Object-centric architectures enable efficient causal representation learning

Amin Mansouri
Mila, Quebec AI Institute

Jason Hartford
Mila, Quebec AI Institute

Yan Zhang
Samsung - SAIT AI Lab Montréal

Yoshua Bengio
Mila, Quebec AI Institute

Abstract

Causal representation learning has showed a variety of settings in which we can disentangle latent variables with identifiability guarantees (up to some reasonable equivalence class). Common to all of these approaches is the assumption that (1) the latent variables are represented as d-dimensional vectors, and (2) that the observations are the output of some injective generative function of these latent variables. While these assumptions appear benign, we show that when the observations are of multiple objects, the generative function is no longer injective and disentanglement fails in practice. We can address this failure by combining recent developments in object-centric learning and causal representation learning. By modifying the Slot Attention architecture (Locatello et al., 2020b), we develop an object-centric architecture that leverages weak supervision from sparse perturbations to disentangle each object's properties. This approach is more data-efficient in the sense that it requires significantly fewer perturbations than a comparable approach that encodes to a Euclidean space and we show that this approach successfully disentangles the properties of a set of objects in a series of simple image-based disentanglement experiments.

1 Introduction

Consider the image in Figure 1 (left). We can clearly see four different colored balls, each at a different position. But asking, "Which is the first shape? And which is the second?" does not have a clear answer: the image just depicts an unordered set of objects. This observation seems trivial, but it implies that there exist permutations of the objects which leave the image unchanged. For example, we could swap the positions of the two blue balls without changing a single pixel in the image.

In causal representation learning, the standard assumption is that our observations x are "rendered" by some generative function $g(\cdot)$ that maps the latent properties of the image z to pixel space (i.e. $x = g(z)$); the goal is to *disentangle* the image by finding an "inverse" map that recovers z from x up to some irrelevant transformation. The only constraint on $g(\cdot)$ that is assumed by all recent papers (for example Hyvarinen and Morioka, 2016, 2017; Locatello et al., 2020a; Khemakhem et al., 2020a,b; Lachapelle et al., 2022; Ahuja et al., 2022a,b, 2023), is that $g(\cdot)$ is injective[1], such that $g(z_1) = g(z_2)$ implies that $z_1 = z_2$. But notice that if we represent the latents z as some d-dimensional vectors in Euclidean space, then whenever we observe objects like those shown in Figure 1, this injectivity assumption fails: symmetries in the objects' pixel representation imply that

[1]Some papers place stronger constraints on $g(\cdot)$, such as linearity Hyvärinen and Oja, 2000; Squires et al., 2023, sparsity Moran et al., 2022; Zheng et al., 2022, or constraints on g's Jacobian Gresele et al., 2021; Brady et al., 2023 but injectivity is the weakest assumption common to all approaches.

Accepted to the NeurIPS 2023 Workshop on Causal Representation Learning.

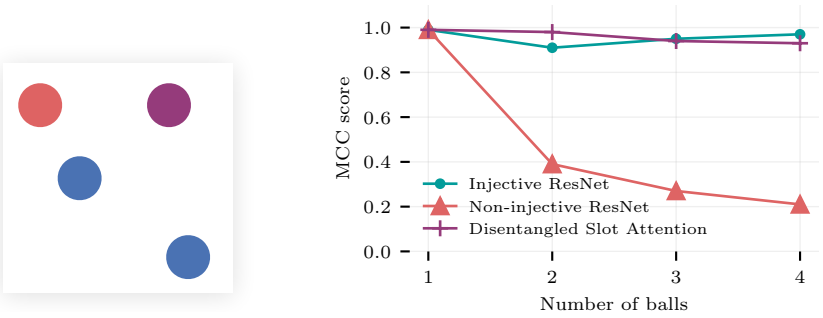

Figure 1: *(Left)* An example image of simple objects. *(Right)* Mean correlation coefficient (MCC) score which measures the correlation between inferred latent variables and their associated ground truth values. Ahuja et al. (2022b)'s approach achieves almost perfect MCC scores (i.e. a score ≈ 1) when the ball color is used to make the generative function injective ("Injective ResNet"), but achieves an MCC score of at most $\frac{1}{k}$ where k is the number of objects when colors are selected randomly ("Non-injective ResNet"). We show that it is possible to recover the injective performance by disentangling object-centric representations ("Disentangled Slot Attention").

there exist non-trivial permutation matrices Π, such that $g(z) = g(\Pi z)$. This is not just a theoretical inconvenience: Figure 1 (right) shows that when the identity of the balls is not distinguishable, the disentanglement performance of a recent approach from Ahuja et al. (2022b) is upper-bounded by $1/k$ where k is the number of balls.

In parallel to this line of work, there has been significant progress in the object-centric learning literature (e.g. van Steenkiste et al., 2018a; Goyal et al., 2019; Locatello et al., 2020b; Goyal et al., 2020; Lin et al., 2020; Zhang et al., 2023) that has developed a suite of architectures that allow us to separate observations into sets of object representations. Two recent papers (Brady et al., 2023; Lachapelle et al., 2023) showed that the additive decoders used in these architectures give rise to provable object-wise disentanglement, but they did not address the task of disentangling the objects' associated properties. In this paper we show that by leveraging object-centric architectures, *we effectively reduce the multi-object problem to a set of single-object disentanglement problems* which not only addresses injectivity failures, but also results in a significant reduction in the number of perturbations we need to observe to disentangle properties using Ahuja et al. (2022b)'s approach. We illustrate these results by developing a property disentanglement algorithm that combines Zhang et al. (2023)'s SA-MESH object-centric architecture with Ahuja et al. (2022b)'s approach to disentanglement and show that our approach is very effective at disentangling the properties of objects on both 2D and 3D synthetic benchmarks.

In summary, we make the following contributions:

- We highlight two problems that arise from objects which violate standard assumptions used to identify latent variables (Section 2).
- We show that these problems can be addressed by leveraging object-centric architectures, and that using object-centric architectures also enables us to use a factor of k fewer perturbations to disentangle properties, where k is the number of objects (Section 3).
- We implement the first object-centric disentanglement approach that disentangles object properties with identifiability guarantees (Section 4).
- We achieve strong empirical results on both 2D and 3D synthetic benchmarks (Section 5).

We will build on a recent line of work that leverages paired samples from sparse perturbations to identify latent variables (Locatello et al., 2020a; Brehmer et al., 2022; Ahuja et al., 2022b). Our approach generalizes Ahuja et al. (2022b) to address the non-injectivity induced by objects, so we will briefly review their main results. Ahuja et al. assume that they have access to paired samples, (x, x') where $x = g(z)$, $x' = g(z')$, and z_i is perturbed by a set of sparse offsets $\Delta = \{\delta_1, \dots, \delta_k\}$, such that $z'_i = z_i + \delta_i$ for all $i \in \{1, \dots, k\}$. They show that if $g(\cdot)$ is an injective analytic function from $\mathbb{R}^d \to \mathcal{X}$, every $\delta \in \Delta$ is 1-sparse, and at least d linearly independent offsets are observed, then an encoder, f that minimizes the following objective recovers the true z up to permutations, scaling

and an offset (Ahuja et al., 2022b, Theorem 1),

$$\hat{f} \in \arg\min_{f'} E_{x,x',\delta}\left[\left(f'(x) + \delta - f'(x')\right)^2\right] \quad \Rightarrow \quad \hat{f}(x) = \hat{z} = \Pi\Lambda z + c \tag{1}$$

where Π is a permutation matrix, Λ is an invertible diagonal matrix and c is an offset.

2 Objects result in non-identifiability

We begin by formally characterizing the challenges that arise when images contain multiple objects.

Data generating process. We assume that a set $Z := \{z_i\}_{i=1}^k$ of k objects is drawn from some joint distribution, $\mathbb{P}_Z$. In order to compare set and vector representations, let $\mathbf{vec}_\pi(Z)$ denote a flattened vector representation of Z ordered according to some permutation $\pi \in \text{Sym}(k)$, the symmetric group of permutations of k objects; when π is omitted, $\mathbf{vec}(Z)$ simply refers to an arbitrary default ordering (i.e. the identity element of the group). Each object is described by a d-dimensional vector of properties $z_i \in \mathbb{R}^d$, and hence $\mathbf{vec}(Z) \in \mathbb{R}^{kd}$. We say objects have *shared properties* if the coordinates of z_i have consistent meaning across objects, and restrict our focus to such properties.

The non-injectivity problem. We observe images x which are generated via a generative function $\mathfrak{g}(\cdot)$ that renders a set of object properties into a scene in pixel space, such that $x = \mathfrak{g}(Z)$. While $\mathfrak{g}(\cdot)$ is a set function, we can define an equivalent vector generative function, g, which, by definition, produces the same output as $\mathfrak{g}(Z)$; i.e. for all $\pi \in \text{Sym}(k)$, $g(\mathbf{vec}_\pi(Z)) = \mathfrak{g}(Z)$. This generative function g taking vectors as input is consistent with standard disentanglement assumptions except that it is not injective:

Proposition 1. *If $g(\boldsymbol{vec}_\pi(Z)) = \mathfrak{g}(Z)$ for all $\pi \in Sym(k)$, then $g(\cdot)$ is not injective.*

This proposition (Proof given in appendix A.1) simply states that if images are composed of *sets* of objects, then if we model the generative function as a map from a Euclidean space, this map will not be injective by construction. With the exception of Lachapelle et al. (2023), all of the causal representation learning papers cited in appendix B (related work) assume the generative function g is injective. To see why injectivity is necessary in general, consider an image with two objects. If the two objects are identical, then there are two disentangled solutions corresponding to the two permutations, so it is not possible to identify a unique solution.

The object identity problem. When applying sparse perturbations on Z, we are effectively perturbing one coordinate of one object. However, how can we know which object of the multiple possible objects in Z we have perturbed? In the case of injective mappings, this is simple: since there is a consistent ordering for them, we know that a coordinate in $\mathbf{vec}(Z)$ corresponds to the same object before and after the perturbation.

This is no longer the case in our setting. Since the objects are part of a set, we cannot rely on their ordering. In fact, we know these ordering changes *must* be present due to the *responsibility problem*:

Proposition 2 (Zhang et al. (2020); Hayes et al. (2023))**.** *If the data is generated according to the data generating process described above with $g(\boldsymbol{vec}_\pi(Z)) := \mathfrak{g}(Z)$ and $k > 1$, then $f(\cdot)$ is discontinuous.*

See appendix A.2 and Hayes et al. (2023) for a general proof. These discontinuities manifest themselves as changes in permutation from one $\mathbf{vec}_{\pi_1}(Z)$ to another $\mathbf{vec}_{\pi_2 \neq \pi_1}(Z)$. In disentanglement approaches that leverage paired samples (e.g. Ahuja et al., 2022b; Brehmer et al., 2022), continuity enables the learning algorithm to implicitly rely on the object identities to stay consistent. Without continuity, one cannot rely on the fact that $\mathbf{vec}(Z)$ and $\mathbf{vec}(Z) + \delta$ should be the same up to the perturbation vector δ, because the perturbation may result in a discontinuous change of $\mathbf{vec}(Z) + \delta$ when an observation is encoded back to latent space. As a consequence, we lose track of which object we have perturbed in the first place, so naïve use of existing disentanglement methods fails. Another challenge is that the encoder f (Equation 1) has to map observations to $\mathbf{vec}(Z)$ in a *discontinuous* way, which is traditionally difficult to model with standard machine learning techniques.

In summary, the unordered nature of objects in Z results in non-injectivity, losing track of object identities, and the need for learning discontinuous functions. These all contribute to the non-identifiability of traditional disentanglement methods in theory and practice.

3 Object-centric causal representation learning

A natural solution to this problem is to recognize that the latent representations of multi-object images are sets and should be treated as such by our encoders and decoders in order to enforce invariance among these permutations. Both Brady et al. (2023) and Lachapelle et al. (2023) showed that architectures that enforce an appropriate object-wise decomposition in their decoders *provably* disentangle images into object-wise blocks of latent variables. These results do not disentangle the properties of objects, but they solve an important precursor: the assumption that there exists an object-wise decomposition of the generative function is sufficient to partition the latents into objects. Like these two papers, we will assume that natural images can be decomposed into objects, each of which occupies a disjoint set of pixels. When this is the case, we say that an image is *object-separable*, i.e., there exists an **object-wise** partition P and a bijection $\sigma : \mathcal{I}_P \to \mathcal{I}_Z$ that associates each subset of pixels in P with a particular element of the set of latents, z_i (See appendix C for more details).

Disentangling properties with object-centric encoding. In section 2 we showed that the assumptions underlying sparse perturbation-based disentanglement approach are violated in multi-object scenes. But, the results from Brady et al. (2023) and Lachapelle et al. (2023) show that the objects can be separated into disjoint (but entangled) sets of latent variables. This suggests a natural approach to disentangling properties in multi-object scenes:

- we can reduce the multi-object disentanglement problem to a single-object problem with an object-wise partition of the image. Within each patch of object pixels injectivity holds, and so we no longer have multiple solutions at a patch level. This partition is identifiable and we can use an object-centric architecture to learn the object-wise partition. We require that this object-centric architecture can handle the responsibility problem.
- we leverage Ahuja et al. (2022b)'s approach to using weak supervision to disentangle the properties of each object individually. Since we assume that properties between objects are shared, this requires a factor of k fewer perturbations in the perturbation set Δ.
- we address the object identity problem where we lose track of object identities after perturbations through an explicit matching procedure that re-identifies the object being perturbed.

This approach not only addresses the challenges outlined in Section 2, it also significantly reduces the number of perturbations that we have to apply in order to disentangle shared properties (See the following theorem and its proof in appendix A.3).

Theorem 1 (informal)**.** *If a data generating process outputs observations with k objects that have shared properties, then an object-centric architecture of the form $F(x) := \{\mathfrak{f}(x^{(i)})\}_{x^{(i)} \in P}$ where P is an object-wise partition and $\mathbf{f} : \mathcal{X} \to \mathbb{R}^d$ will disentangle in k times fewer perturbations than an encoder of the form $f : \mathcal{X} \to \mathbb{R}^{kd}$.*

4 Method

Object-wise partitions There exist a number of ways to decompose an image into objects, but for our purposes, pixel segmentation-based approaches (Greff et al., 2019; Locatello et al., 2020b; Zhang et al., 2023) let us directly adapt existing disentanglement techniques to work with object-centric encoders. A pixel segmentation encoder $\hat{f}$ maps from images x to a set of slot vectors $\{s_1, \ldots, s_k\}$, each of which depends on a subset of the pixels $x^{(i)} \in P$. Images are then reconstructed using a slot decoder $\hat{g}$ that maps from the set of slot representation back to pixel space. We use Zhang et al.'s SA-MESH modification of the original Locatello et al. slot attention architecture, which adds an entropy regularization term that encourages the subsets of pixels $x^{(i)}$ to be disjoint (for details on the architectures, see Appendix D). Importantly for us, Zhang et al. (2023) is *exclusively multiset-equivariant* (Zhang et al., 2022), which allows it to model discontinuous functions, thus handling the responsibility problem.

Slot attention is usually trained with a reconstruction loss from relatively high-dimensional per-object slot representations, $s_i \in \mathbb{R}^D$, but for the images that we work with, we want a relatively low dimensional latent description (in the simplest case, just the two dimensions representing the (x, y) coordinates of each object). To disentangle these high-dimensional slot representations, we simply add a projection head, $\hat{p} : s_i \to \hat{z}_i$, that is trained by a latent space loss.

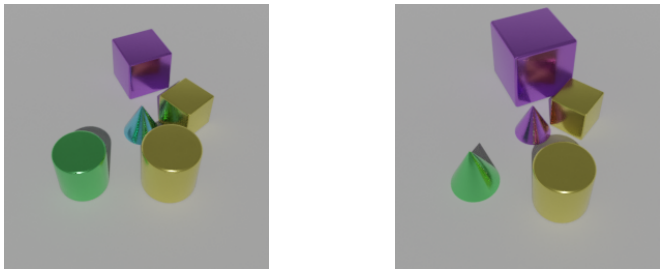

Figure 2: *(Left)* An example image before any perturbation. *(Right)* Possible perturbations in the synthetic 3D dataset, i.e. change in size, orientation, color, position, and shape.

Disentanglement via weak supervision with matching Ahuja et al. use a disentanglement loss (section 1) that requires the latent representations of a pair of images differ by δ, such that $\hat{f}(x) + \delta = \hat{f}(x')$. When using a slot attention architecture, we introduce a *matching step* to infer the object to which the offset δ was applied. With 1-sparse δ vectors, the matching step reduces to a simple minimization over a cost matrix that measures $\|\hat{z}(x'^{(j)}) - (\hat{z}(x^{(i)}) + \delta)\|^2$ for all pairs of slots i, j. In Appendix F, we provide a more general matching procedure that applies to settings with dense offsets δ. We jointly optimize the following reconstruction and disentanglement loss,

$$\hat{f}, \hat{g}, \hat{p} \in \arg\min_{f,g,p} E_x[\|x - g(f(x))\|^2] + E_{x,x',\delta}[\min_{i,j} \|(p(f(x')^{(j)}) - (p(f(x)^{(i)}) + \delta_t)\|^2] \quad (2)$$

The first term in this loss enforces that the encoder / decoder pair $\hat{f}, \hat{g}$ capture enough information in the slot representations s_i to reconstruct x. The second term ensures that the function that projects from slot representation to latents $\hat{p}$ disentangles the slot representations into individual properties. The offset δ could be known or unknown to the model, and for the remainder of this paper, we focus on the more challenging and natural case of unknown offsets. See appendix E for more details.

5 Empirical evaluation

Setup. We evaluated our method on 2D and 3D synthetic image datasets that allowed us to carefully control various aspects of the environment, such as the number of objects, their sizes, shapes, colors, relative position, and dynamics. Examples of our 2D and 3D datasets are shown in figures 1,2 respectively. The object-wise true latents in either dataset consist of $z = (p_x, p_y, h, s, r, \phi)$, where p_x, p_y denote the coordinates of the center of an object, followed by color hue h, shape s, size r, and rotation angle ϕ about the z-axis. Therefore, we deal with both discrete and continuous properties. For further details on the dataset generation, see appendix H.6.

Table 1: Permutation Disentanglement (MCC) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR (supervised) achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$			$\text{pos}_x, \text{pos}_y$, color, size, rotation		
Model	$k = 2$	$k = 3$	$k = 4$	$k = 2$	$k = 3$	$k = 4$
Ours	**1.00** ± 0.01	**1.00** ± 0.01	0.98 ± 0.01	**0.95** ± 0.01	**0.93** ± 0.00	**0.94** ± 0.01
SA-RP	0.80	0.90	0.82	0.58	0.52	0.50
SA-PC	**1.00**	**1.00**	**1.00**	0.86	0.85	0.84
CNN†	0.96 ± 0.02	0.99 ± 0.01	0.98 ± 0.02	0.91 ± 0.01	0.89 ± 0.01	0.90 ± 0.01
CNN	0.40 ± 0.01	0.25 ± 0.03	0.21 ± 0.01	0.58 ± 0.00	0.42 ± 0.00	0.27 ± 0.01

Baselines. Our baselines were selected to use a series of linear probes to evaluate how much of the property information is already present in the slot representations s_i of a vanilla SA-MESH implementation which was only trained for reconstruction. We compare this with our approach which

Table 2: MCC scores on 3D shapes test set under *unknown* fully sparse perturbations. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$, color			$\text{pos}_x, \text{pos}_y$, color, size, rotation		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**0.99** ± 0.01	**0.99** ± 0.00	**0.99** ± 0.01	**0.89** ± 0.02	**0.92** ± 0.03	**0.92** ± 0.02
SA-RP	0.62	0.54	0.54	0.49	0.50	0.46
SA-PC	0.69	0.68	0.70	0.64	0.77	0.78

explicitly optimizes to disentangle the properties with the weakly supervised loss in Eq 2. For the baselines, we mapped from the slot representations to a d-dimensional latent space with *random projections* (RP)—which preserve distances in the projected space to obtain a crude estimate of vanilla slot attention disentanglement—the first d *principal components* (PC), and *linear regression* (LR) which provides a supervised upper-bound on what is achievable with linear maps. Finally, we also included a standard ResNet18 (He et al., 2016) (denoted *CNN*) trained with Ahuja et al. (2022b)'s procedure that does not address injectivity issues, and the same trained on a DGP which is modified to be injective[2] (denoted *CNN*†).

Disentanglement Metrics. We compared $\hat{z}$—the projections of non-background slots—to the true latents z of objects to measure the disentanglement of the properties in $\hat{z}$. We evaluated identifiability of $\hat{z}$ either up to affine transformations or up to permutation and scaling. These two metrics were computed by fitting a linear regression between $z, \hat{z}$ and reporting the coefficient of determination R^2, and using the mean correlation coefficient (MCC) (Hyvarinen and Morioka, 2016, 2017).

Results. The results in Tables 1 (with additional results in appendix G.1) confirmed that as long as the generative function is injective we can empirically achieve identification (see *CNN*†). But the moment we drop any ordering over the objects and render x via a non-injective function, then identification via ResNet18, which is suited only to injective generative functions, fails disastrously (see the row corresponding to *CNN*). On the other hand, we can see that our method has no difficulty identifying object properties because it treats them as a set by leveraging slot attention and a matching procedure. Additionally, the shared structure of learned latents in our method significantly improves the sample efficiency for disentanglement (see appendix G.3). The strong performance of the principal components of vanilla SA-MESH on the position disentanglement task likely results from the positional encoding. On the more complex task that also involves color, size and rotation, MCC performance drops for SA-PC, though it is still surprisingly high given that the model is just trained for reconstruction. This is likely because these are very simple images with properties that were selected independently, uniformly at random so the slot principal components align with the ground-truth axes of variation in the data. For 3D shapes we do not evaluate non-injective CNN since it failed consistently in the simpler 2D dataset. The results in Table 2 and G.2 essentially confirm our findings in the simpler 2D dataset, and demonstrate how treating the scene as a set with our method results in perfect disentanglement of object properties.

6 Conclusion

This study establishes a connection between causal representation learning and object-centric learning, and (to the best of our knowledge) for the first time shows how to achieve disentangled representations in environments with multiple interchangeable objects. The importance of recognizing this synergy is two-fold. Firstly, causal representation learning has largely ignored the subtleties of objects in assuming injectivity and fixed $\mathbb{R}^d$ representations. Conversely, object-centric learning has not dealt with the challenge of unsupervised disentanglement. Yet disentangled representations can significantly improve a model's generalization capabilities under distribution shifts, and could also allow for learning parsimonious models of the dynamics when such proper representations are achieved, which we deem as important avenues for future research. In this study we provided empirical evidence showcasing the successful disentanglement of object-centric representations through the fusion of slot attention with recent advances in causal representation learning.

[2]We make g injective by using the properties that are not the target of disentanglement, i.e., if x, y are the target properties, we will uniquely color each object based on its order in the default permutation.

References

Ahuja, K., Hartford, J., and Bengio, Y. (2022a). Properties from mechanisms: an equivariance perspective on identifiable representation learning. In *International Conference on Learning Representations*.

Ahuja, K., Hartford, J., and Bengio, Y. (2022b). Weakly supervised representation learning with sparse perturbations. In *Neural Information Processing Systems*.

Ahuja, K., Mahajan, D., Wang, Y., and Bengio, Y. (2023). Interventional causal representation learning.

Brady, J., Zimmermann, R. S., Sharma, Y., Schölkopf, B., von Kügelgen, J., and Brendel, W. (2023). Provably learning object-centric representations. *arXiv preprint arXiv: 2305.14229*.

Brehmer, J., De Haan, P., Lippe, P., and Cohen, T. (2022). Weakly supervised causal representation learning. *arXiv preprint arXiv:2203.16437*.

Chang, M., Griffiths, T. L., and Levine, S. (2022). Object representations as fixed points: Training iterative inference algorithms with implicit differentiation. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv: 1406.1078*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transportation distances. *NEURIPS*.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2019). Genesis: Generative scene inference and sampling with object-centric latent representations. *CoRR*, abs/1907.13052.

Goyal, A., Lamb, A., Gampa, P., Beaudoin, P., Levine, S., Blundell, C., Bengio, Y., and Mozer, M. (2020). Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv preprint arXiv:2006.16225*.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2019). Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*.

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.-T. D., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M. S. M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K. M., Zhong, F., and Tagliasacchi, A. (2022). Kubric: a scalable dataset generator. *Computer Vision And Pattern Recognition*.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR.

Gresele, L., Kügelgen, J. V., Stimper, V., Schölkopf, B., and Besserve, M. (2021). Independent mechanism analysis, a new concept? In *Advances in Neural Information Processing Systems*.

Hayes, B., Saitis, C., and Fazekas, G. (2023). The responsibility problem in neural networks with unordered targets.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Huang, Q., He, H., Singh, A., Zhang, Y., Lim, S.-N., and Benson, A. R. (2020). Better set representations for relational reasoning. In *NeurIPS*.

Hyvarinen, A. and Morioka, H. (2016). Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in Neural Information Processing Systems*, 29.

Hyvarinen, A. and Morioka, H. (2017). Nonlinear ica of temporally dependent stationary sources. In *Artificial Intelligence and Statistics*, pages 460–469. PMLR.

Hyvarinen, A., Sasaki, H., and Turner, R. (2019). Nonlinear ica using auxiliary variables and generalized contrastive learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 859–868. PMLR.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430.

Khemakhem, I., Kingma, D., Monti, R., and Hyvarinen, A. (2020a). Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR.

Khemakhem, I., Monti, R., Kingma, D., and Hyvarinen, A. (2020b). Ice-beem: Identifiable conditional energy-based deep models based on nonlinear ica. *Advances in Neural Information Processing Systems*, 33:12768–12778.

Kipf, T., van der Pol, E., and Welling, M. (2020). Contrastive learning of structured world models.

Lachapelle, S. and Lacoste-Julien, S. (2022). Partial disentanglement via mechanism sparsity.

Lachapelle, S., Mahajan, D., Mitliagkas, I., and Lacoste-Julien, S. (2023). Additive Decoders for Latent Variables Identification and Cartesian-Product Extrapolation. arXiv:2307.02598 [cs, stat].

Lachapelle, S., Rodriguez, P., Sharma, Y., Everett, K. E., PRIOL, R. L., Lacoste, A., and Lacoste-Julien, S. (2022). Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA. In *First Conference on Causal Learning and Reasoning*.

Lin, Z., Wu, Y., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020). SPACE: unsupervised object-oriented scene representation via spatial attention and decomposition. *CoRR*, abs/2001.02407.

Lippe, P., Magliacane, S., Löwe, S., Asano, Y. M., Cohen, T., and Gavves, E. (2022). Citris: Causal identifiability from temporal intervened sequences. *arXiv preprint arXiv:2202.03169*.

Lippe, P., Magliacane, S., Löwe, S., Asano, Y. M., Cohen, T., and Gavves, E. (2023a). Biscuit: Causal representation learning from binary interactions. In *The 39th Conference on Uncertainty in Artificial Intelligence*.

Lippe, P., Magliacane, S., Löwe, S., Asano, Y. M., Cohen, T., and Gavves, E. (2023b). Causal representation learning for instantaneous and temporal effects in interactive systems.

Locatello, F., Poole, B., Rätsch, G., Schölkopf, B., Bachem, O., and Tschannen, M. (2020a). Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020b). Object-centric learning with slot attention. *CoRR*, abs/2006.15055.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *International Conference on Learning Representations*.

Moran, G. E., Sridhar, D., Wang, Y., and Blei, D. (2022). Identifiable deep generative models via sparse decoding. *Transactions on Machine Learning Research*.

Nguyen, T., Mansouri, A., Madan, K., Khuong, N. D., Ahuja, K., Liu, D., and Bengio, Y. (2023). Reusable slotwise mechanisms. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Reddy, P., Wisdom, S., Greff, K., Hershey, J., and Kipf, T. (2023). Audioslots: A slot-centric generative model for audio separation. *IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634.

Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.-J., He, T., Zhang, Z., Scholkopf, B., Brox, T., and Locatello, F. (2022). Bridging the gap to real-world object-centric learning. *International Conference on Learning Representations*.

Shinners, P. (2011). Pygame. http://pygame.org/.

Sinkhorn, R. and Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21:343–348.

Squires, C., Seigal, A., Bhate, S. S., and Uhler, C. (2023). Linear causal disentanglement via interventions. In *International Conference on Machine Learning*. PMLR.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018a). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *CoRR*, abs/1802.10353.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018b). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations*.

Zhang, Y., Hare, J., and Prügel-Bennett, A. (2020). FSPool: Learning set representations with featurewise sort pooling. In *International Conference on Learning Representations*.

Zhang, Y., Zhang, D. W., Lacoste-Julien, S., Burghouts, G. J., and Snoek, C. G. (2022). Multiset-equivariant set prediction with approximate implicit differentiation. In *International Conference on Learning Representations*.

Zhang, Y., Zhang, D. W., Lacoste-Julien, S., Burghouts, G. J., and Snoek, C. G. M. (2023). Unlocking slot attention by changing optimal transport costs. In *International Conference on Learning Representations*.

Zheng, Y., Ng, I., and Zhang, K. (2022). On the identifiability of nonlinear ica: Sparsity and beyond. *Advances in Neural Information Processing Systems*, 35:16411–16422.

A Proofs

A.1 Proof of proposition 1

Proof. The contrapositive of the definition of injectivity states that $z_1 \neq z_2$ implies $g(z_1) \neq g(z_2)$, but by definition of $g(\cdot)$, there exist $z_1 \neq z_2$ such that $g(z_1) = g(z_2)$. In particular, for any set Z and permutations $\pi_1 \neq \pi_2 \in \text{Sym}(k)$, the vectors $\mathbf{vec}_{\pi_1}(Z) = z_1 \neq z_2 = \mathbf{vec}_{\pi_2}(Z)$. □

A.2 Proof of proposition 2

Proof Sketch. Consider Figure 3, notice that if we perform a $90°$ rotation in pixel space of the image, the image is identical, but the latent space has been permuted, since each ball has swapped positions. Because the image on the left and the image on the right are identical in pixel space, any encoder, $f : \mathcal{X} \to \mathbb{R}^{kd}$, will map them them to identical latents. There exists a continuous pixel-space rotation from $0°$ to $90°$, but it must entail a discontinous swap in which latent is *responsible* for which part of pixel-space according to the encoder. □

A.3 Proof of theorem 1

We want to compare the number of perturbations needed to disentangle shared properties with a standard encoder to those needed by an object-centric encoder. Our strategy will be as follows,

1. Setup a data generating process with multiple objects where injectivity holds by construction so that we can restate Theorem 1 from Ahuja et al. (2022b) to show they need $k \times d$ perturbations.
2. Define an object-centric architecture in terms of the object-wise partitions that we defined in Definition 1.
3. Restate an analog of Theorem 1 from Ahuja et al. based on the object-centric encoder.
4. Theorem 1 in the main text will follow as a collary of the difference between the number of perturbations used in the two theorems above.

We begin by defining a data generating process such that $g(\mathbf{vec}_\pi(Z))$ is injective by construction. We can achieve this by appending an id, i, to each z_i in Z, such that $Z = \{z_i \oplus [i]\}_{i=1}^k$ where $\oplus$ denotes concatenation, and then choosing $\mathfrak{g}$ such that x depends on i (for example, each i could be rendered in a different color). Like Ahuja et al., we assume we have data that is perturbed by $\Delta := \{\{\delta_{i,j}\}_{j=1}^d\}_{i=1}^k$, a set of 1-sparse perturbations that perturbs each of the d properties from each of the k objects. Taken together, we have the following data generating process (DGP),

$$Z = \{z_i \oplus [i]\}_{i=1}^k \sim \mathbb{P}_Z,\ x := \mathfrak{g}(Z) \quad \tilde{z}_{j,l} := z_j + \delta_{j,l}\, \forall\, \delta_{i,j} \in \Delta, \quad \tilde{x}_{j,l} := \mathfrak{g}(\{z_1, \ldots, \tilde{z}_{j,l}, \ldots, z_k\}) \tag{3}$$

where each object has d shared properties, $z_i \in \mathbb{R}^d$, and $z_{i,j}$ and $z_{i',j}$ are of the same type—e.g. position x, hue, etc.— for all j. As before, assume $\mathfrak{g}$ is injective, and define $g^* = g(\mathbf{vec}_{\pi^*}(Z))$ where π^* is the permutation that sorts Z by the index i, so that g^* is injective by construction.

Now, Ahuja et al. show that if the encoder, $\hat{f} : \mathcal{X} \to \mathbb{R}^{kd}$, is chosen to minimize the following loss,

$$\hat{f} \in \arg\min_{f'} E_{x,x',\delta}\left[\left(f'(x) + \delta - f'(x')\right)^2\right] \tag{4}$$

and the following assumptions hold,

Assumption 1. *The dimension of the span of the perturbations in equation 3 is kd, i.e.,* $\dim\Big(\text{span}(\Delta)\Big) = kd$.

Assumption 2. *$a(z) := f \circ g^*(z)$ is an analytic function. For each component $i \in \{1, \cdots, kd\}$ of $a(z)$ and each component $j \in \{1, \cdots, kd\}$ of z, define the set $\mathcal{S}^{ij} = \{\theta \mid \nabla_j a_i(z + b) = \nabla_j a_i(z) + \nabla_j^2 a_i(\theta) b, z \in \mathbb{R}^{kd}\}$, where b is a fixed vector in $\mathbb{R}^{kd}$. Each set $\mathcal{S}^{ij}$ has a non-zero Lebesgue measure in $\mathbb{R}^{kd}$.*

Then we have,

Theorem 2 ((Ahuja et al., 2022b)). *Assume we have data from the DGP in equation 3 and assumption 1 and 2 hold and the number of perturbations per example equals the latent dimension, $m = kd$, then the encoder that solves equation 3 identifies true latents up to permutation and scaling, i.e. $\hat{z} = \Pi\Lambda z + c$, where $\Lambda \in R^{kd\times kd}$ is an invertible diagonal matrix, $\Pi \in R^{kd\times kd}$ is a permutation matrix and c is an offset.*

Proof. See Ahuja et al. for the proof. □

Now, consider an object-centric architecture encoder of the form $F(x) := \{\mathfrak{f}(x^i)\}_{x^i\in P}$ where P is an object-wise partition and $\mathfrak{f} : \mathcal{X} \to \mathbb{R}^d$. Let $\sigma \in \Sigma$ denote a permutation of the latents from the set of all $k-$permutations. Let:

$$\hat{F}(x) \in \arg\min_{\mathfrak{f}} E_{x,x',\delta}[\min_{\sigma'\in\Sigma} \|((\mathfrak{f}(x')^{\sigma'(i)}) - ((\mathfrak{f}(x)^{(i)}) + \delta)\|^2] \tag{5}$$

Note that since δ is non-zero for only one pair of patches $x^{(i)}, x^{(i)'}$ and zero otherwise, the minimizer over Σ is almost surely unique. Assumptions 3 and 4 are analogs of 1 and 2 above, but make reference to the dimensionality of the co-domain of $\mathfrak{f}$ rather than f.

Assumption 3. *The dimension of the span of the perturbations in equation 3 is d, i.e., $\mathsf{dim}\Big(\mathsf{span}(\Delta)\Big) = d$.*

Assumption 4. *$a(z) := \mathfrak{f} \circ g^*(z)$ is an analytic function. For each component $i \in \{1, \cdots, d\}$ of $a(z)$ and each component $j \in \{1, \cdots, d\}$ of z, define the set $\mathcal{S}^{ij} = \{\theta \mid \nabla_j a_i(z + b) = \nabla_j a_i(z) + \nabla_j^2 a_i(\theta)b, z \in \mathbb{R}^d\}$, where b is a fixed vector in $\mathbb{R}^d$. Each set $\mathcal{S}^{ij}$ has a non-zero Lebesgue measure in $\mathbb{R}^d$.*

With this setup, the following theorem follows directly from Theorem 2 as a reduction from the multi-object to single-object setting,

Theorem 3. *Assume we have data from the DGP in equation 3 and assumption 3 and 4 hold and the number of perturbations per example equals the latent dimension, $m = d$, then the encoder that solves equation 5 for an object-wise partition P, identifies true latents up to permutation and scaling, i.e. $\hat{z} = \Pi\Lambda z + c$, where $\Lambda \in R^{kd\times kd}$ is an invertible diagonal matrix, $\Pi \in R^{kd\times kd}$ is a permutation matrix and c is an offset.*

Proof. Because P is an object-wise partition, the function that produces each $x^{(i)} \in P$ is injective with respect to some $z_{\sigma(i)}$ (i.e. one of the object's latents). Thus for each $x^{(i)}$, the solution to equation 5 is equivalent to the single object setting with $k = 1$, and thus theorem 2 applies, which implies that $\mathfrak{f}(x^{(i)}) = \hat{z}_i = \Pi\Lambda z_i + c$ for all i. Now let $\hat{z} = \mathbf{vec}_\pi(\{\hat{z}_i\}_{i=1}^k)$. Because each $\hat{z}_i$ is identified up to a permutation, scaling and offset, and for any π, there exists a Π such that $\hat{z} = \Pi\Lambda z + c$ which completes the result. □

Corollary 1. *If the assumptions for Theorem 2 and 3 hold and a data generating process outputs observations containing k objects with shared properties, then an object-centric architecture of the form $F(x) := \{\mathfrak{f}(x^{(i)}\}_{x^{(i)}\in P}$ will disentangle in $1/k$ fewer perturbations than an encoder of the form $f : \mathcal{X} \to \mathbb{R}^{kd}$.*

Proof. This follows directly from comparing the the number of perturbations required in Theorems 2 and 3. □

B Related Work

Causal representation learning Our work builds on the nascent field of causal representation learning (Schölkopf et al., 2021). In particular, our disentanglement approaches builds on ideas in Ahuja et al. (2022b) which uses the same assumptions as Locatello et al. (2020a) but relaxes the requirement that the latent variables are independently distributed. These approaches form part of a larger body of recent work that shows the importance of sparsity and weak supervision from actions in disentanglement (Lachapelle et al., 2022; Lachapelle and Lacoste-Julien, 2022; Brehmer et al., 2022; Lippe et al., 2022, 2023b,a). This literature builds on the foundational work from the nonlinear

independent component analysis (ICA) literature (Hyvarinen and Morioka, 2016, 2017; Hyvarinen et al., 2019; Khemakhem et al., 2020a).

Object-centric learning. Natural data can often be decomposed into smaller entities—objects—that explain the data. The overarching goal of object-centric learning is to model such data in terms of these multiple objects. The reason for this is simple: it is usually easier to reason over a small set of relevant objects rather than, for example, a large grid of feature vectors. Representing data in this way has downstream benefits like better robustness (Huang et al., 2020). An important line of research in this area is how to obtain such objects from data like images and video in the first place. Typically, a reconstruction setup is used: given an image input, the model learns the objects in the latent space, which are then decoded back into the original image with a standard reconstruction loss (Locatello et al., 2020b; van Steenkiste et al., 2018b). GENESIS (Engelcke et al., 2019) was introduced as the first 3D object-centric generative model for visual scenes capable of both generating and decomposing scenes. The Slot Attention (Locatello et al., 2020b) module is presented as an architectural component to extract object-centric representations from perceptual features. It produces task-dependent abstract representations called "slots" through competitive attention rounds, enabling generalization to unseen compositions. Contrastively-trained Structured World Models (C-SWMs) (Kipf et al., 2020) utilize a contrastive approach for object representation learning with graph neural networks modeling object interactions to learn a structured world model from raw sensory data. Nguyen et al. (2023) propose RSM, a conceptually close idea to our work. They jointly learn object-centric representations with a modular dynamics model by minimizing a rolled out reconstruction loss. However, they do not obtain any disentanglement of object properties, and the form of our proposed weak-supervision provides insights to the effectiveness of their method for improving generalization.

We use slot attention since it makes very few assumptions about the desired data. For instance, some methods model foreground differently from background. Additionally, DINOSAUR (Seitzer et al., 2022) shows recent success on more complex images, which demonstrates the versatility of the slot attention approach. While in general object-centric models operate on image inputs and thus identify visual objects, it is in principle applicable to other domains like audio (Reddy et al., 2023) as well.

C Definition of object separability

To define object separability formally, we will need to consider a partition P of an image into k disjoint subsets of pixels $P = \{x^{(1)}, \ldots, x^{(k)}\}$ indexed by an index set $\mathcal{I}_P = \{1, \ldots, k\}$; further, denote an index set that indexes the set of latent variables Z as $\mathcal{I}_Z$. We can then say,

Definition 1. *An image, x, is **object-separable** if there exists an **object-wise** partition P and a bijection $\sigma : \mathcal{I}_P \to \mathcal{I}_Z$ that associates each subset of pixels in P with a particular element of the set of latents, z_i, such that each subset of pixels $x^{(i)} \in P$ is the output of an injective map with respect to its associated latent $z_{\sigma(i)}$. That is, for all i, $({x^{(i)}}' \subset \mathfrak{g}(Z'),\ x^{(i)} \subset \mathfrak{g}(Z))$, we have that ${x^{(i)}}' = x^{(i)}$ implies $z'_{\sigma(i)} = z_{\sigma(i)}$.*

This definition says that an image can be separated into objects if it can be partitioned into parts such that each part is rendered via an injective map from some latent z_i. We can think of each $x^{(i)}$ as a patch of pixels, with a bijection σ that relates each of the k patches of pixels in the partition $\{x^{(i)}\}_{i=1}^k$ to a latent variable in $Z = \{z_i\}_{i=1}^k$. Each patch "depends" on its associated latent via an injective map.

Brady et al. (2023) and Lachapelle et al. (2023) give two different formal characterizations of partitions P that are consistent with our object-wise definition. Brady et al.'s characterization requires that a differentiable generative function $\mathfrak{g}$ is *compositional*, in the sense that each $x^{(i)} \in P$ only functionally depends[3] on a single $z_j \in Z$, and *irreducible* in the sense no $x^{(i)} \in P$ can be further decomposed into non-trivial subsets that have functionally independent latents. Lachapelle et al.'s assumption is weaker than ours in that they only require that the generative function is defined as $\mathfrak{g}(Z) = \sigma(\sum_{z_i \in Z} g_i(z_i))$ where σ is an invertible function, and that $\mathfrak{g}$ is a diffeomorphism that is "sufficiently nonlinear" (see Assumption 2 Lachapelle et al., 2023); object-separable images are a special case with σ as the identity function and each $g_i(\cdot)$ rendering a disjoint subset of x, and hence their results apply to our setting.

[3]Functional dependence is defined by non-zero partial derivatives, i.e. $\frac{\partial x^i}{\partial z_j} \neq 0$.

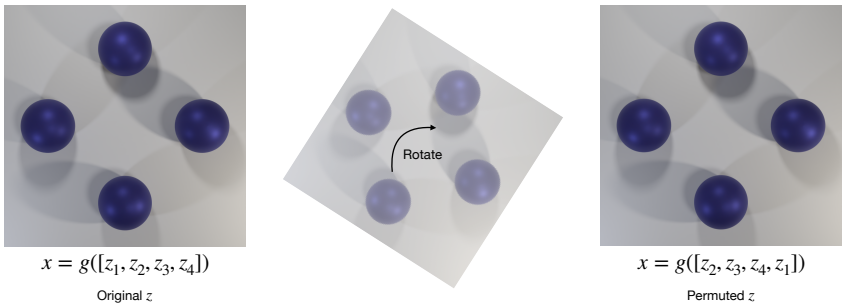

Figure 3: An illustration of the object identity problem. Permuting the order of the latents $[z_1, z_2, z_3, z_3]$ is equivalent to a 90 degree rotation in pixel-space.

D Background on slot-attention-based architectures

Slot attention (Locatello et al., 2020b) is a neural network component that, intuitively, summarizes the relevant information in the input set (most commonly, image features with position embeddings) into a smaller set of so-called "slots". Each slot is a feature vector that can be thought of as capturing information about one "object" in the input set, which usually comprises multiple elements of the input set. This is done by repeating cross-attention between the inputs and the slots to compute per-slot updates.

In the traditional set-up, these slots are then used to reconstruct the input with an auto-encoder objective: each slot is decoded into a separate image through a shared image decoder, which is followed by merging these per-slot images into a single reconstructed image. Ideally, slot attention is able to decompose the original image into distinct objects, each of which is modeled by a single slot.

More concretely, slot attention takes as input a matrix $\mathbf{X} \in \mathbb{R}^{n \times c}$ with n as the number of inputs and c the dimensionality of each input. We also randomly initialize the slots $\mathbf{Z}^{(0)} \in \mathbb{R}^{m \times d}$. We start by computing the query, key, and value matrices as part of cross-attention.

$$\mathbf{Q}^{(t)} = \mathbf{Z}^{(t)}\mathbf{W}_Q \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V \tag{6}$$

This is followed by the normalized cross-attention to determine the attention map $\mathbf{A} \in \mathbb{R}^{m \times n}$, then a GRU (Cho et al., 2014) to apply this update to the slots.

$$\mathbf{A}^{(t)} = \mathrm{normalize}(\mathbf{Q}^{(t)}\mathbf{K}^\top) \tag{7}$$

$$\mathbf{Z}^{(t+1)} = \mathrm{GRU}(\mathbf{Z}^{(t)}, \mathbf{A}^{(t)}\mathbf{V}) \tag{8}$$

The function $\mathrm{normalize}$ encourages slots to compete for inputs by applying a softmax over slots and normalizing the weights for each input to sum to one. After T steps, the algorithm outputs $\mathbf{Z}^{(T)}$, a set of m embedding vectors $\{\mathbf{z}_i^{(T)}\}_{i=1}^m$ that can be used as input to a shared image decoder.

SA-MESH. The specific version of slot attention we use in this paper is SA-MESH (Zhang et al., 2023). It makes regular slot attention more powerful by giving it the ability to break ties between slots more effectively. [4] In practice, this improves the quality of the individual slot representations significantly due to less mixing of unrelated inputs into the same slot.

The key difference with regular slot attention is that it features an entropy minimization procedure to approximate an optimal transport solution, which makes the attention map sparse. The connection to optimal transport is made by the use of the standard Sinkhorn algorithm (Sinkhorn and Knopp, 1967; Cuturi, 2013).

$$\mathrm{MESH}(\mathbf{C}) = \arg\min_{\mathbf{C}} H(\mathrm{sinkhorn}(\mathbf{C})) \tag{9}$$

$$\mathbf{A}^{(t)} = \mathrm{sinkhorn}(\mathrm{MESH}(\mathbf{Q}^{(t)}\mathbf{K}^\top)) \tag{10}$$

The optimization problem is solved by unrolling gradient descent, with a noisy initialization to ensure that ties are broken.

[4] Concretely, it makes the mapping from the initial slots to the final slots *exclusively multiset-equivariant* (Zhang et al., 2022) rather than permutation-equivariant/set-equivariant.

E Alternative perturbation mechanisms

Dense vs. Sparse. We can have a number of assumptions on the perturbation mechanisms and the nature of model's knowledge about those mechanisms. In the most general case, suppose $\mathcal{M} = \{m^1(\cdot), m^2(\cdot), ..., m^k(\cdot)\}$ denotes the set of all possible perturbation mechanisms. To obtain x', the perturbed variant of x, we then select a subset of $k' \leq k$ objects as targets that undergo perturbations determined by a subset of k' mechanisms $\mathcal{M}' \subset \mathcal{M}$ from the set of all possible perturbation mechanisms. The correspondence between the k' mechanisms and k' perturbed objects is decided by a random permutation $\pi_t^{\mathcal{M}}$, i.e. $i = \pi_t^{\mathcal{M}}[j]$ means that mechanism i governs the transition dynamics of object j to produce z_{t+1}^j (for objects that are not supposed to change from $t \to t+1$ a dummy mechanism with index -1 can be assumed which results in no change). A mechanism $m^i(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ in $\mathcal{M}$ is a vector-valued function that operates on object-wise true latents z_t^j and outputs $z_{t+1}^j = z_t^j + \delta_i$ such that $i = \pi_t^{\mathcal{M}}[j]$. Perturbation vectors δ_i could be sparse or not. The subset $\mathcal{M}'$ can contain $k' = k$ mechanisms to perturb all of the k objects in the environment, and if none of the $k' = k$ resulting perturbations is sparse, we denote the set $\mathcal{M}'$ as *fully dense perturbations*, i.e., all of the properties of all objects will change from $t \to t+1$. $\mathcal{M}'$ can also contain at least one object but not all of them ($1 \leq k' < k$) with sparse or dense perturbations, or it may consist of only a single object ($k' = 1$) that is perturbed by a fully sparse mechanism, one that only alters a single property and leaves the rest unchanged. We denote this scenario as *fully sparse perturbation*.

F Matching

Perturbations alter the properties of objects from $t \to t+1$ and the model has to infer which object's properties were perturbed to update its representations and minimize the latent loss equation 1. But recall that the model has no direct access to objects. It receives the observations at $t, t+1$ and encodes each of them to a set of slots $\mathcal{S}_t, \mathcal{S}_{t+1}$. These slots do not follow any fixed ordering, and moreover, there is no guarantee that each slot binds to exactly one unique object. Slots can also correspond to the background. Each perturbation δ^i changes the properties of some object z_t^j, so the model requires to find a pair of slots (s_t^u, s_{t+1}^v) that are bound to object z^j at t and $t+1$, respectively. Once the model figures out such a matching, then the latent loss that results in disentanglement can be computed via the projections of these slots $\hat{z}$:

$$\mathcal{L}_z = \sum_{i=1}^{m} \|\hat{z}_t^i + \delta_t^{\pi_t^{\mathcal{M}'}[i]} - \hat{z}_{t+1}^i\|^2, \qquad \hat{z}_t^i = p(s_t^u), \;\; \hat{z}_{t+1}^i = p(s_t^v) \tag{11}$$

The problem of finding a correspondence between slot projections at $t, t+1$ and the perturbations is an instance of the 3-dimensional matching. However, recall that we use fully sparse perturbations for our mechanisms, thus, the problem is significantly simplified.

When the model is presented with fully sparse perturbations, the 3-dimensional matching reduces to finding the minimum element in a $1-d$ cost array of size $|\mathcal{S}|^2$, where $|\mathcal{S}|$ is the number of slots. The reason is that since only one property of a single object z_t^i is being altered (and the model knows perturbations are fully sparse), the model can apply that 1-sparse perturbation δ_t (or the transformed version in the unknown setting) to all of slot projections $\hat{z}_t^j$ for $j \in \{1, \ldots, |\mathcal{S}|\}$ at time t, and only find *one pair* of slots from the set of $|\mathcal{S}|^2$ possible pairs at $t, t+1$, which correspond to the perturbed object:

$$(i, j) = \arg\min_{i', j'} \|\hat{z}_{t+1}^{j'} - (\hat{z}_t^{i'} + \delta_t)\|^2 \tag{12}$$

However, this minimization can be made simpler if we use the slots obtained at t to initialize the slots at $t+1$. This way, in practice, the order of slots at $t, t+1$ is very likely to be preserved, not only since the slots at $t+1$ are initialized with hints from t, but also the sparse perturbation makes it much easier for all slots to bind to the same object as only a small subset of the scene needs to be readjusted among the slots. Therefore the matching would reduce to a simple minimization over $|\mathcal{S}|$ elements:

$$i = \arg\min_{i'} \|\hat{z}_{t+1}^{i'} - (\hat{z}_t^{i'} + \delta_t)\|^2 \tag{13}$$

Note however that we still would use all slot projections for evaluation, the only difference with this matching scheme is that gradient signals are only propagated from the perturbed object (as they also should, since there is no change in other slots, there is nothing there to be learned that could help disentanglement.)

G Further Experimental Results

G.1 2D Shapes

Table 3 gives the linear disentanglement scores for the same experiment as that shown in table 1 in the main text.

Table 3: Linear Disentanglement (LD) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR, which is supervised by the ground truth latents, and is an upper bound on the disentanglement performance, achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$			$\text{pos}_x, \text{pos}_y$, color, size, rotation		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	**0.95** ± 0.01	**0.93** ± 0.01	**0.94** ± 0.02
SA-RP	0.92	0.96	0.94	0.75	0.70	0.68
SA-PC	1.00	1.00	1.00	0.93	0.88	0.86
CNN†	0.94 ± 0.05	0.99 ± 0.00	0.96 ± 0.03	0.87 ± 0.01	0.84 ± 0.01	0.86 ± 0.01
CNN	0.24 ± 0.01	0.13 ± 0.01	0.07 ± 0.01	0.35 ± 0.00	0.19 ± 0.00	0.08 ± 0.01

Tables 4,5 extend our results under unknown fully sparse perturbations on the 2D shapes dataset to more combinations of disentanglement target properties. We can observe that our results stay very close to the upper bound on the achievable performance which uses a *supervised* linear regression from slot projections $\hat{z}$ to *ground truth* latents z. These tables highlight once again the pivotal role of the injectivity assumption in achieving identification with conventional encoders that ignore the object-centricity of the environment (see the performance drop from *CNN*† to CNN, where the latter drops the unrealistic injectivity assumption).

Table 4: Linear Disentanglement (LD) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$, color			$\text{pos}_x, \text{pos}_y$, shape		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**1.00** ± 0.01	**0.98** ± 0.01	**0.99** ± 0.00	**1.00** ± 0.01	**0.98** ± 0.01	**0.99** ± 0.01
SA-RP	0.77	0.61	0.60	0.71	0.68	0.70
SA-PC	0.97	0.98	0.99	0.80	0.66	0.87
CNN†	1.00 ± 0.00	0.99 ± 0.01	0.98 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00
CNN	0.35 ± 0.00	0.15 ± 0.00	0.07 ± 0.01	0.32 ± 0.01	0.15 ± 0.01	0.11 ± 0.01

	$\text{pos}_x, \text{pos}_y$, color, shape		
Model	$k=2$	$k=3$	$k=4$
Ours	**0.99** ± 0.00	**0.98** ± 0.01	**0.99** ± 0.00
SA-RP	0.69	0.73	0.60
SA-PC	0.74	0.75	0.52
CNN†	1.00 ± 0.00	0.99 ± 0.01	1.00 ± 0.00
CNN	0.40 ± 0.00	0.21 ± 0.00	0.11 ± 0.00

G.2 3D Shapes

Quantitative Results. Tables 6, 7, 8 extend our results under unknown fully sparse perturbations on the 3D shapes dataset to more combinations of disentanglement target properties. Again, we can

Table 5: Permutation Disentanglement (MCC) scores on 2D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$, color			$\text{pos}_x, \text{pos}_y$, shape		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**1.00** ± 0.01	**0.95** ± 0.05	**0.97** ± 0.02	**0.99** ± 0.01	**0.99** ± 0.01	**0.99** ± 0.01
SA-RP	0.74	0.60	0.60	0.66	0.63	0.59
SA-PC	0.87	0.89	0.90	0.83	0.81	0.89
CNN†	1.00 ± 0.00	0.99 ± 0.01	0.99 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00
CNN	0.55 ± 0.01	0.35 ± 0.01	0.24 ± 0.01	0.52 ± 0.02	0.33 ± 0.02	0.28 ± 0.02

	$\text{pos}_x, \text{pos}_y$, color, shape		
Model	$k=2$	$k=3$	$k=4$
Ours	**0.99** ± 0.01	**0.98** ± 0.01	**0.99** ± 0.01
SA-RP	0.54	0.68	0.55
SA-PC	0.64	0.63	0.57
CNN†	1.00 ± 0.00	0.99 ± 0.01	1.00 ± 0.00
CNN	0.61 ± 0.00	0.43 ± 0.00	0.30 ± 0.00

observe the applicability of our method to this more complex 3D dataset that contains artifacts related to depth, occlusion, and lighting, to name a few. Again our results stay very close to the upper bound on the achievable performance which uses a *supervised* linear regression from slot projections $\hat{z}$ to *ground truth* latents z.

Table 6: Linear Disentanglement (LD) scores on 3D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$, size			$\text{pos}_x, \text{pos}_y$, color, rotation		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**0.98** ± 0.01	**0.98** ± 0.01	**0.98** ± 0.00	**0.98** ± 0.00	**0.97** ± 0.01	**0.98** ± 0.01
SA-RP	0.61	0.62	0.53	0.59	0.54	0.55
SA-PC	0.78	0.84	0.78	0.70	0.72	0.69

Table 7: Permutation Disentanglement (MCC) scores on 3D shapes test set under *unknown* fully sparse perturbations. All results are averaged over 3 seeds except those requiring to train SA-MESH from scratch that were trained only once. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$, size			$\text{pos}_x, \text{pos}_y$, color, rotation		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**0.96** ± 0.02	**0.96** ± 0.05	**0.96** ± 0.03	**0.98** ± 0.01	**0.98** ± 0.02	**0.97** ± 0.01
SA-RP	0.60	0.57	0.52	0.55	0.51	0.49
SA-PC	0.87	0.90	0.86	0.73	0.76	0.74

Qualitative Results. Figures 4-6 illustrate the learned disentangled (object-centric) representations. Each figure shows a sequence of 3D samples evolving over 5 steps (shown on the left), and how the learned representations respond to the perturbations (shown on the right). Perturbations include changing the object's $\text{pos}_x, \text{pos}_y$, color, ϕ (rotation). The model used here is trained with 3 slots, and the learned representations are the result of a projection layer learned via our weakly-supervised method applied to $64-$dimensional object-centric representations. The projection maps each slot from $\mathbb{R}^{64} \rightarrow \mathbb{R}^4$, i.e., the disentanglement target space. In figures 4-6, the 4 dimensions of such projections for object slots are presented over 5 steps, i.e., each set of colored lines shows the evolution of the projection of a slot corresponding to the object with the same color. Please refer to the figures for details of the perturbations. Lastly, we have kept the number of objects in these

Table 8: LD scores on 3D shapes test set under *unknown* fully sparse perturbations. SA-LR achieves a score of 1.0 in all settings.

	$\text{pos}_x, \text{pos}_y$, color			$\text{pos}_x, \text{pos}_y$, color, size, rotation		
Model	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
Ours	**0.99** ± 0.01	**0.99** ± 0.00	**1.00** ± 0.01	**0.91** ± 0.03	**0.95** ± 0.01	**0.93** ± 0.01
SA-RP	0.67	0.58	0.58	0.51	0.56	0.60
SA-PC	0.64	0.62	0.64	0.56	0.76	0.76

scenes to two for clarity of the presentation, however tables 8,2, 6,7 show that we achieve similar performances with other sets of properties and number of objects in the scene.

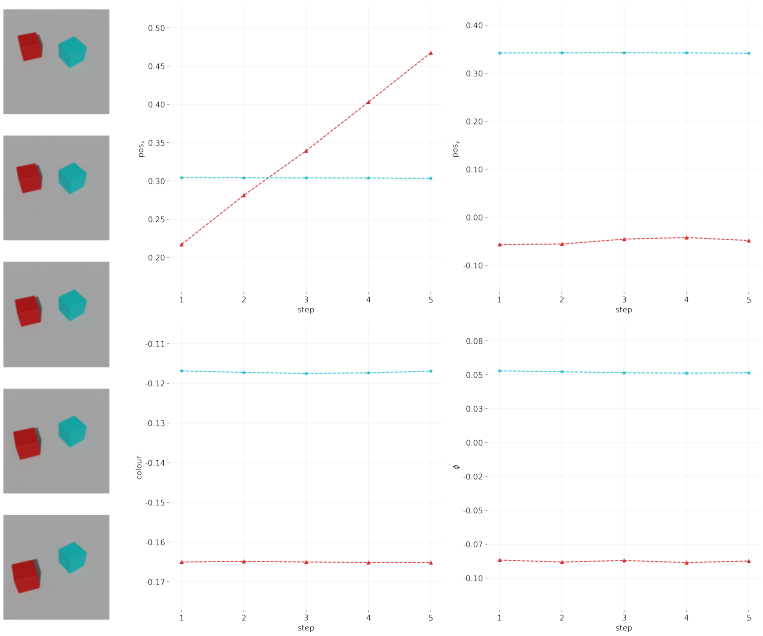

Figure 4: *(Left)* From top to bottom, each steps perturbs the pos_x coordinate of the red object by 0.2 in the *ground-truth latent space*, *(Right)* which is reflected (through an affine transform) as a linear increase in pos_x, while the rest of the properties for both objects remain the same, demonstrating that the learned representations are indeed disentangled. In generating these samples, camera has some non-zero angle w.r.t. the origin, therefore, perturbations in the x direction appear as vertical displacements.

G.3 Comparison of Sample Efficiency

Figure 7 demonstrates the sample efficiency of our object-centric model compared to a ResNet that achieves disentanglement with an injective DGP. Both models are trained with varying number of training samples that contain $k = 4$ objects for which $\text{pos}_x, \text{pos}_y$, color, shape are the disentanglement target properties. Since we sample 1-sparse perturbations uniformly, the training dataset size could be thought of as a proxy for the number of different perturbations a given configuration of objects would encounter. Although according to theory, the injective ResNet should require *at least* k times more perturbations to identify the latents up to affine transformations, we observe that the advantage of our object-centric model in terms of sample efficiency is much more pronounced in practice. Our method can achieve close to perfect disentanglement with as few as 100 training samples, while an injective ResNet takes 100 times more samples to raise to a comparable performance. This highlights the practical importance of exploiting the inherent set structure of objects in a scene for representation learning.

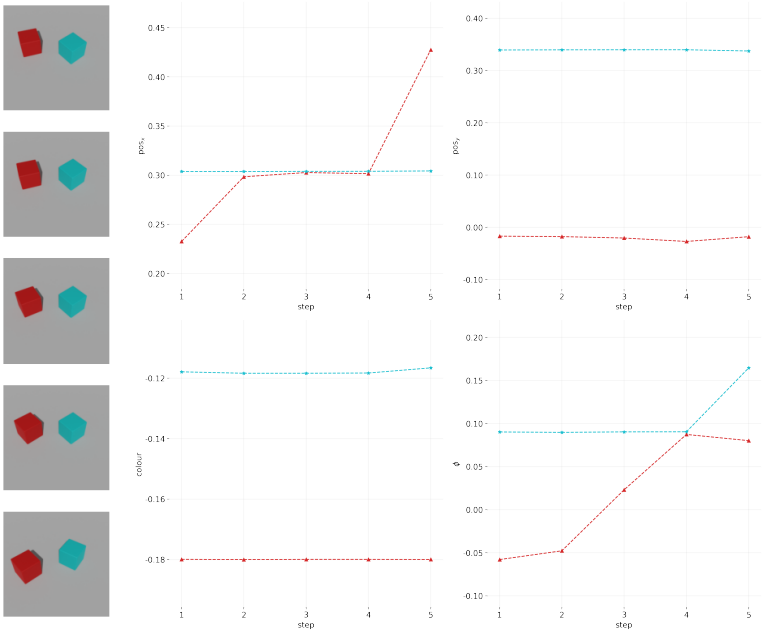

Figure 5: *(Left)* From top to bottom, the objects are perturbed as follows; Red: Downward perturbation so it sits at the same x coordinate as the Blue cube (1-2), Rotates counterclockwise along the $z-$axis to be at the same orientation as the Blue cube (2-4), Further downward perturbation (2 times the displacement from step 1 to 2). Blue: Rotates counterclockwise along the $z-$axis (4-5). *(Right)* Note how the learned representations mapping correctly reflects the similar positions and rotations in the ground-truth, i.e., both by having properties of objects coincide at the same value, and by preserving the ratio of perturbations.

H Implementation and Experimental Details

H.1 SA-MESH Architecture

For the slot attention architecture, we closely follow Locatello et al. (2020b), in particular, we use the same CNN encoder and decoder as they use for CLEVR, except for the initial resolution of the spatial broadcast decoder with 3D shapes where we use 4×4 since we are dealing with 64×64 images. We use a slot size of 64 and always use $n + 1$ number of slots, where n is the number of objects in the scene. We use 3 iterations for the recurrent updates in SA-MESH. For details concerning SA-MESH we follow Zhang et al. (2023). Additionally, we also truncate the backpropagation through slot updates as suggested by Chang et al. (2022) to improve training stability.

H.2 Disentanglement Heads

SA-MESH outputs $n + 1$ slots that are of size 64, yet we need to project each of these slots to a $d-$dimensional space so we can leverage the disentanglement method from Ahuja et al. (2022a,b). We can simply achieve this projection by a single MLP, however, we decided to allocate more parameters for this projection and use d separate projection heads mapping 64-dimensional vectors to d separate scalars. This way identification of different properties will not affect one another due to model capacity constraints. We stack the layers shown in table 9 to obtain a projection head per each property. The same set of d projections will be shared among all slots.

H.3 ConvNet Baseline

As a baseline for injective scenarios, we use a ResNet18 (He et al., 2016) with an output width of 128 that is passed through `LeakyReLU` activation, which is then followed by d linear projection heads (for the same reason we use separate disentanglement heads) that map the 128-dimensional output of the

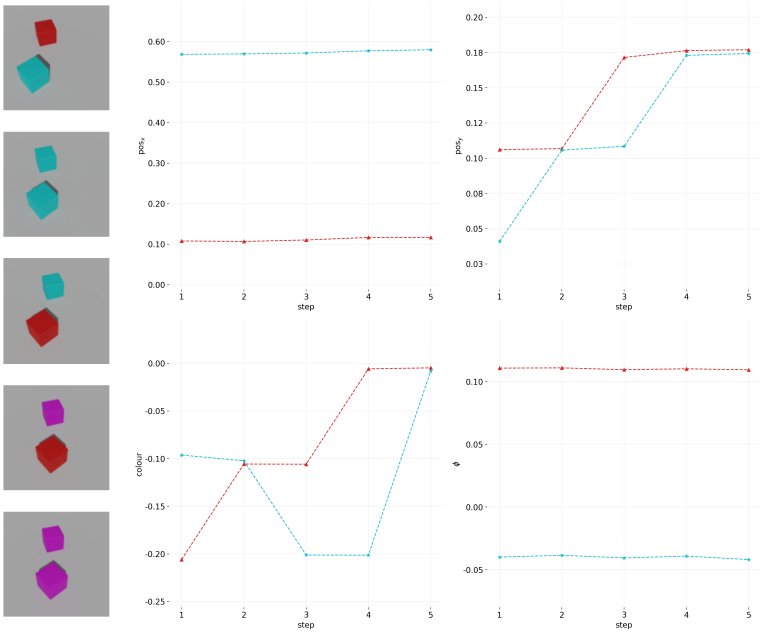

Figure 6: *(Left)* From top to bottom, the objects are perturbed as follows; Since the objects change color, let us call the red cube in the top frame to be object 1, and the blue one to be object 2. Object 1: The color is perturbed to be equal to object 2 (1-2), Moves to the right as its pos_y is perturbed by 0.2 (2-3), The color hue is once again perturbed to become purple (3-4). Object 2: Moves toward right (perturbation in pos_y by 0.2) to be at the same y coordinate as object 1 (1-2), The color hue is *decreased* so now the colors of objects 1,2 are swapped (2-3). Moves by 0.2 in the y direction to align with the other object once again (3-4), Change its color hue twice the previous color perturbation with the opposite sign to match the color of the other object (4-5). *(Right)* Red curves correspond to object 1, and the blue curves correspond to object 2. Again, notice the sections where the curves coincide, as well as the ratio of jumps in the properties, showing consistency of the learned representations with the ground-truth causal representation that gives rise to these observations.

Table 9: Layers in a projection head for disentanglement.

Layer	Input Size	Output Size	Bias	Activation
Linear (1)	64	32	True	ReLU
Linear (2)	32	32	True	ReLU
Linear (3)	32	16	False	ReLU
Linear (4)	16	1	False	ReLU

CNN encoder to d separate 1-dimensional scalars that should correspond to the target $d-$dimensional space.

H.4 Training

For each k and for each set of disentanglement target properties, we first train SA-MESH for 2000 epochs with a batch size of 64 for 2D shapes (as the images are 128×128), and a batch size of 128 for 3D shapes (since images are 64×64) on a single A100 GPU with 40GB of memory. We used a fixed schedule for the learning rate at 2×10^{-4}, and we used AdamW (Loshchilov and Hutter, 2017) with a weight decay of 0.01 along with $\epsilon = 10^{-8}, \beta_1 = 0.9, \beta_2 = 0.999$. SA-MESH was first solely trained by minimizing reconstruction error on the training set, then its disentanglement performance was reported on the test set for projection-based baselines (RP, PC, LR). Due to the high number of combinations of target disentanglement properties and k, we just trained SA-MESH for each configuration only once.

Unsupervised disentanglement with our method has an additional stage which takes the aforementioned pre-trained SA-MESH models and *jointly* minimizes the loss in equation 2. Note that at this

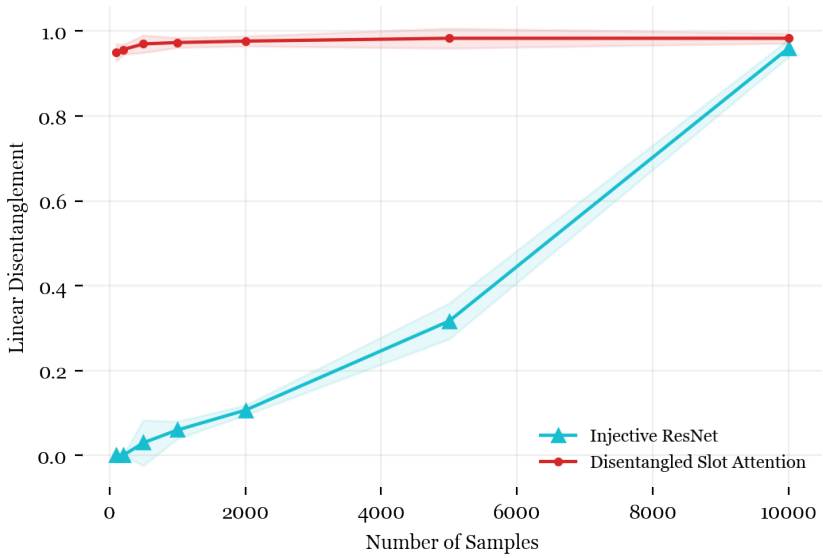

Figure 7: Comparing the disentanglement performance of an injective ResNet vs. our object-centric method based on the number of training samples. The dataset contains four 2D objects in which $\text{pos}_x, \text{pos}_y, \text{color}, \text{shape}$ can vary.

stage, the SA-MESH model is *not* frozen, so the gradients flow through its network as well and help adjust the slot representations with the signal from the latent loss. Under *known perturbations*, we use the actual perturbations from the DGP as δ_t to guide the model via equation 2, however, under *unknown perturbations* setting, we replace all perturbations δ_t by a hyperparameter C (see section 4). CNN baselines were trained similar to SA-MESH but for much shorter, i.e., 200 epochs, and usually converge very fast in less than 50 epochs.

H.5 Hyperparameter Optimization

We started around the hyperparameters used by Locatello et al. (2020b) and Zhang et al. (2023) where applicable, and tuned on small subsets of the 2D shapes training data based on linear and permutation disentanglement metrics. We considered 5 values for the learning rate $[2 \times 10^{-3}, 2 \times 10^{-4}, 10^{-4}, 6 \times 10^{-5}, 2 \times 10^{-5}]$. Larger batch sizes were always better and we were only constrained by memory in the case of 128×128 images of the 2D shapes dataset. We considered 2 values $[0.1, 0.5]$ for $|C|$, the fixed value representing all unknown perturbations, and found $|C| = 0.1$ to perform better. We also considered slot sizes $[64, 128]$ on a small subset of the 2D shapes training dataset. Lastly we considered 9 combinations for the relative importance of latent loss and reconstruction loss when training the disentanglement heads, i.e., we considered all combinations of $w_{\text{latent}} \in \{1, 10, 100\}$, $w_{\text{recons}} \in \{1, 10, 100\}$, and found the combination of $w_{\text{recons}} = 100, w_{\text{latent}} = 10$ to strike the optimal balance between maintaining good reconstructions and allowing the slot representations to give rise to disentangled projections.

H.6 Datasets

2D Shapes. We use `pygame` engine (Shinners, 2011) for generating multi-object 2D scenes. Object properties in both datasets include p_x, p_y, color, shape, size, and rotation angle. Based on object properties, they are each rendered and placed on a white background (for the 2D dataset) or placed on a floor that is illuminated by source lights and is being visited from somewhere above the floor (for the 3D dataset), and then aggregated to produce x_t. We discretized the range of color hues in order to test the model's ability to obtain disentangled representations in the simultaneous presence of both continuous (position, size, and rotation angle) and discrete properties (color and shape).

In the 2D dataset:

- p_x, p_y are generated randomly and uniformly in the $[0, 1]$ range, i.e., the boundaries of the scene, such that no two objects overlap and no object falls even partially outside the boundaries. Positional coordinates can be perturbed by any value in $[-0.2, 0.2]$.
- For color, we use HSV color representations and fix saturation (S) and value (V) at 0.6 and choose hue (H) from a set of values predefined before training (for instance $[0.0, 0.25, 0.5, 0.75]$). We adopted this 1-d representation to be consistent and have each property be represented by a scalar. Additionally we wanted to test the model's capacity when dealing with mixed discrete properties. Also, training Slot Attention or SA-MESH with discrete colors is computationally advantageous since the model will not have to deal with reconstructing all colors. However it should be noted that HSV is a cylindrical geometry with color hues being the angular dimension which results in values that have a distance of 1.0 being exactly the same color (given a fixed saturation and value). That is why a list of color hues such as $[0.0, 0.33, 0.66, 1.00]$ would not work since 0.0 and 1.0 are the same color, yet our model interprets the difference as a perturbation with the amount of 1.0, which is clearly wrong. A change of color from color i to j where i, j index the list of color hues H would be provided to the model as a perturbation in the amount of $(H[j] - H[i])/|H|$, where $|H|$ denotes the number of colors in H.
- Shape is also clearly discrete and is selected at random uniformly from the following set of shapes $S = \{\text{circle}, \text{square}, \text{triangle}, \text{heart}, \text{diamond}\}$. Note however, that the effects of perturbations need to be visible in the pixel space, and we should be wary of the disentanglement target properties; for instance if the rotation angle ϕ_t is a property we aim to disentangle with perturbations, then we should exclude *circle* from the set of possible shapes since a circle does not reflect in the pixel space the angle perturbations. A shape transformation from shape i to j where i, j index $\mathcal{S}$, would be provided to the model as a perturbation with the amount of $(j - i)/|\mathcal{S}|$.
- Size is a continuous property in the range $[0.12, 0.24]$ of the height or width of the image which is 1. It can be perturbed by any amount in $[-0.02, 0.02]$.
- Rotation angle is also another continuous property in $[0, \pi/4]$. Similar to color hues, since this property is also angular, we have limited the range not to encounter situations that appear the same in the pixel space but have very different rotation angles (a square that is rotated $\pi/2$ clockwise seems unaltered, or $\pi/4$ and $3\pi/4$ rotations both look the same for a square.). Angular perturbations can vary in the $[-0.2, 0.2]$ range.

We generate samples in pairs corresponding to $t, t+1$. For fully dense perturbations, we generate k vectors of dimension d, where k is the number of objects. We repeat the generation until the conditions of non-overlapping objects and non-identifiability are met, i.e., no two objects at either t or $t+1$ should overlap (before and after the perturbations), no object should fall in whole or partially out of the scene, and no two objects should be perturbed by d-dimensional offsets that are closer than some ϵ. The last condition is necessary for fully dense perturbations as otherwise the matching has no way of distinguishing which perturbation to assign to which object since the matching solely relies on the *difference* between $t, t+1$. For fully sparse perturbations, we are not constrained by the latter, and we only need to choose perturbations that do not push the chosen object out of boundaries, or make it overlap with another object. For any experiment we can have a subset of $\{\text{pos}_x, \text{pos}_y, \text{color}, \text{shape}, \text{size}, \text{rotation angle}\}$ as the properties we wish to disentangle by observing perturbations in the pixel space, and we call them *disentanglement target properties*. In the generation process, any non-target property will be fixed for all objects in the whole dataset to avoid introducing unwanted variance to the disentanglement of target properties, i.e., if we choose $\{\text{pos}_x, \text{pos}_y, \text{color}, \text{shape}, \text{rotation angle}\}$ as target properties, then all the objects in all samples would have the same fixed size. Lastly, we can choose to make a DGP injective or not. If we choose to make a DGP injective, we would index the objects and choose a property to be set for objects according to the indices, i.e., we can choose to make the DGP injective by color; Suppose $k = 4$ and the list of our color hues is $[0.0, 0.25, 0.5, 0.75]$. We would color the objects, which are now ordered according to some index set $\mathcal{I}$, according to $\mathcal{I}$. Non-target properties (excluding the injectivity imposing property) will again be kept fixed for the whole dataset, and target properties are generated according to fully dense or fully sparse perturbation schemes. The perturbations to all properties are *signed*, and this is especially crucial for discrete properties such as shape. The reason is that disentanglement is achieved through observing *relative* distances in the pixel space, and having only positive or only negative perturbations deprives the model of having a reference for each property.

For training we generate 1000 pair per target property such that the model on average sees at least 500 samples for either positive or negative perturbations to each property, i.e., if we choose $\{\text{pos}_x, \text{pos}_y, \text{color}\}$ as target properties, we will generate 3000 samples for training. The validation and test sets always have 1000 samples. For the 2D dataset, we generate 128×128 images for better visual quality that is not distorted due to artifacts caused by perturbations. We then normalize and clip the image features (RGB values) to be in $[-1, 1]$ range.

3D Shapes. For generating the 3D datasets we leverage `kubric` library (Greff et al., 2022) to obtain realistic scenes which we can highly customize. Objects sit on a floor, a perspective camera is situated at $(2.5, 0, 3.0)$ and looks at $(0.0, 0.0, 0.0)$. Directional light illuminates the scene from $(1.0, 0.0, 1.0)$ towards the center. The set of possible target properties are similar to 2D shapes, and the range of properties in which each object is spawned is as follows:

- $\text{pos}_x, \text{pos}_y$ are generated randomly and uniformly in the $[-1.5, 1.5]$ and $[-1.0, 1.0]$ ranges respectively, such that no two objects overlap and no object falls even partially outside the boundaries. Note however, by overlap we mean that objects are spawned such that they do not mutually fill a volume in the 3D space, and we only prevent such occurrences, but we *do* allow *occlusions* from the perspective of the camera, which adds to the complexity of this synthetic dataset. pos_z is never a disentanglement target property and is always set such that objects sit on the floor (except when rotated). The reason for fixing the z coordinate is that any possible perturbation to the 3D coordinates is always going to be interpreted on a 2D scene that is observed by a camera that is placed somewhere above the floor. Therefore, introducing a third coordinate in the DGP and target properties has no point. Positional coordinates can be perturbed by offsets in $[-0.3, 0.3]$ range.
- color is similarly parameterized by a scalar in HSV format as in 2D.
- Shape can be any of $\{\text{sphere}, \text{cube}, \text{cylinder}, \text{cone}\}$. Again, since the effects of perturbations need to be visible in the pixel space, we will not use spheres if the rotation angle ϕ is a property we aim to disentangle with perturbations, as sphere rotations do not reflect in the pixel space. For rotation in the 3D space we choose the z-axis as the axis of rotation so that angular perturbations are maximally visible (w.r.t. the perspective camera's location).
- Size is a continuous property in the range $[0.3, 0.7]$ and can be perturbed by offsets in $[-0.15, 0.15]$ range.
- Rotation angle follows the convention of 2D shapes DGP, except that the rotations are around the z-axis for better visual quality.

Since `kubric` already generates high fidelity images, we use 64×64 images to lower the computational burden of SA-MESH autoencoder. The number of samples is always fixed at $20,000$ regardless of the target properties since the 3D dataset is more complex. We use a similar transformation as in 2D, and normalize and clip the image features (RGB values) to be in $[-1, 1]$ range.