± 0.001	-0.233 ± 0.004
Ours (Image)	0.710 ± 0.016	0.883 ± 0.005	0.044 ± 0.003	0.027 ± 0.001	-0.202 ± 0.007

Table 1: ***Performance Metrics for*** *Adjacent-Goals – Methods trained on* 3-Cubes *and evaluated on* Adjacent-Goals*. Values calculated on* 400 *random goals per random seed.*

D.2 Compositional Generalization

In this section, we investigate our method's ability to achieve zero-shot compositional generalization. Agents were trained *from images* with our method using GT rewards and require purely image inputs during inference. We present several inference scenarios requiring compositional generalization:

Different Number of Cubes than in Training We train an agent on the 3-Cubes environment and deploy the obtained policy on the N-Cubes environment for $N \in [1, 6]$. Colors are sampled uniformly out of 6 options and are distinct for each cube. Visual results on 6 cubes are presented in Figure 5 (left) and evaluation metrics in Table 4. We see that our agent generalizes to a changing number of objects with some decay in performance as the number of objects grows. Notably, the decay in the average return, plotted in Figure 5 (right), is approximately linear in the number of Cubes.

Cube Sorting We train an agent on the 3-Cubes environment with constant cube colors (red, green, blue). During inference, we provide a goal image containing $X \leq 3$ cubes of different colors and

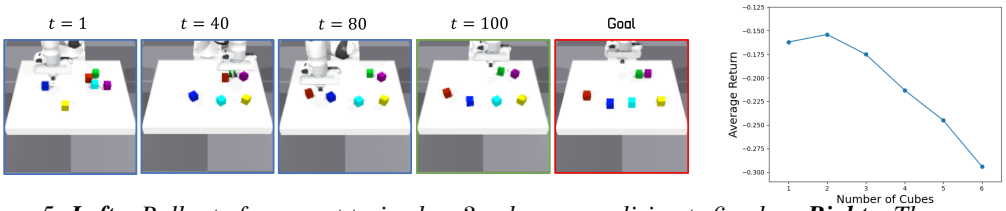


Figure 5: ***Left*** *– Rollout of an agent trained on* 3 *cubes generalizing to* 6 *cubes.* ***Right*** *– The average return of an agent trained on* 3 *cubes vs. the number of cubes in the environment it was deployed in during inference. Values are averaged over* 400 *episodes with randomly initialized goal and initial configurations.*

then deploy the policy on an environment containing $4X$ cubes, 4 of each color. The agent sorts the cubes around each goal cube position with matching color, and is also able to perform the task with cube colors unseen during RL training (but which the DLP model was pretrained on). Visual results on 12 cubes in 3 colors are presented in Figure 6. We find these results exceptional, as they require compositional generalization from both the EIT policy (trained on 3 cubes) and the DLP model (trained on 6 cubes) to significantly more cubes, occupying a large portion of the table's surface.

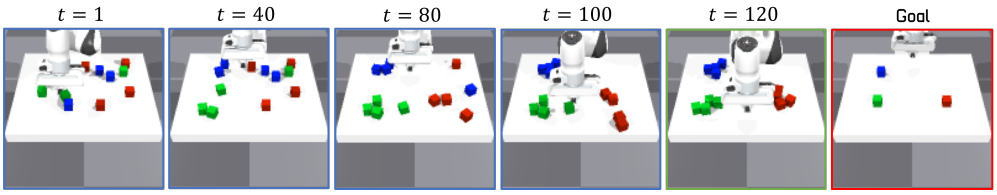


Figure 6: *Rollout of an agent trained on* 3 *cubes, then provided a goal image containing* 3 *different colored cubes and deployed in an environment with* 12 *cubes,* 4 *of each color. The agent sorts the cubes around each goal position with mathcing color.*

Video demonstrations are available on our website.

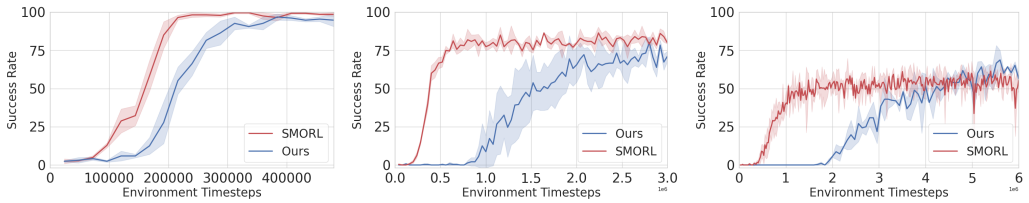
E Chamfer Reward

We desire a reward that captures the task of moving objects to goal configurations. However, because particles in different images are not aligned, and some particles may be occluded or missing, we cannot directly construct a reward based on distances between the particles. Instead, we define a reward from the DLP representations of images as the Generalized Density-Aware Chamfer (GDAC) distance between state and goal particles, which we term *Chamfer reward*. The GDAC distance is defined between two sets $X = \{x_i\}_{i=1}^{N},\ Y = \{y_j\}_{j=1}^{M},\ x_i, y_i \in \mathbb{R}^d$ in the following manner:

$$Dist_{GDAC}\left(X,Y\right)=\frac{1}{\sum_j I(|X_j|>0)}\sum_j\frac{1}{|X_j|+\varepsilon}\sum_{x\in X_j}D_1(x,y_j)+\frac{1}{\sum_i I(|Y_i|>0)}\sum_i\frac{1}{|Y_i|+\varepsilon}\sum_{y\in Y_i}D_1(y,x_i) \tag{1}$$

where $X_j = \left\{x_i | \arg\min_{y_k\in Y}\left(D_2\left(x,y_k\right)\right)=j\right\}$, $Y_i = \left\{y_j | \arg\min_{x_k\in X}\left(D_2\left(y,x_k\right)\right)=i\right\}$ $D_1(x,y)$ and $D_2(x,y)$ are two distance functions between entities. The standard Chamfer distance is obtained by setting $D_1(x,y) = D_2(x,y) = \|x-y\|_2^2$ and substituting $\frac{1}{\sum_j I(|X_j|>0)}\cdot\frac{1}{|X_j|+\varepsilon}$, $\frac{1}{\sum_i I(|Y_i|>0)}\cdot\frac{1}{|Y_i|+\varepsilon}$ with $\frac{1}{|X|}$, $\frac{1}{|Y|}$ respectively.

The Chamfer distance measures the average distance between each entity in X and the closest entity to it in Y and vice versa. The *Density-Aware* Chamfer distance [31] takes into account the fact that multiple entities from one set can be mapped to the same entity in the other set, and re-weights their contribution to the overall distance accordingly. The *Generalized* Density-Aware Chamfer distance, decouples the distance function that is used to match between entities $D_2(x,y)$ and the one used to



(a) 1-Cube (b) 2-Cubes (c) 3-Cubes

Figure 7: ***Success Rate vs. Environment Timesteps (Image-Based Rewards)*** – *Values calculated based on* 96 *randomly sampled goals.*

calculate the distance between them $D_1(x, y)$. Decoupling these two allows using entity-identifying attributes for matching while calculating the actual distances between matching entities based on localization features. For example, we can use the DLP visual features z_f to match between objects in the current and goal images, and then measure their distance using the (x, y) coordinate attributes z_k.

E.1 Focused Chamfer Reward

In many robotic object manipulation settings, we do not care about the robot in our goal specification as long as the objects reach the desired configuration. In order to consider only a subset of the entities for the image-based reward (e.g. particles corresponding to objects and not the agent), we train a simple multi-layer perceptron (MLP) binary classifier on the latent visual features of the DLP representation, differentiating objects of interest from the rest of the particles. We train this classifier on annotated particles extracted from 20 images of the environment. Annotation required 5 minutes of our time and training the classifier itself took just a few seconds. We then filter out particles based on the classifier output before inputting them to the Chamfer reward. We emphasize that this supervision is only required for training the classifier which is used for the image-based reward exclusively during RL training.

E.2 Training with the Chamfer Reward

We compare our method to SMORL, trained entirely from images. Results on N-Cubes for $N \in \{1, 2, 3\}$ are presented in Figure 7 and Table 2. Training our method with the image-based reward obtains lower success rates compared to training with the GT reward. While this is expected, we believe the large differences are due to noise originated in the DLP representation and occlusions, which make the reward signal less consistent and harder to learn from. This is especially hard with increasing number of objects, as the chances of at least one object being occluded are very high. This is highlighted by the drop in performance from 1 to 2 cubes, compared to the GT reward. Image-based reward calculation for single object manipulation, as in SMORL, is slightly more consistent as occlusions in a single view will not affect the overall reward as much. Adding more viewpoints for the reward calculation might improve these results without increasing inference complexity. We see that in the 3-Cubes environment, our method surpasses SMORL, although SMORL's reward is based on a single object regardless of the number of objects in the environment. This could be as result of object-object interactions being more significant in this case.

Method	Success Rate	Success Fraction	Max Obj Dist	Avg Obj Dist	Avg Return
Ours	0.765 ± 0.025	0.875 ± 0.015	0.037 ± 0.002	0.026 ± 0.001	-0.210 ± 0.009
SMORL	0.838 ± 0.016	0.911 ± 0.008	0.038 ± 0.004	0.025 ± 0.002	-0.320 ± 0.007
Ours	0.580 ± 0.093	0.822 ± 0.052	0.063 ± 0.008	0.035 ± 0.005	-0.251 ± 0.022
SMORL	0.509 ± 0.044	0.794 ± 0.024	0.092 ± 0.006	0.047 ± 0.004	-0.451 ± 0.031

Table 2: **Performance Metrics: Image-Based Rewards** Methods trained and evaluated on the 2-Cubes (top) and 3-Cubes (bottom) environments. Values calculated on 400 random goals per random seed.

F Additional Results

F.1 Multi-Object Manipulation

Performance metrics for the 2-Cubes and 3-Cubes are presented in Table 3

Method	Success Rate	Success Fraction	Max Obj Dist	Avg Obj Dist	Avg Return
Ours (State)	0.991 ± 0.004	0.995 ± 0.003	0.014 ± 0.001	0.010 ± 0.001	-0.129 ± 0.006
SMORL (State)	0.980 ± 0.006	0.989 ± 0.005	0.014 ± 0.002	0.009 ± 0.002	-0.142 ± 0.016
Ours	0.968 ± 0.019	0.983 ± 0.009	0.020 ± 0.002	0.015 ± 0.001	-0.150 ± 0.008
Ours (State)	0.978 ± 0.006	0.991 ± 0.002	0.016 ± 0.001	0.010 ± 0.001	-0.124 ± 0.007
SMORL (State)	0.932 ± 0.022	0.974 ± 0.009	0.028 ± 0.005	0.015 ± 0.002	-0.201 ± 0.011
Ours	0.919 ± 0.008	0.969 ± 0.004	0.026 ± 0.002	0.016 ± 0.001	-0.157 ± 0.007

Table 3: **Performance Metrics: GT Reward** Methods trained and evaluated on the 2-Cubes (top) and 3-Cubes (bottom) environments. Values calculated on 400 random goals per random seed.

F.2 Generalization

Performance metrics for the compositional generalization to different numbers of objects of an agent trained on the 3-Cubes environment are presented in Table 4.

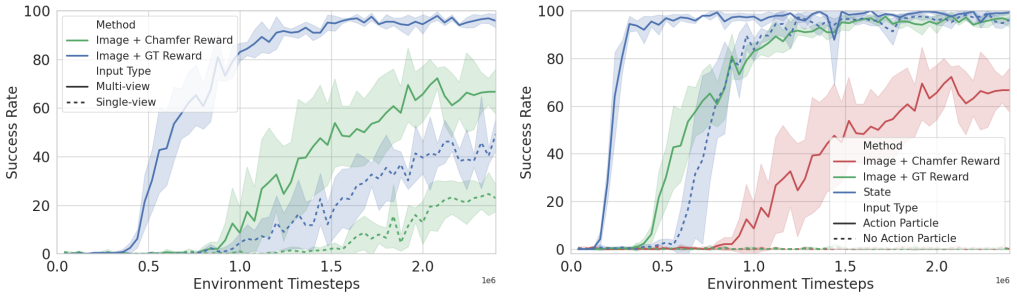
An additional scenario we consider is providing the agent a goal image which contains some cube colors that are not present in the environment. We term these cubes *distracters*. The agent is able to disregard the distracters in the goal image while successfully manipulating the other cubes to their respective goal locations.
While we designed our algorithm to facilitate compositional generalization, it has demonstrated some ability to generalize to cube colors it has not seen during training. Videos demonstrating these capabilities are available on our website.

Number of Cubes	Success Rate	Success Fraction	Max Obj Dist	Avg Obj Dist	Avg Return
1	0.973	0.973	0.016	0.016	-0.162
2	0.963	0.981	0.023	0.017	-0.154
3	**0.838**	**0.942**	**0.034**	**0.02**	**-0.175**
4	0.723	0.912	0.051	0.027	-0.213
5	0.57	0.876	0.068	0.031	-0.245
6	0.398	0.826	0.09	0.036	-0.294

Table 4: ***Performance Metrics for Different Number of Cubes than in Training*** *– Our method's performance on different numbers of cubes in the N-cubes environment, trained on the 3-cubes environment (results in **bold**) with cubes of 6 different colors. Values are averaged over 400 episodes with randomly initialized goal and initial configurations.*

F.3 Ablation Study

We explore how aspects we found to be key to the success of our proposed method effect sample efficiency. Figure 8a compares our method with multi-view vs. single-view image inputs. We find that both with GT and image-based reward, multi-view inputs substantially improve sample efficiency. We believe that the connections formed between particles from different views in the transformer blocks makes it easier for the agent to learn the correlations between actions defined in 3D space and latent attributes defined in 2D pixel space. In addition, multiple viewpoints decrease the degree of partial observability. Figure 8b compares treating the action as a separate input entity to the Q-function transformer blocks vs. concatenating the action to the output of the final transformer block, before the output MLP. We find that the action having equal status as the state and goal entities is crucial to the performance of our method. Without it, our experiments exhibit decreased sample efficiency in state observations and failure to learn in the given environment timestep budget with image observations.



(a) Multiview (b) Action Entity

Figure 8: ***Success Rate vs. Environment Timesteps (Ablations)*** – *Values calculated based on* 96 *randomly sampled goals.*

G Implementation Details and Hyper-parameters

In this section, we provide extensive implementation details in addition to the open-source code that will be made available.

G.1 Environment

We implement our environments with IsaacsGym [17], by adapting code from IsaacGymEnvs[2] and OSCAR[3].

Ground-truth State Denote $s_i = (x_i, y_i)$, $g_i = (x_i^g, y_i^g)$ the xy coordinates of the state and goal of entity i respectively. The input to the networks in the structured methods are two sets of vectors $\{v_i\}_{i=1}^N$, $v_i = [s_i, \text{one-hot}\,(i|N)] \in \mathbb{R}^{2+N}$, $\{u_i\}_{i=1}^N$, $u_i = [g_i, \text{one-hot}\,(i|N)] \in \mathbb{R}^{2+N}$, $[\cdot]$ denoting concatenation, where the one-hot vectors serve as entity-identifying features. In the unstructured case, the input is $[s_1, s_2, ..., s_N, g_1, g_2, ..., g_N]$.

Ground-truth Reward The reward calculated from the ground-truth state of the system, which we refer to as the ground-truth (GT) reward, is the mean negative L_2 distance between each cube and its desired goal position on the table:

$$r_t = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{L}\left\|g_i^d - g_i^a\right\|_2, \tag{2}$$

where g_i^d and g_i^a denote the desired and achieved goal for object i respectively, N the number of objects, r_t the immediate reward at timestep t and L a normalization constant for the reward corresponding to the dimensions of the table.

Image-Based Reward The reward calculated from the DLP OCR for our method is the negative GDAC distance (see Eq. 1) between state and goal sets of particles, averaged over viewpoints:

$$r_t = -\frac{1}{K}\sum_{k=1}^{K} Dist_{GDAC}\left(\{p_m^k\}_{m=1}^M, \{q_m^k\}_{m=1}^M\right), \tag{3}$$

where we use $D_1(x,y) = \left\|z_p^x - z_p^y\right\|_1$ and $D_2(x,y) = \left\|z_f^x - z_f^y\right\|_2$ in the GDAC distance, $z_p^{(\cdot)}$, $z_f^{(\cdot)}$ denoting DLP latent attribute z_p, z_f of particle $(\cdot)$ respectively. We filter out particles that do not correspond to cubes (see section E.1) for the distance calculation. When a particle has no match (i.e. $\min_y \left\|z_f^x - z_f^y\right\|_2 > C$), a negative bonus is added to the reward to avoid "reward hacking" by pushing blocks off the table or occluding them on purpose.

Evaluation Metrics We evaluate the performance of the different methods based on the following:

[2] `https://github.com/NVIDIA-Omniverse/IsaacGymEnvs`

[3] `https://github.com/NVlabs/oscar`

Success: $\mathbb{I}\left(\sum_{i=1}^{N}\mathbb{I}\left(\left\|g_i^d - g_i^a\right\|_2 < R\right) = N\right)$, all N objects are at a threshold distance from their desired goal. R denotes the success threshold distance and is slightly smaller than the effective radius of a cube. $\mathbb{I}$ denotes the indicator function. This metric most closely captures task success, but does not capture intermediate success or timestep efficiency.

Success Fraction: $\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\left\|g_i^d - g_i^a\right\|_2 < R\right)$, fraction of objects that reach individual success.

Maximum Object Distance: $\max_i\left\{\left\|g_i^d - g_i^a\right\|_2\right\}$, largest distance of an object from its desired goal.

Average Object Distance: $\frac{1}{N}\sum_{i=1}^{N}\left\|g_i^d - g_i^a\right\|_2$, average distance of objects from their desired goal.

Average Return: $\frac{1}{T}\sum_{t=1}^{T} r_t$, the immediate GT reward averaged across timesteps, where T is the evaluation episode length. A high average return means that the agent solved the task quickly.

G.2 Reinforcement Learning

We implement our RL algorithm with code adapted from `stable-baselines3` [24]. Specifically, we use TD3 [7] with HER [1]. We use ε-greedy and Gaussian action noise for exploration, that decays to half its initial value with training progress, similar to Zhou et al. [36]. We use Adam for neural network optimization. Related hyper-parameters can be found in Table 5 and Table 6.

Learning Rate	5e-4
Batch Size	512
γ	0.98
τ	0.05
# Episodes Collected per Training Loop	16
Update-to-Data Ratio	0.5
HER Ratio	0.8
Exploration Action Noise σ	0.2
Exploration ε	0.3

Table 5: General hyper-parameters used for RL training.

Number of Cubes	1	2	3
Episode Horizon	30	50	100
Replay Buffer Size	100000	100000	200000

Table 6: Environment specific hyper-parameters used for RL training.

For the Entity Interaction Transformer (EIT) we adapted components from DDLP's Particle Interaction Transformer (PINT, Daniel and Tamar [5]), which is based on a transformer decoder architecture [14] and utilizing the open-source minGPT [14] code base. Related hyper-parameters can be found in Table 7.

Attention Dimension	64
Attention Heads	8
MLP Hidden Dimension	256
MLP Number of Layers	3

Table 7: Hyper-parameters for the EIT architecture.

Attention Masking: The DLP model extracts a fixed number of particles, which often include particles that do not represent objects in the image. These particles are assigned low transparency (z_t) values by the DLP model as to not affect the reconstruction quality. We disregard these particles in our policy and Q-function by directly masking the attention entries related to them. We found that this slightly improves sample efficiency but is not crucial to performance as the EIT is able to learn to disregard these particles by assigning them very low attention values.

Policies for the unstructured baselines have 5 layer MLPs with hidden dimension 256.

G.3 Pre-trained Image Representations

In this section, we detail the various *unsupervised* pre-trained image representation methods used in this work. We begin with the non-object-centric baselines, i.e., methods that given an image $I \in \mathbb{R}^{H \times W \times C}$, encode a single-vector representation $z \in \mathbb{R}^D$, where D is the latent dimension, of the entire input image. Then, we describe the object-centric representation (OCR) method that provides a structured latent representation $z \in \mathbb{R}^{K \times d}$ of a given image I, where K is the number of entities in the scene, each described by latent features of dimension d.

Data: We collect $600,000$ images from 2 viewpoints by interacting with the environment using a random policy for $300,000$ timesteps. For all methods, we use RGB images at a resolution of 128×128, i.e., $I \in \mathbb{R}^{128 \times 128 \times 3}$.

Variational Autoencoder (VAE): We train a β-VAE [12] with a latent bottleneck of size $D = 256$, i.e., each image I is encoded as $z \in \mathbb{R}^{256}$. We adopt a similar autoencoder architecture as Rombach et al. [25] based on the open-source implementation[4] and add a 2-layer MLP with 512 hidden units after the encoder and before the decoder to ensure the latent representation is of dimension 256. We use $\beta = 1e-10$, a batch size of 16 and an initial learning rate of $2e-4$ which is gradually decayed with a linear schedule. The model is trained for 40 epochs with a perceptual reconstruction loss and L_1 pixel-wise loss, similarly to Rombach et al. [25], and we keep the default values for the rest of the hyper-parameters.

Deep Latent Particles (DLP): We train a DLPv2 [5] using the publicly available code base[5] as our unsupervised OCR model. Recall that DLP provides a disentangled latent space structured as a set of foreground particles $z = \{(z_p, z_s, z_d, z_t, z_f)_i\}_{i=0}^{K-1} \in \mathbb{R}^{K \times (6+m)}$, where K is the number of particles, $z_p \in \mathbb{R}^2$ is the position of the particle as (x, y) coordinates in Euclidean space, $z_s \in \mathbb{R}^2$ is a scale attribute, $z_d \in \mathbb{R}$ is a depth attribute, $z_t \in \mathbb{R}$ is a transparency attribute and $z_f \in \mathbb{R}^m$ are the visual features that encodes the visual appearance in the particle region, where m is the dimension of learned visual features. A background particle $z_{\text{bg}} \in \mathbb{R}^{m_{\text{bg}}}$ describes the visual features of the background. Figure 9 illustrates an example of object-centric decomposition for a single image using a DLP model pre-trained on our data. We keep the default recommended hyper-parameters and report the data-specific hyper-parameters in Table 8. Note that our DLP model represents an image I by a total of $m_{\text{bg}} + K \times (6 + m) = 1 + 24 * (6 + 4) = 241$ latent features.

Batch Size	64
Posterior KP K	24
Prior KP Proposals L	32
Reconstruction Loss	MSE
β_{KL}	0.1
Prior Patch Size	16
Glimpse Size S	32
Feature Dim m	4
Background Feature Dim m_{bg}	1
Epochs	60

Table 8: Hyper-parameters used for the Deep Latent Particles (DLP) object-centric model.

G.4 SMORL Reimplementation

We re-implement SMORL [34] based on the official implementation[6] using the same code-base as we used for the EIT, for the SMORL attention architecture. SMORL specific hyper-parameters are detailed in Table 9. We extend SMORL to multiple views, which includes modifications to several aspects of the algorithm:

Attention Architecture: We extend SMORL's attention policy by adding a goal-conditioned and goal-unconditional attention block for the additional view. The outputs of attention layers from both

[4] `https://github.com/CompVis/latent-diffusion`

[5] `https://github.com/taldatech/ddlp`

[6] `https://github.com/martius-lab/SMORL`

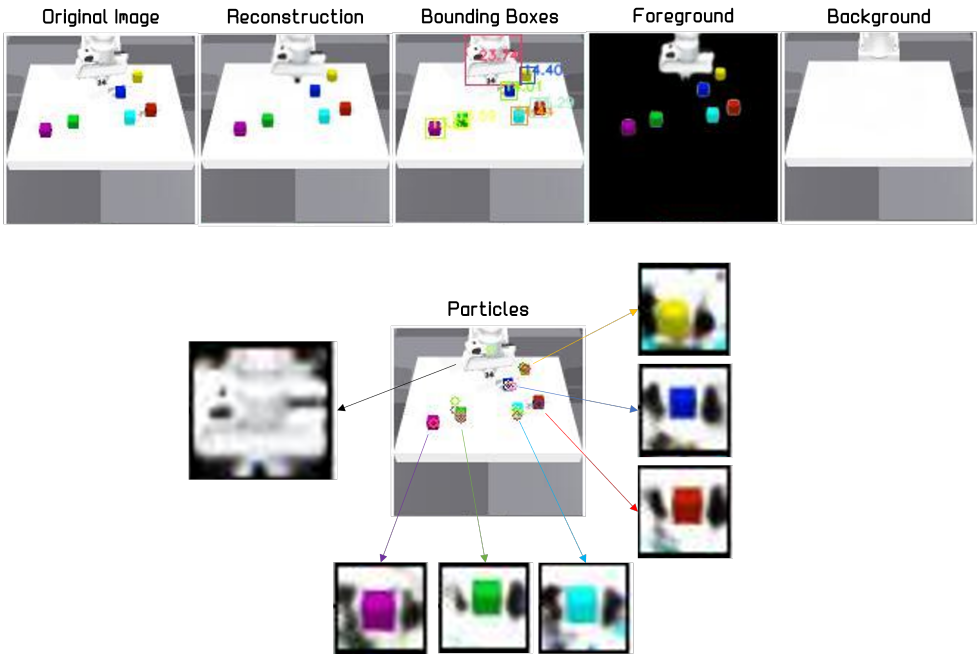


Figure 9: *Object-centric Decomposition with DLP. DLP decomposes a single image into latent particles, each characterized by attributes including position (keypoints in the images), scale (bounding boxes), and visual appearance features around the keypoint (displayed as decoded glimpses from these features).*

views are concatenated and fed to an MLP, as in the single-view version. An outline of SMORL's single-view attention-based architecture is described in Figure 10.

Selecting a Single Goal: SMORL decomposes the multi-object goal-conditioned task to single objects by selecting a single goal at a time, and rewarding the agent with respect to this sub-goal alone. Working with multiple views requires selecting a goal particle corresponding to the same object from both viewpoints, which requires explicit matching. We do this by selecting a goal particle from one viewpoint and choosing the closest matching particle from the second viewpoint based on the L_2 distance in latent attribute z_f.

Reward: The image-based reward calculated from the DLP OCR for SMORL is the negative L_2 distance in attribute z_p between the goal particle to the closest matching state particle based on attribute z_f, averaged over viewpoints:

$$r_t = -\frac{1}{K}\sum_{k=1}^{K}\left\| z_p^{g^k} - z_p^{s^k_{m_k}} \right\|_2, \quad m_k = \arg\min_m \left\| z_f^{g^k} - z_f^{s^k_m} \right\|_2, \tag{4}$$

g^k denoting the goal particle from view k and s^k_m denoting particle m from view k. When there is no match for the goal particle in viewpoint k (i.e. $\min_m \left\| z_f^{g^k} - z_f^{s^k_m} \right\|_2 > C$), the minimal reward is given for that view. Note that this reward is a special case of the Chamfer reward we define in this work.

H Attention in RL Policies - Comparison to Previous Work

In this section, we compare our use of attention to two previous approaches, Zhou et al. [36] which is state-based and SMORL [34] which is image-based.
Zhou et al. [36] also propose a transformer-based policy. They define an entity as the concatenation

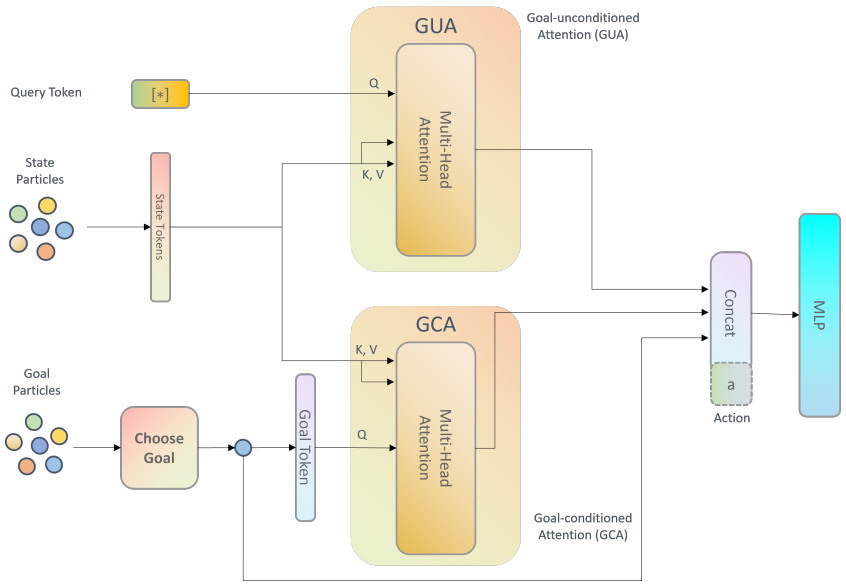


Figure 10: ***Outline of SMORL's Attention Architecture*** *The policy is conditioned on the goal by choosing a single goal particle and feeding it to a cross-attention block between the goal particle and the state particles. In parallel, cross-attention between a learned particle and the state particles is performed to extract features from the state that are not goal-dependant. The outputs of the two attention layers are then concatenated to the original goal particle and fed to an MLP to produce the action. For the Q-function, the input action is additionally concatenated to the output of the attention to produce the value.*

Attention Dimension	64
Unconditional Attention Heads	8
Goal-conditioned Attention Heads	8
MLP Hidden Dimension	256
MLP Layers	4
Scripted Meta-policy Steps	15

Table 9: SMORL hyper-parameters.

of each object's state, goal, and the state of the agent (and the action when we consider the input to the Q-function). This requires explicitly matching between entities in state and goal as well as identifying the agent, which is trivial when working with GT state observations. When learning from OCRs of images, this is not at all trivial and is not always possible due to lack of existence of a one-to-one match or occlusion, which also limits the use of multiple viewpoints. We tackle this in our EIT by using a cross-attention block for goal-conditioning. Figure 11 describes the differences between their definition of an input entity to ours.

SMORL uses an OCR of images and does not require matching between entities in the single-view case. The goal-conditioned attention in their architecture (see Figure 10) matches a single goal particle to the relevant state particle via cross-attention. Different from us, the attention mechanism is used only to extract sub-goal specific entities from the set of state entities, and does not explicitly model relationships between the different entities in the state. We do this explicitly by incorporating self-attention transformer blocks in our architecture.

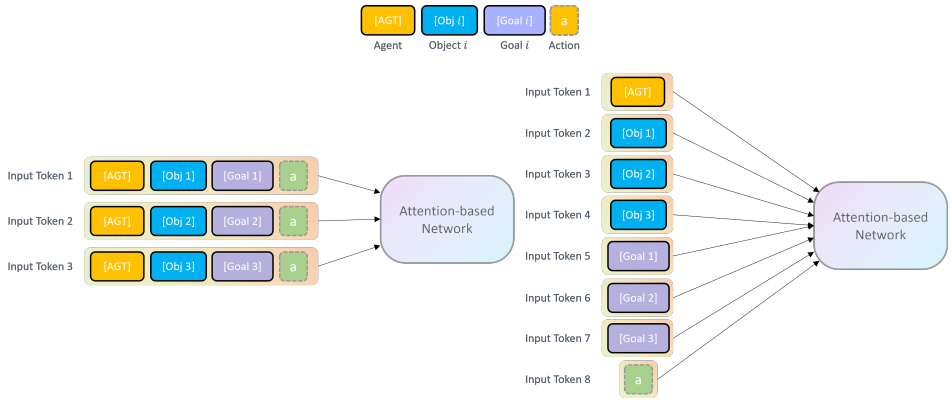


Figure 11: ***Entity Definition Comparison*** *Left – description of an input token defined by Zhou et al. [36], where each token is a concatenation of the object and corresponding goal as well as global entities such as the agent and action; Right – Our definition of input tokens, where each entity is treated as a separate token.*