# Generalization and Robustness Implications in Object-Centric Learning

**Andrea Dittadi** [1 2] **Samuele Papa** [1] **Michele De Vita** [1]
**Bernhard Schölkopf** [2] **Ole Winther** [1 3 4] **Francesco Locatello** [5]

## Abstract

The idea behind object-centric representation learning is that natural scenes can better be modeled as compositions of objects and their relations as opposed to distributed representations. This inductive bias can be injected into neural networks to potentially improve systematic generalization and performance of downstream tasks in scenes with multiple objects. In this paper, we train state-of-the-art unsupervised models on five common multi-object datasets and evaluate segmentation metrics and downstream object property prediction. In addition, we study generalization and robustness by investigating the settings where either a single object is out of distribution—e.g., having an unseen color, texture, or shape—or global properties of the scene are altered—e.g., by occlusions, cropping, or increasing the number of objects. From our experimental study, we find object-centric representations to be useful for downstream tasks and generally robust to most distribution shifts affecting objects. However, when the distribution shift affects the input in a less structured manner, robustness in terms of segmentation and downstream task performance may vary significantly across models and distribution shifts.

## 1. Introduction

In object-centric representation learning, we make the assumption that visual scenes are composed of multiple entities or objects that interact with each other, and exploit this compositional property as inductive bias for neural networks. Informally, the goal is to find transformations $r$ of the data $\mathbf{x}$ into a *set* of vector representations $r(\mathbf{x}) = \{\mathbf{z}_k\}$ each corresponding to an individual object, without supervision (Burgess et al., 2019; Chen et al., 2020; Crawford & Pineau, 2019; Engelcke et al., 2020b; Eslami et al., 2016; Greff et al., 2017; 2019; Gregor et al., 2015; Kosiorek et al., 2018; Lin et al., 2020b; Locatello et al., 2020; Mnih et al., 2014; Weis et al., 2020; Yuan et al., 2019). Relying on this inductive bias, object-centric representations are conjectured to be more robust than distributed representations, and to enable the systematic generalization typical of symbolic systems while retaining the expressiveness of connectionist approaches (Bengio et al., 2013; Greff et al., 2020; Lake et al., 2017; Schölkopf et al., 2021). Grounding for these claims comes mostly from cognitive psychology and neuroscience (Spelke, 1990; Téglás et al., 2011; Wagemans, 2015). E.g., infants learn about the physical properties of objects as entities that behave consistently over time (Baillargeon et al., 1985; Spelke & Kinzler, 2007) and are able to re-apply their knowledge to new scenarios involving previously unseen objects (Dehaene, 2020). Similarly, in complex machine learning tasks like physical modelling and reinforcement learning, it is common to train from the internal representation of a simulator (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2020) or of a game engine (Berner et al., 2019; Vinyals et al., 2019) rather than from raw pixels, as more abstract representations facilitate learning. Finally, learning to represent objects separately is a crucial step towards learning causal models of the data from high-dimensional observations, as objects can be interpreted as causal variables that can be manipulated independently (Schölkopf et al., 2021). Such causal models are believed to be crucial for human-level generalization (Pearl, 2009; Peters et al., 2017), but traditional causality research assumes causal variables to be given rather than learned (Schölkopf, 2019).

As object-centric learning developed recently as a subfield of representation learning, we identify three key hypotheses and design systematic experiments to test them. (1) *The unsupervised learning of objects as pretraining task is useful for downstream tasks.* Besides learning to separate objects without supervision, current approaches are expected to separately represent information about each object's properties, so that the representations can be useful for arbitrary downstream tasks. (2) *In object-centric models, distribution shifts affecting a single object do not affect the representations of other objects.* If objects are to be represented independently

[1]Technical University of Denmark [2]Max Planck Institute for Intelligent Systems, Tübingen, Germany [3]University of Copenhagen [4]Rigshospitalet, Copenhagen University Hospital [5]Amazon. Correspondence to: Andrea Dittadi <adit@dtu.dk>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

of each other to act as compositional building blocks for higher-level cognition (Greff et al., 2020), changes to one object in the input should not affect the representation of the unchanged objects. This should hold even if the change leads to an object being out of distribution (OOD). (3) *Object-centric models are generally robust to distribution shifts, even if they affect global properties of the scene.* Even if the whole scene is OOD—e.g., if it contains more objects than in the training set—object-centric approaches should be robust thanks to their inductive bias.

In this paper, we systematically investigate these three concrete hypotheses by re-implementing popular unsupervised object discovery approaches and testing them on five multi-object datasets.[1] We find that: (1) Object-centric models achieve good downstream performance on property prediction tasks. We also observe a strong correlation between segmentation metrics, reconstruction error, and downstream property prediction performance, suggesting potential model selection strategies. (2) If a single object is out of distribution, the overall segmentation performance is not strongly impacted. Remarkably, the downstream prediction of in-distribution (ID) objects is mostly unaffected. (3) Under more global distribution shifts, the ability to separate objects depends significantly on the model and shift at hand, and downstream performance may be severely affected.

As an additional contribution, we provide a library[2] for benchmarking object-centric representation learning, which can be extended with more datasets, methods, and evaluation tasks. We hope this will foster further progress in the learning and evaluation of object-centric representations.

## 2. Study design and hypotheses

**Problem definition:** Vanilla deep learning architectures learn distributed representations that do not capture the compositional properties of natural scenes—see, e.g., the "superposition catastrophe" (Bowers et al., 2014; Greff et al., 2020; Von Der Malsburg, 1986). Even in disentangled representation learning (Chen et al., 2018; Eastwood & Williams, 2018; Higgins et al., 2017; Kim & Mnih, 2018; Kumar et al., 2018; Ridgeway & Mozer, 2018), factors of variations are encoded in a vector representation that is the output of a standard CNN encoder. This introduces an unnatural ordering of the objects in the scene and fails to capture its compositional structure in terms of objects. Formally defining objects is challenging (Greff et al., 2020) and there is no consensus even outside of machine learning (Green, 2019; Smith, 1998). Greff et al. (2020) put forth three properties for object-centric representations: *separation*, i.e., object features in the set of vectors $r(\mathbf{x})$ do not interact with each other, and each object is individually captured in a single element of $r(\mathbf{x})$; *common format*, i.e., each element of $r(\mathbf{x})$ shares the same representational format; and *disentanglement*, i.e., each element of $r(\mathbf{x})$ is represented in a disentangled format that exposes the factors of variation. In this paper, we consider representations $r(\mathbf{x})$ that are sets of vectors with each element sharing the representational format. We take a pragmatic perspective and focus on two clear desiderata for object-centric approaches:

**Desideratum 1:** *Object embodiment.* The representation should contain information about the object's location and its embodiment in the scene. As we focus on unsupervised object discovery, this translates to segmentation masks. This is related to separation and common format, as the decoder is applied to the elements of $r(\mathbf{x})$ with shared parameters.

**Desideratum 2:** *Informativeness of the representation.* Instead of learning disentangled representations of objects, which is challenging even in single-object scenarios (Locatello et al., 2019b), we want the representation to contain useful information for downstream tasks, not necessarily in a disentangled format. We define objects through their properties as annotated in the datasets we consider, and predict these properties from the representations. Note that this may not be the only way to define objects (e.g., defining faces and edges as objects and deducing shapes as composition of those). The fact that existing models learn informative representations is our first hypothesis (see below).

**Design principle:** These desiderata offer well-defined quantitative evaluations for object-centric approaches and we want to understand the implications of learning such representations. To this end, we train four different state-of-the-art methods on five datasets, taking hyperparameter configurations from the respective publications and adapting them to improve performance when necessary. Assuming these models succeeded in learning an object-centric representation, we investigate the following hypotheses.

**Hypothesis 1:** *The unsupervised learning of objects as pretraining task is useful for downstream tasks.* Existing empirical evaluations largely focus on *Desideratum 1* and measure performance in terms of segmentation metrics. The hope, however, is that the learned representation would be useful for other downstream tasks besides segmentation (*Desideratum 2*). We test this hypothesis by training small downstream models on the frozen object-representations to predict the object properties. We match the predictions to the ground-truth properties with the Hungarian algorithm (Kuhn, 1955) following Locatello et al. (2020).

**Hypothesis 2:** *In object-centric models, distribution shifts affecting a single object do not affect the representations of other objects.* A change in the properties of one object in

[1] Training and evaluating all the models for the main study requires approximately 1.44 GPU years on NVIDIA V100.

[2] https://github.com/addtt/object-centric-library

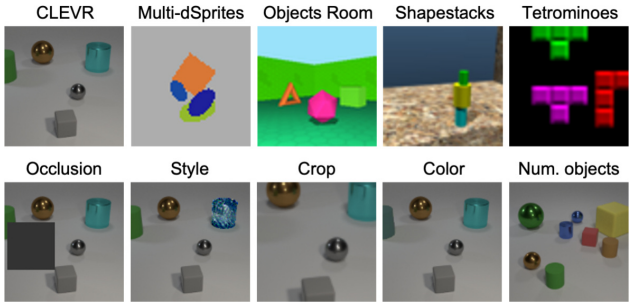


Figure 1: **Top**: examples from the five datasets in this study. **Bottom**: distribution shifts in CLEVR.

the input should not affect the representation of the other objects. Even OOD objects with previously unseen properties should be segmented correctly by a network that learned the notion of objects (Greff et al., 2020; Schölkopf et al., 2021). We test this hypothesis by (1) evaluating the segmentation of the scene after the distribution shift, and (2) training downstream models to predict object properties, and evaluating them on representations extracted from scenes with one OOD object. More specifically, we test changes in the shape, color, or texture of one object.

**Hypothesis 3:** *Object-centric models are generally robust to distribution shifts, even if they affect global properties of the scene.* Early evidence (Romijnders et al., 2021) points to the conjecture that learning object-centric representations biases the network towards learning more robust representations of the overall scene. Intuitively, the notion of objects is an additional inductive bias for the network to exploit to maintain accurate predictions if simple global properties of the scene are altered. We test this hypothesis by training downstream models to predict object properties, and evaluating them on representations of scenes with OOD global properties. In this case, we test robustness by cropping, introducing occlusions, and increasing the number of objects.

## 3. Experimental setup

Here we provide an overview of our experimental setup. After introducing the relevant models and datasets, we outline the evaluation protocols for segmentation accuracy (Desideratum 1) and downstream task performance (Desideratum 2). Then, we discuss the distribution shifts that we use to test robustness—the aforementioned evaluations are repeated once again under these distribution shifts. We conclude with a discussion on the limitations of this study.

**Models and datasets.** We implement four state-of-the-art object-centric models—MONet (Burgess et al., 2019), GENESIS (Engelcke et al., 2020b), Slot Attention (Locatello et al., 2020), and SPACE (Lin et al., 2020b)—as well as vanilla variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) as baselines for distributed representations. We use one VAE variant with a broadcast decoder (Watters et al., 2019) and one with a regular convolutional decoder. See Appendix A for an overview of the models with implementation details. We then collect five popular multi-object datasets: *Multi-dSprites*, *Objects Room*, and *Tetrominoes* from DeepMind's Multi-Object Datasets collection (Kabra et al., 2019), *CLEVR* (Johnson et al., 2017), and *Shapestacks* (Groth et al., 2018). The datasets are shown in Fig. 1 (top row) and described in detail in Appendix B. For each dataset, we define train, validation, and test splits. The test splits, which always contain at least 2000 images, are exclusively used for evaluation. We train each model on all datasets, using 10 random seeds for object-centric models and 5 for each VAE variant, resulting in 250 models in total.

**Metrics.** We evaluate the segmentation accuracy of object-centric models with the Adjusted Rand Index (ARI) (Hubert & Arabi, 1985), Segmentation Covering (SC) (Arbelaez et al., 2010), and mean Segmentation Covering (mSC) (Engelcke et al., 2020b). For all models, we additionally evaluate reconstruction quality via the mean squared error (MSE). Appendix C.1 includes detailed definitions of these metrics.

**Downstream property prediction.** We evaluate object-centric representations by training downstream models to predict ground-truth object properties from the representations. More specifically, exploiting the fact that object slots share a common representational format, a single downstream model $f$ can be used to predict the properties of each object independently: for each slot representation $\mathbf{z}_k$ we predict a vector of object properties $\hat{\mathbf{y}}_k = f(\mathbf{z}_k)$. As in previous work on object property prediction (Locatello et al., 2020), each model simultaneously predicts all properties of an object. For learning, we use the cross-entropy loss for categorical properties and MSE for numerical properties, and denote by $\ell(\hat{\mathbf{y}}_k, \mathbf{y}_m)$ the overall loss for a single object, where $\mathbf{y}_m$ are its ground-truth properties. Here $k \in \{1, \dots, K\}$ and $m \in \{1, \dots, M\}$ with $K$ the number of slots and $M$ the number of objects. In order to optimize the downstream models, the vector $\hat{\mathbf{y}}_k$ (the properties predicted from the $k$th representational slot) needs to be matched to the ground-truth properties $\mathbf{y}_m$ of the $m$th object. This is done by computing a $M \times K$ matrix of *matching losses* for each slot–object pair, and then solving the assignment problem using the Hungarian algorithm (Kuhn, 1955) to minimize the total matching loss, which is the sum of $\min(M, K)$ terms from the loss matrix. As matching loss we use either the negative cosine similarity between predicted and ground-truth masks (as in Greff et al. (2019)), or the downstream loss $\ell(\hat{\mathbf{y}}_k, \mathbf{y}_m)$ itself (as in Locatello et al. (2020)). In the following, we will refer to these strategies as *mask matching* and *loss matching*, respectively. For property prediction, we use 4 different downstream models: a linear model, and MLPs with up to 3 hidden layers of size 256 each. Given a pretrained object-centric model, we train each downstream model on the representations of 10 000

images. The downstream models are then tested on 2000 held-out images from the test set, which may exhibit distribution shifts as discussed below. Further details on this evaluation are provided in Appendix C.2

**Evaluating distributed representations.** Since in non-slot-based models, such as classical VAEs, the representations of the single objects are not readily available, matching representations to objects for downstream property prediction is not trivial. Although this is an inherent limitation of distributed representations, we are nevertheless interested in evaluating their usefulness. Using the matching framework presented above, we require the downstream model $f$ to output the predicted properties of *all* objects, and then match these with the true object properties to evaluate prediction quality. Our downstream model in this case will thus take as input the entire representation $\mathbf{z} = r(\mathbf{x})$ (which is now a single vector rather than a set of vectors) and output the predictions for all objects together as a vector $f(\mathbf{z})$. Finally, we split $f(\mathbf{z})$ into $K$ vectors $\{\hat{\mathbf{y}}_k\}_{k=1}^K$, where $K$ loosely corresponds to the number of slots in object-centric models. At this point, we can compute the loss $\ell(\hat{\mathbf{y}}_k, \mathbf{y}_m)$ for each pair, as usual. We now consider two matching strategies: As before, *loss matching* simply defines the matching loss of a slot–object pair as the prediction loss itself. In the *deterministic matching* strategy, following Greff et al. (2019), we lexicographically sort objects according to a canonical order of object properties. Calling $\pi$ the permutation that defines this sorting, the $k$th slot is deterministically matched with the $m$th object, where $m = \pi^{-1}(k)$.

**Baselines.** To correctly assess performance on downstream tasks, it is fundamental to compare with sensible baselines. Here we consider as baseline the best performance that can be achieved by a downstream model that outputs a constant vector that does not depend on the image. When predicting properties independently for each object (in slot-based models), the optimal solution is to predict the mean of continuous properties and the majority class for categorical ones. When using deterministic matching in the distributed case, the downstream model can exploit the predefined total order to predict more accurately than random guessing even without using information from the input (this effect is non-negligible only for the properties that are most significant in the order). Finally, in a few cases, loss matching for distributed representations can be significantly better than deterministic matching.[3] For simplicity, for both matching strategies in the distributed case, we directly learn a vector $\hat{\mathbf{y}}$ by gradient descent to minimize the prediction loss. As this depends on random initialization and optimization dynamics, we repeat this for 10 random seeds and report error bars in the plots.

[3]Intuitively, a (constant) diverse set of uninformed predictions $\{\hat{\mathbf{y}}_k\}$ might be sufficient for the matching algorithm to find suitable enough objects for most predictions.

**Distribution shifts.** We test the robustness of the learned representations under two classes of distribution shifts: one where one object goes OOD, and one where global properties of the scene are changed. All such distribution shifts occur at test time, i.e., the unsupervised models are always trained on the original datasets. To evaluate generalization to distribution shifts affecting a *single object*, we systematically induce changes in the color, shape, and texture of objects. To change color, we apply a random color shift to one random object in the scene, using the available masks (we do not do this in Multi-dSprites, as the training distribution covers the entire RGB color space). To test robustness to unseen textures, we apply neural style transfer (Gatys et al., 2016) to one random object in each scene, using *The Great Wave off Kanagawa* as style image. When either a new color or a new texture is introduced, prediction of material (in CLEVR only) and color is not performed. To introduce a new shape, we select images from Multi-dSprites that have at most 4 objects (in general, they have up to 5), and add a randomly colored triangle, in a random position, at a random depth in the object stack. In this case, shape prediction does not apply. Finally, to test robustness to *global* changes in the scene, we change the number of objects (in CLEVR only), introduce occlusions (a gray square at a random location), or crop images at the center and restore their original size via bilinear interpolation. See Fig. 1 for examples, and Appendix C.3 for further details.

**Limitations of this study.** While we aim to conduct a sound and informative experimental study to answer the research questions from Section 3, inevitably there are limitations regarding datasets, models, and evaluations. Although the datasets considered here vary significantly in complexity and visual properties, they all consist of synthetic images where object properties are independent of each other and independent between objects. Regarding object-centric models, we only focus on autoencoder-based approaches that model a scene as a mixture of components. As official implementations are not always available, and none of the methods in this work has been applied to all the datasets considered here, we re-implement these methods and choose hyperparameters following a best-effort approach. Finally, we only consider the downstream task of object property prediction, and assess generalization using only a few representative single-object and global distribution shifts.

## 4. Results

In this section, we highlight our findings with plots that are representative of our main results. The full experimental results are presented in Appendix D. In Section 4.1 we focus on the different evaluation metrics and the performance we obtained re-training the methods considered in this study. We then focus on our three hypotheses in Sections 4.2 to 4.4.

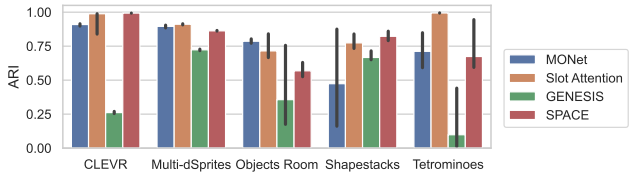


Figure 2: ARI of all models and datasets on 2000 test images. Medians and 95% confidence intervals with 10 seeds.

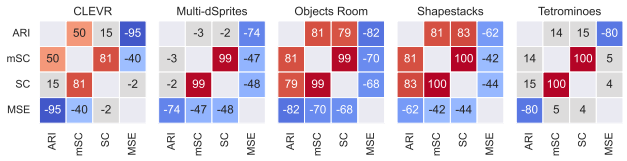


Figure 3: Spearman rank correlations between evaluation metrics across models and random seeds (color-coded only when p<0.05).

### 4.1. Learning and evaluating object discovery

Since all methods included in our study were originally evaluated only on a subset of the datasets and metrics considered here, we first test how well these models perform.

Fig. 2 shows the segmentation performance of the models in terms of ARI across models, datasets, and random seeds. Fig. 12 in Appendix D provides an overview of the reconstruction MSE and all segmentation metrics. Although these results are in line with published work, we observe substantial differences in the ranking between models depending on the metric. This indicates that, in practice, these metrics are not equivalent for measuring object discovery.

This is confirmed in Fig. 3, which shows rank correlations between metrics on different datasets (aggregating over different models). We also observe a strong negative correlation between ARI and MSE across models and datasets, suggesting that models that learn to more accurately reconstruct the input tend to better segment objects according to the ARI score. This trend is less consistent for the other segmentation metrics, as MSE significantly correlates with mSC in only three datasets (Multi-dSprites, Objects Room, and CLEVR), and with SC in two (Multi-dSprites and Objects Room). SC and mSC measure very similar segmentation notions and therefore are significantly correlated in all datasets, although to a varying extent. However, they correlate with ARI only on two and three datasets, respectively (the same datasets where they correlate with the MSE).

**Summary:** We observe strong differences in performance and ranking between the models depending on the evaluation metric. In the tested datasets, we find that the ARI, which requires ground-truth segmentation masks to compute, correlates particularly well with the MSE, which is unsupervised and provides training signal.

### 4.2. Usefulness for downstream tasks (Hypothesis 1)

To test Hypothesis 1, we first evaluate whether frozen object-centric representations can be used to train downstream models measuring Desideratum 2 from Section 2. As discussed in Section 3, this type of downstream task requires matching the true object properties with the predictions of the downstream model. In the following, we will only present results obtained with *loss matching*, and show results for other matching strategies in Appendix D.

Fig. 4 shows downstream prediction performance on all datasets and models, when the downstream model is a single-layer MLP. Although results vary across datasets and models, accurate prediction of object properties seems to be possible in most of the scenarios considered here. Fig. 14 in Appendix D shows similar results when using a linear model or MLPs with up to 3 hidden layers.

In Fig. 4, we also compare the downstream prediction performance from object-centric and distributed representations. We observe that VAE representations tend to achieve lower scores in downstream prediction, although not always by a large margin. In particular, color and size in CLEVR and color in Tetrominoes are predicted relatively well, and significantly better than the baseline. On the other hand, in many cases where VAE representations perform well, they have in fact a considerable advantage if we take the baselines into account (scale in Multi-dSprites, color in Shapestacks, x and y in CLEVR, Multi-dSprites, and Tetrominoes). Moreover, performance from distributed representations often does not improve significantly when using a larger downstream model (see Fig. 15). In conclusion, although the two classes of representations are difficult to compare on this task, these results suggest that the quantities of interest are present in the VAE representations, but they appear to be less explicit and less easily usable.

Finally, we investigate the relationship between downstream performance and evaluation metrics. Fig. 5 shows the Spearman rank correlation of the segmentation and reconstruction metrics with the test performance of downstream predictors. For all datasets and object properties, downstream performance is strongly correlated with the ARI. On the other hand, SC and mSC exhibit inconsistent trends across datasets. Models that correctly separate objects according to the ARI are therefore useful for downstream object property prediction, confirming Hypothesis 1. Downstream prediction performance is also significantly correlated with the reconstruction MSE in all datasets. This is not particularly surprising, since the representation of a model that cannot properly reconstruct the input might not contain the information necessary for property prediction. However, the correlation is generally stronger with the ARI than with the MSE, suggesting that having a notion of objects is more important for downstream tasks than reconstruction accuracy. This is consistent with the findings by Papa et al. (2022), where the ARI still correlates strongly with downstream

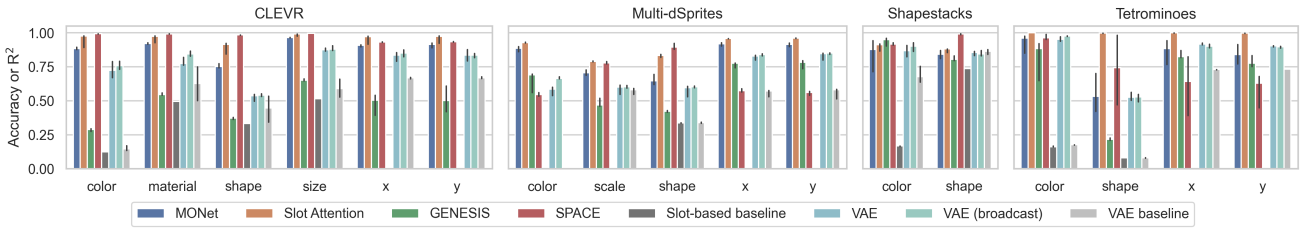


Figure 4: Comparison of downstream property prediction performance for object-centric (slot-based) and distributed (VAE) representations, using an MLP with one hidden layer as downstream model. The metric is accuracy for categorical properties or $R^2$ for numerical ones. The baselines in gray indicate the best performance that can be achieved by a model that outputs a constant vector that does not depend on the input. The bars show medians and 95% confidence intervals with 10 random seeds.

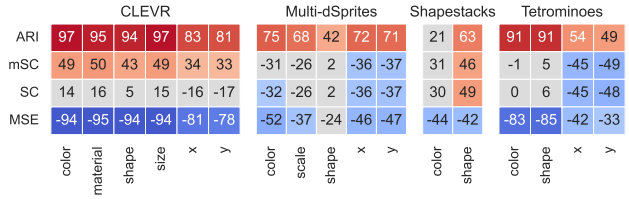


Figure 5: Spearman rank correlations between evaluation metrics and downstream performance with an MLP. The correlations are color-coded only when p<0.05.

performance when objects have complex textures, while the MSE does not. When segmentation masks are available for validation, ARI should therefore be the preferred metric to select useful representations for downstream tasks. Fig. 16 in Appendix D shows analogous results for mask matching and for the three other downstream models—these results are broadly similar, except that correlations with ARI tend to be stronger when using mask matching (perhaps unsurprisingly) or larger downstream models.

**Summary:** Models that accurately segment objects allow for good downstream prediction performance. Despite often having an advantage, distributed representations generally perform worse, but not always significantly: the information is present but less easily accessible. The ARI is consistently correlated with downstream performance, and is therefore useful for model selection when masks are available. The MSE can be a practical unsupervised alternative on these datasets, but it may be less robust on complex textures.

### 4.3. Generalization with one OOD object (Hypothesis 2)

To test Hypothesis 2, we construct settings where a single object is OOD and the others are ID. We change the object style with neural style transfer, change the color of one object at random (only in CLEVR, Tetrominoes, and Shapestacks), or introduce a new shape (only in Multi-dSprites). The unsupervised models are always trained on the original datasets. Then we train downstream models to predict the object properties from the learned representations. We consider two scenarios for this task: (1) train the predictors on the original datasets and test them on the variants with a modified object, (2) train and test the predictors on each variant. In both cases, we test the predictors on representations that might be inaccurate, because the representation function (encoder) is OOD. However, since in case (2) the downstream model is *trained* under distribution shift, this experiment quantifies the extent to which the representation can still be used by a downstream task that is allowed to adapt to the shift—although the representation might no longer represent objects faithfully, it could still contain useful information.

For Desideratum 1, we observe in Fig. 6 that the models are generally robust to distribution shifts affecting a single object. Introducing a new color or a new shape typically does not affect segmentation quality (but note Slot Attention on Tetrominoes), while changing the texture of an object via neural style transfer leads to a moderate drop in ARI in some cases. In Fig. 17 (Appendix D) we observe that SC and mSC show a compatible but less pronounced trend, while the MSE more closely mirrors the ARI. We conclude that the encoder is still partially able to separate objects when one object undergoes a distribution shift at test time.

For Desideratum 2, we observe in Fig. 7 (left) that property prediction performance for objects that underwent distribution shifts (color, shape, or texture) is often significantly worse than in the original dataset, whereas the prediction of ID objects is largely unaffected. This is in agreement with Hypothesis 2: changes to one object do not affect the representation of other objects, even when these objects are OOD. Extensive results, including further splits and all downstream models, are shown in Fig. 19 in Appendix D. On the right plot in Fig. 7, we observe that retraining the downstream models after the distribution shifts does not lead to significant improvements. This suggests that the shifts introduced here negatively affect not only the downstream model, but also the representation itself. This result also holds with different downstream models and with mask matching (see Figs. 20 and 22). While in principle we observe a similar trend for VAEs (see e.g. Figs. 27 and 28 in Appendix D), their performance is often too close to the respective baseline (Fig. 4) for a definitive conclusion to be drawn.

**Summary:** The models are generally robust to distribution

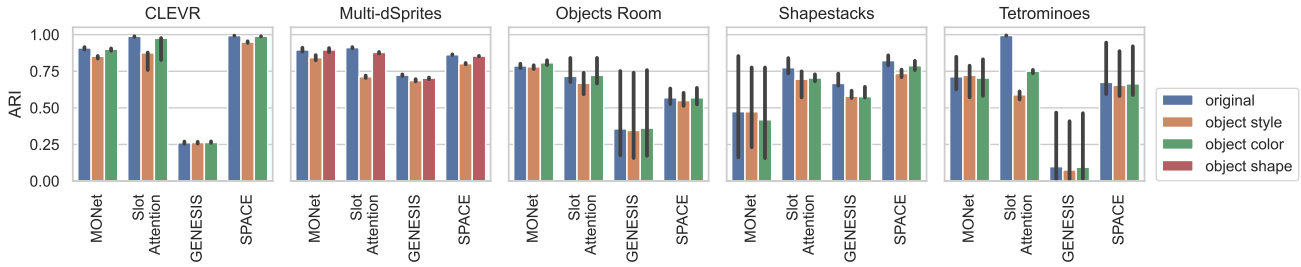


Figure 6: Effect of single-object distribution shifts on the ARI. Medians and 95% confidence intervals with 10 random seeds.

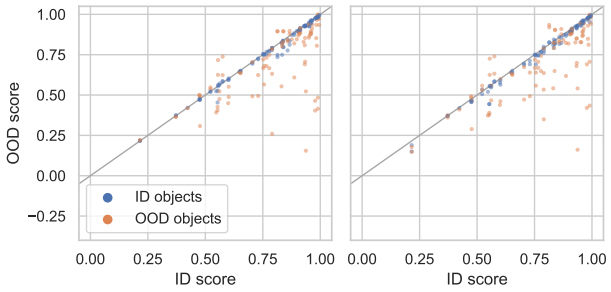


Figure 7: **ID vs OOD downstream performance** with single-object distribution shifts. All datasets, models, and object properties are shown. Metrics: accuracy for categorical attributes, $R^2$ for numerical attributes. The downstream model (an MLP with one hidden layer) is tested zero-shot out-of-distribution (left) or retrained after the distribution shift has occurred (right).

shifts affecting a single object. Downstream prediction is largely unaffected for ID objects, but may be severely affected for OOD objects. Finally, there seems to be no clear benefit in retraining downstream models after the shifts, indicating that the deteriorated representations cannot easily be adjusted *post hoc*.

### 4.4. Robustness to global shifts (Hypothesis 3)

Finally, we investigate the robustness of object-centric models to transformations changing the global properties of a scene at test time. Here, we consider variants of the datasets with occlusions, cropping, or more objects (only on CLEVR). We train downstream predictors on the original datasets and report their test performance on the dataset variants with global shifts. As before, we also report results of downstream models retrained on the OOD datasets.

For Desideratum 1, Fig. 8 shows that segmentation quality is generally only marginally affected by occlusion, but cropping often leads to a significant degradation. In CLEVR, the effect on the ARI of increasing the number of objects is comparable to the effect of occlusions, which suggests that learning about objects is useful for this type of systematic generalization. These trends persist when considering SC and mSC, but appear less pronounced and less consistent across datasets (see Fig. 18 in Appendix D for detailed results). As might be expected, when the number of objects is increased in CLEVR, the MSE increases more conspicuously for VAEs than for object-centric models (Fig. 18, bottom left), likely due to their explicit modeling of objects. However, Fig. 41 shows that VAEs may, in fact, generalize relatively well to an unseen number of objects, although not nearly as well as some object-centric models.

For Desideratum 2, we train a downstream model on the original dataset and test it under global distribution shifts. These shifts generally have a negative effect on downstream property prediction (Fig. 9, left), although this is comparable to the effect on OOD objects when only one object is OOD. This is in agreement with the observation made in Section 4.3 that these shifts negatively affect the representation, which is no longer accurate because the encoder is OOD (cf. the "OOD2" scenario in Dittadi et al. (2021)). When retraining the downstream models on the OOD datasets while keeping the representation frozen, the performance improves slightly but does not reach the corresponding results on the training distribution (Fig. 9, right), as in Section 4.3. These observations also hold for different downstream models and with mask matching (Figs. 23 to 26), as well as for distributed representations (see, e.g., Fig. 31) although with similar caveats as in Section 4.3.

**Summary:** The impact of global distribution shifts on the segmentation capability of object-centric models depends on the chosen shift; e.g., cropping consistently has a significant effect. Moreover, the usefulness for downstream tasks decreases substantially in many cases, and the performance of downstream prediction models cannot be satisfactorily recovered by retraining them.

## 5. Other related work

Recent years have seen a number of systematic studies on disentangled representations (Locatello et al., 2019a;b; Träuble et al., 2020; van Steenkiste et al., 2019), some of which focusing on their effect on generalization (Dittadi et al., 2021; Esmaeili et al., 2019; Gondal et al., 2019; Montero et al., 2021; Träuble et al., 2022). In the context of object-centric learning, Engelcke et al. (2020a) investigate their

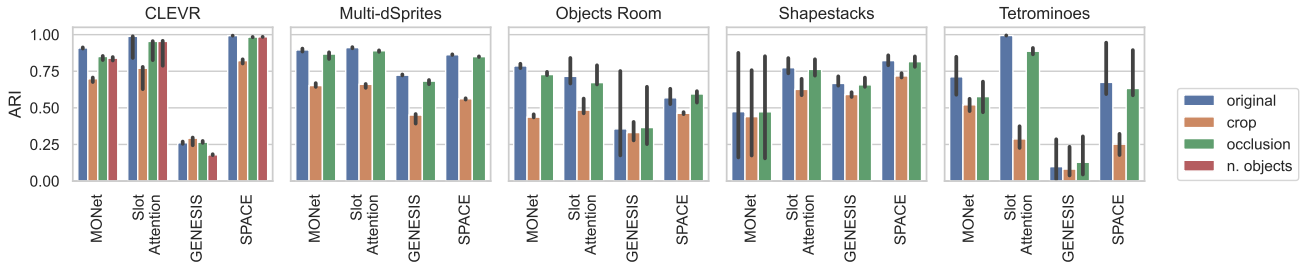


Figure 8: Effect of distribution shifts on global scene properties on the ARI. Medians and 95% confidence intervals with 10 seeds.

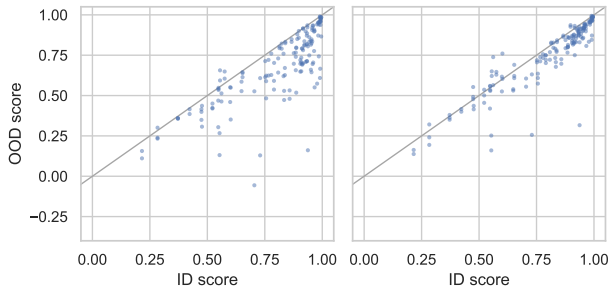


Figure 9: **ID vs OOD downstream performance** with global distribution shifts. All datasets, models, and object properties are shown. Metrics: accuracy for categorical attributes, $R^2$ for numerical attributes. The downstream model (an MLP with one hidden layer) is tested zero-shot out-of-distribution (left) or retrained after the distribution shift has occurred (right).

reconstruction bottlenecks to understand how these models can separate objects from the input in an unsupervised manner. In contrast, we specifically test some key implications of learning object-centric representations.

Slot-based object-centric models can be classified according to their approach to separating the objects at a representational level (Greff et al., 2020). In models that use *instance slots* (Chen et al., 2019; Goyal et al., 2019; Greff et al., 2016; 2017; 2019; Huang et al., 2020; Kipf et al., 2019; 2021; Le Roux et al., 2011; Locatello et al., 2020; Löwe et al., 2020; Racah & Chandar, 2020; van Steenkiste et al., 2018; 2020; Yang et al., 2020), each slot is used to represent a different part of the input. This introduces a routing problem, because all slots are identical but they cannot all represent the same object, so a mechanism needs to be introduced to allow slots to communicate with each other. In models based on *sequential slots* (Burgess et al., 2019; Engelcke et al., 2020b; 2021; Eslami et al., 2016; Kosiorek et al., 2018; Kossen et al., 2019; Stelzner et al., 2019; von Kügelgen et al., 2020), the representational slots are computed in a sequential fashion, which solves the routing problem and allows to dynamically change the number of slots, but introduces dependencies between slots. In models based on *spatial slots* (Crawford & Pineau, 2019; 2020; Deng et al., 2021; Dittadi & Winther, 2019; Jiang et al., 2020; Lin et al., 2020a;b; Nash et al., 2017), a spatial coordinate is associated with each slot, introducing a dependency between slot and spatial location. In this work, we focus on four scene-mixture models as representative examples of approaches based on instance slots (Slot Attention), sequential slots (MONet and GENESIS), and spatial slots (SPACE).

## 6. Conclusions

In this paper, we identify three key hypotheses in object-centric representation learning: learning about objects is useful for downstream tasks, it facilitates strong generalization, and it improves overall robustness to distribution shifts. To investigate these hypotheses, we re-implement and systematically evaluate four state-of-the-art unsupervised object-centric learners on a suite of five common multi-object datasets. We find that object-centric representations are generally useful for downstream object property prediction, and downstream performance is strongly correlated with segmentation quality and reconstruction error. Regarding generalization, we observe that when a single object undergoes distribution shifts the overall segmentation quality and downstream performance for in-distribution objects is largely unaffected. Finally, we find that object-centric models can still relatively robustly separate objects even under global distribution shifts. However, this may depend on the specific shift, and downstream performance appears to be more severely affected.

An interesting avenue for future work is to continue our systematic investigation of object-centric learning on more complex data with diverse textures, as well as a wide range of more challenging downstream tasks. Furthermore, it would be interesting to compare object-centric and non-object-centric models more fairly: while learning about objects offers clear advantages, the full potential of distributed representations in this context is still not entirely clear, particularly when scaling up datasets and models. Finally, while we limit our study to unsupervised object discovery, it would be relevant to consider methods that leverage some form of supervision when learning about objects. We believe our benchmarking library will facilitate progress along these and related lines of research.

## Acknowledgements

We would like to thank Thomas Brox, Dominik Janzing, Sergio Hernan Garrido Mejia, Thomas Kipf, and Frederik Träuble for useful comments and discussions, and the anonymous reviewers for valuable feedback.

## References

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

Baillargeon, R., Spelke, E. S., and Wasserman, S. Object permanence in five-month-old infants. *Cognition*, 20(3): 191–208, 1985.

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., and kavukcuoglu, k. Interaction networks for learning about objects, relations and physics. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv:1912.06680*, 2019.

Bowers, J. S., Vankov, I. I., Damian, M. F., and Davis, C. J. Neural networks learn highly selective representations in order to overcome the superposition catastrophe. *Psychological review*, 121(2):248, 2014.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Chen, C., Deng, F., and Ahn, S. Learning to infer 3d object models from images. *arXiv preprint arXiv:2006.06130*, 2020.

Chen, M., Artières, T., and Denoyer, L. Unsupervised object segmentation by redrawing. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Chen, T. Q., Li, X., Grosse, R., and Duvenaud, D. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

Crawford, E. and Pineau, J. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3412–3420, 2019.

Crawford, E. and Pineau, J. Exploiting spatial invariance for scalable unsupervised object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3684–3692, 2020.

Dehaene, S. *How We Learn: Why Brains Learn Better than Any Machine ... for Now*. Viking, New York, 2020.

Deng, F., Zhi, Z., Lee, D., and Ahn, S. Generative scene graph networks. In *International Conference on Learning Representations*, 2021.

Dittadi, A. and Winther, O. LAVAE: Disentangling Location and Appearance. In *Neural Information Processing Systems (NeurIPS) Workshop on Perception as Generative Reasoning*, 2019.

Dittadi, A., Träuble, F., Locatello, F., Wüthrich, M., Agrawal, V., Winther, O., Bauer, S., and Schölkopf, B. On the Transfer of Disentangled Representations in Realistic Settings. In *International Conference on Learning Representations*, 2021.

Eastwood, C. and Williams, C. K. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

Engelcke, M., Jones, O. P., and Posner, I. Reconstruction bottlenecks in object-centric generative models. *arXiv preprint arXiv:2007.06245*, 2020a.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. In *ICLR 2020*, 2020b.

Engelcke, M., Parker Jones, O., and Posner, I. GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement. *arXiv preprint arXiv:2104.09958*, 2021.

Eslami, S. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Eslami, S. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

Esmaeili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J., and van de Meent, J.-W. Structured disentangled representations. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2525–2534. PMLR, 16–18 Apr 2019.

Fowlkes, E. B. and Mallows, C. L. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

Gatys, L., Ecker, A., and Bethge, M. A Neural Algorithm of Artistic Style. *Journal of Vision*, 16(12):326–326, August 2016. ISSN 1534-7362. doi: 10.1167/16.12.326. Publisher: The Association for Research in Vision and Ophthalmology.

Gondal, M. W., Wüthrich, M., Miladinović, D., Locatello, F., Breidt, M., Volchkov, V., Akpo, J., Bachem, O., Schölkopf, B., and Bauer, S. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In *Advances in Neural Information Processing Systems*, 2019.

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

Green, E. J. A theory of perceptual objects. *Philosophy and Phenomenological Research*, 99(3):663–693, 2019.

Greff, K., Rasmus, A., Berglund, M., Hao, T., Valpola, H., and Schmidhuber, J. Tagger: Deep Unsupervised Perceptual Grouping. *Advances in Neural Information Processing Systems*, 29, 2016.

Greff, K., van Steenkiste, S., and Schmidhuber, J. Neural expectation maximization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6694–6704, 2017.

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

Greff, K., van Steenkiste, S., and Schmidhuber, J. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pp. 1462–1471. PMLR, 2015.

Groth, O., Fuchs, F. B., Posner, I., and Vedaldi, A. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 702–717, 2018.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Huang, Q., He, H., Singh, A., Zhang, Y., Lim, S. N., and Benson, A. R. Better set representations for relational reasoning. *Advances in Neural Information Processing Systems*, 33, 2020.

Hubert, L. and Arabi, P. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985. ISSN 1432-1343. doi: 10.1007/BF01908075.

Jacq, A. and Herring, W. Neural Transfer Using PyTorch, 2021. URL https://github.com/pytorch/tutorials.

Jiang, J., Janghorbani, S., Melo, G. D., and Ahn, S. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2020.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2901–2910, 2017.

Kabra, R., Burgess, C., Matthey, L., Kaufman, R. L., Greff, K., Reynolds, M., and Lerchner, A. Multi-object datasets, 2019. URL https://github.com/deepmind/multi-object-datasets/.

Kim, H. and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Kipf, T., van der Pol, E., and Welling, M. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*, 2021.

Kosiorek, A. R., Kim, H., Posner, I., and Teh, Y. W. Sequential attend, infer, repeat: Generative modelling of moving objects. *arXiv preprint arXiv:1806.01794*, 2018.

Kossen, J., Stelzner, K., Hussing, M., Voelcker, C., and Kersting, K. Structured object-aware physics prediction for video modeling and planning. *arXiv preprint arXiv:1910.02425*, 2019.

Kuhn, H. W. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Kumar, A., Sattigeri, P., and Balakrishnan, A. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

Le Roux, N., Heess, N., Shotton, J., and Winn, J. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

Lin, Z., Wu, Y.-F., Peri, S., Fu, B., Jiang, J., and Ahn, S. Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pp. 6140–6149. PMLR, 2020a.

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition. In *ICLR 2020*, 2020b.

Locatello, F., Abbati, G., Rainforth, T., Bauer, S., Schölkopf, B., and Bachem, O. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems*, pp. 14611–14624, 2019a.

Locatello, F., Bauer, S., Lucic, M., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, 2019b.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

Löwe, S., Greff, K., Jonschkowski, R., Dosovitskiy, A., and Kipf, T. Learning object-centric video models by contrasting sets. *arXiv preprint arXiv:2011.10287*, 2020.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset, 2017. URL https://github.com/deepmind/dsprites-dataset/.

Mnih, V., Heess, N., Graves, A., et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

Montero, M. L., Ludwig, C. J., Costa, R. P., Malhotra, G., and Bowers, J. The role of disentanglement in generalisation. In *International Conference on Learning Representations*, 2021.

Nash, C., Eslami, S. A., Burgess, C., Higgins, I., Zoran, D., Weber, T., and Battaglia, P. The multi-entity variational autoencoder. In *Neural Information Processing Systems (NeurIPS) Workshop on Learning Disentangled Representations: from Perception to Control*, 2017.

Papa, S., Winther, O., and Dittadi, A. Inductive biases for object-centric representations of complex textures. *arXiv preprint arXiv:2204.08479*, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Pearl, J. *Causality*. Cambridge University Press, 2009.

Peters, J., Janzing, D., and Schölkopf, B. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Racah, E. and Chandar, S. Slot contrastive networks: A contrastive approach for representing objects. *arXiv preprint arXiv:2007.09294*, 2020.

Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

Rezende, D. J. and Viola, F. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Ridgeway, K. and Mozer, M. C. Learning deep disentangled embeddings with the f-statistic loss. In *Advances in Neural Information Processing Systems*, 2018.

Romijnders, R., Mahendran, A., Tschannen, M., Djolonga, J., Ritter, M., Houlsby, N., and Lucic, M. Representation learning from videos in-the-wild: An object-centric approach. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 177–187, 2021.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to Simulate Complex Physics with Graph Networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, November 2020.

Schölkopf, B. Causality for machine learning. *arXiv preprint arXiv:1911.10500*, 2019.

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

Smith, B. C. *On the origin of objects*. MIT Press, 1998.

Spelke, E. S. Principles of object perception. *Cognitive Science*, 14:29–56, 1990. ISSN 0364-0213. doi: 10.1016/0364-0213(90)90025-R.

Spelke, E. S. and Kinzler, K. D. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

Stelzner, K., Peharz, R., and Kersting, K. Faster Attend-Infer-Repeat with Tractable Probabilistic Models. In Chaudhuri, K. and Salakhudinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5966–5975. PMLR, June 2019.

Téglás, E., Vul, E., Girotto, V., Gonzalez, M., Tenenbaum, J. B., and Bonatti, L. L. Pure Reasoning in 12-Month-Old Infants as Probabilistic Inference. *Science*, 332:1054–1059, May 2011. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1196404.

Träuble, F., Dittadi, A., Wuthrich, M., Widmaier, F., Gehler, P. V., Winther, O., Locatello, F., Bachem, O., Schölkopf, B., and Bauer, S. The role of pretrained representations for the OOD generalization of RL agents. In *International Conference on Learning Representations*, 2022.

Träuble, F., Creager, E., Kilbertus, N., Locatello, F., Dittadi, A., Goyal, A., Schölkopf, B., and Bauer, S. On disentangled representations learned from correlated data. *arXiv preprint arXiv:2006.07886*, 2020.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

van Steenkiste, S., Locatello, F., Schmidhuber, J., and Bachem, O. Are disentangled representations helpful for abstract visual reasoning? *arXiv preprint arXiv:1905.12506*, 2019.

van Steenkiste, S., Kurach, K., Schmidhuber, J., and Gelly, S. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309–325, 2020.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, November 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1724-z.

Von Der Malsburg, C. Am i thinking assemblies? In *Brain theory*, pp. 161–176. Springer, 1986.

von Kügelgen, J., Ustyuzhaninov, I., Gehler, P., Bethge, M., and Schölkopf, B. Towards causal generative scene models via competition of experts. In *ICLR 2020 Workshops*, 2020.

Wagemans, J. *The Oxford handbook of perceptual organization*. Oxford Library of Psychology, 2015.

Wagner, S. and Wagner, D. *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

Watters, N., Matthey, L., Burgess, C. P., and Lerchner, A. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019.

Weis, M. A., Chitta, K., Sharma, Y., Brendel, W., Bethge, M., Geiger, A., and Ecker, A. S. Unmasking the inductive biases of unsupervised object representations for video sequences. *arXiv preprint arXiv:2006.07034*, 2020.

Yang, Y., Chen, Y., and Soatto, S. Learning to manipulate individual objects in an image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6558–6567, 2020.

Yuan, J., Li, B., and Xue, X. Generative modeling of infinite occluded objects for compositional scene representation. In *International Conference on Machine Learning*, pp. 7222–7231, 2019.

# A. Models

In this section, we give an informal overview of the models included in this study and provide details on the implementation and hyperparameter choices.

## A.1. Overview of the models

**MONet.** In MONet (Burgess et al., 2019), attention masks are computed by a recurrent segmentation network that takes as input the image and the current *scope*, which is the still unexplained portion of the image. For each slot, a variational autoencoder (the *component VAE*) encodes the full image and the current attention mask, and then decodes the latent representation to an image reconstruction and mask. The reconstructed images are combined using the *attention* masks (*not* the masks decoded by the component VAE) into the final reconstructed image. The reconstruction loss is the negative log-likelihood of a spatial Gaussian mixture model (GMM) with one component per slot, where each pixel is modeled independently. The overall training loss is a (weighted) sum of the reconstruction loss, the KL divergence of the component VAEs, and an additional mask reconstruction loss for the component VAEs.

**GENESIS.** Similarly to MONet, GENESIS (Engelcke et al., 2020b) models each image as a spatial GMM. The spatial dependencies between components are modeled by an autoregressive prior distribution over the latent variables that encode the mixing probabilities. From the image, an encoder and a recurrent network are used to compute the latent variables that are then decoded into the mixing probabilities. The mixing probabilities are pixel-wise and can be seen as attention masks for the image. Each of these is concatenated with the original image and used as input to the component VAE, which finds latent representations and reconstructs each scene component. These are combined using the mixing probabilities to obtain the reconstruction of the image. While in MONet the attention masks are computed by a deterministic segmentation network, GENESIS defines an autoregressive prior on latent codes that are decoded into attention masks. GENESIS is therefore a proper probabilistic generative model, and it is trained by maximizing a modification of the ELBO introduced by Rezende & Viola (2018), which adaptively trades off the likelihood and KL terms in the ELBO.

**Slot Attention.** As our focus is on the object discovery task, we use the autoencoder model proposed in the Slot Attention paper (Locatello et al., 2020). The encoder consists of a CNN followed by the Slot Attention module, which maps the feature map to a set of slots through an iterative refinement process. At each iteration, dot-product attention is computed with the input vectors as keys and the current slot vectors as queries. The attention weights are then normalized over the slots, introducing competition between the slots to explain the input. Each slot is then updated using a GRU that takes as inputs the current slot vectors and the normalized attention vectors. After the refinement steps, the slot vectors are decoded into the appearance and mask of each object, which are then combined to reconstruct the entire image. The model is optimized by minimizing the MSE reconstruction loss. While MONet and GENESIS use sequential slots to represent objects, Slot Attention employs instance slots.

**SPACE.** Spatially Parallel Attention and Component Extraction (SPACE) (Lin et al., 2020b) combines the approaches of scene-mixture models and spatial attention models. The foreground objects are segregated using bounding boxes computed through a parallel spatial attention process. The parallelism allows for a larger number of bounding boxes to be processed compared to previous related approaches. The background elements are instead modeled by a mixture of components. The use of bounding boxes for the foreground objects could lead to under- or over-segmentation if the size of the bounding box is not tuned appropriately. An additional boundary loss tries to address the over-segmentation issue by penalizing splitting objects across bounding boxes.

**VAE baselines.** We train variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) as baselines that learn distributed representations. Following Greff et al. (2019), we use two different decoder architectures: one consisting of an MLP followed by transposed convolutions, and one where the MLP is replaced by a broadcast decoder (Watters et al., 2019). The VAEs are trained by maximizing the usual variational lower bound (ELBO).

## A.2. Implementation details

We implement our library in PyTorch (Paszke et al., 2019). All models are either re-implemented or adapted from available code, and quantitative results from the literature are reproduced, when available. As shown in Table 1, all methods included in our study were originally evaluated only on a subset of the datasets considered in our study. Thus, the recommended

Table 1: Datasets used for quantitative and/or qualitative evaluation in the publications corresponding to the four object-centric models considered in this study. Here we train and evaluate all models on all datasets.

| | CLEVR | Multi-dSprites | Objects Room | Shapestacks | Tetrominoes |
|---|---|---|---|---|---|
| MONet | ✓ | ✓* | ✓ | | |
| Slot Attention | ✓ | ✓ | | | ✓ |
| GENESIS | | ✓* | ✓ | ✓ | |
| SPACE | | | | | |

*These publications use a variant of Multi-dSprites with colored background as opposed to grayscale.

Table 2: Overview of the main hyperparameter values for MONet. When dataset-specific values are not given, the defaults are used.

| **Hyperparameter** | **Default value** | **Dataset-specific values** | | |
|---|---|---|---|---|
| | | **CLEVR** | **Shapestacks** | **Tetrominoes** |
| Optimizer | Adam | RMSprop | RMSprop | — |
| Learning rate | 1e-4 | 3e-5 | — | — |
| Batch size | 64 | 32 | — | — |
| Training steps | 500k | — | — | — |
| $\sigma_{\text{bg}}$ | 0.06 | — | 0.2 | 0.3 |
| $\sigma_{\text{fg}}$ | 0.1 | — | 0.24 | 0.36 |
| $\beta$ | 0.5 | — | 0.1 | — |
| $\gamma$ | 0.5 | — | — | — |
| Latent space size | 16 | — | — | — |
| U-Net blocks | 5 | 6 | — | 4 |

hyperparameters for a given model are likely to be suboptimal in the datasets on which such model was not evaluated. When a model performed particularly bad on a dataset, we attempted to find better hyperparameter values for the sake of the soundness of our study. We provide implementation and training details for each model below.

**MONet.** We re-implement MONet following the implementation details in Burgess et al. (2019). In order to make this model work satisfactorily on Shapestacks and Tetrominoes—the two datasets where MONet was not originally tested—we ran a grid search over hyperparameters on both datasets, as follows:

- Optimizer: Adam or RMSprop, both with default PyTorch parameters.
- $\beta \in \{0.1, 0.5\}$.
- Learning rate in {3e-5, 1e-4}.
- $(\sigma_{\text{bg}}, \sigma_{\text{fg}}) \in \{(0.06, 0.1), (0.12, 0.18), (0.2, 0.24), (0.25, 0.3), (0.3, 0.36)\}$.

A summary of the final hyperparameter choices is shown in Table 2.

**Slot Attention.** We re-implement the Slot Attention autoencoder based on the official TensorFlow implementation and the corresponding publication (Locatello et al., 2020). We mostly use the recommended hyperparameter values and learning rate schedule. On Objects Room and Shapestacks, we use the same parameters as for Multi-dSprites, which has the same resolution. On CLEVR, we make a few changes to accommodate the larger image size. For the decoder, we follow the approach in Locatello et al. (2020) and use the broadcast decoder from a broadcasted shape of $8 \times 8$ rather than $128 \times 128$, and use four times a stride of 2 in the decoder. For the encoder, we follow the set prediction architecture in Locatello et al. (2020) and use two strides of 2 in the encoder. Finally, we use a batch size of 32 rather than 64.

**GENESIS.** We re-implement GENESIS based on the official implementation and the corresponding publication (Engelcke et al., 2020b), and use the recommended hyperparameter values. On Objects Room, we use the same hyperparameters as described in the paper for Multi-dSprites and Shapestacks, which have the same resolution. On CLEVR, which has

Table 3: Hyperparameters for SPACE experiments. Here we show: the hyperparameters recommended by Lin et al. (2020b) for the 3D-Rooms dataset on the official code repository; the hyperparameter space considered for our random search; the chosen default values across datasets; the dataset-specific values for CLEVR and Tetrominoes, which override the defaults. We omit some of the hyperparameters that we left unchanged from Lin et al. (2020b).

| **Hyperparameter** | **Original (3D-Rooms)** | **Sweep values** | **Default value** | **Dataset-specific values** | |
|---|---|---|---|---|---|
| | | | | **CLEVR** | **Tetrominoes** |
| FG optimizer | RMSprop | RMSprop | RMSprop | — | — |
| FG learning rate | 1e-5 | {3e-6, 1e-5, 3e-5, 1e-4} | 3e-5 | 1e-4 | 1e-4 |
| BG optimizer | Adam | Adam | Adam | — | — |
| BG learning rate | 1e-3 | 1e-3 | 1e-3 | — | — |
| Batch size | 12 | $\{16, 32\}$ | 32 | — | — |
| $\sigma_{\text{bg}}$ | 0.15 | $\{0.05, 0.15, 0.35\}$ | 0.15 | 0.05 | — |
| $\sigma_{\text{fg}}$ | 0.15 | $\{0.02, 0.05, 0.15, 0.35\}$ | 0.15 | 0.05 | — |
| $G$ (FG grid size) | 8 | $\{4, 8\}$ | 8 | — | 4 |
| $K$ (BG n. of slots) | 5 | $\{1, 5\}$ | 5 | — | — |
| Boundary loss off step | 100k | {20k, 100k} | 20k | — | 100k |
| $\tau$ anneal end step | 20k | {20k, 50k} | 50k | 20k | — |
| Mean of $p(\mathbf{z}_{\text{pres}})$ (start/end values) | (0.1, 0.01) | {(0.1, 0.01), (0.5, 0.05)} | (0.5, 0.05) | (0.1, 0.01) | (0.1, 0.01) |
| Mean of $p(\mathbf{z}_{\text{scale}})$ (start/end values) | $(-1, -2)$ | $\{(-1, -2), (0, -1)\}$ | $(0, -1)$ | — | — |

$128 \times 128$ images, we use an additional stride of 2 in the convolutional layer at the middle of both encoder and decoder (the output padding in the decoder is adjusted accordingly). In this case we also reduce the batch size from 64 to 32. On Tetrominoes ($32 \times 32$ images), we change the first stride in the encoder and the last stride in the decoder from 2 to 1.

**SPACE.** We adapt the official PyTorch implementation of SPACE to integrate it in our library. While in Lin et al. (2020b) the authors train SPACE for 160k steps, here we train it for 200k. Since SPACE was not tested on any of the five datasets considered here (see Table 1), we perform a hyperparameter sweep for all datasets. For each of the five datasets, we run a random search over hyperparameters by training 100 models for 100k steps. Table 3 shows the random search definition, the hyperparameter values used for each dataset, and how they differ from those used in the original publication for the *3D-Rooms* dataset (although we omit some hyperparameters that we leave unchanged).

**VAEs.** The architecture details for the VAEs are presented in Tables 4 to 6. These are used for Shapestacks, Multi-dSprites, and Objects Room. For CLEVR, an additional ResidualBlock with 64 channels and a AvgPool2D layer is added at the end of the stack of ResidualBlocks, to downsample the image one more time. This is mirrored in the decoder, where a ResidualBlock with 256 channels and a (bilinear) Interpolation layer is added at the beginning of the stack of ResidualBlocks. The same happens in the broadcast decoder case. For Tetrominoes, the number of layers is the same, but the last AvgPool2D layer is removed from the encoder and the first Interpolation layer is removed from the decoder, to have one less downsampling and upsampling, respectively. The latent space size is chosen to be 64 times the number of slots that would be used when training an object-centric model on the same dataset. Note that the default number of slots varies depending on the dataset, as shown in Table 7.[4]

[4]Here we consider the default for MONet, Slot Attention, and GENESIS, and we disregard SPACE. Although SPACE has a much larger number of slots, this is not comparable with the other models because of the grid-based spatial attention mechanism.

Table 4: Structure of the encoder for both the vanilla and broadcast VAE, excluding the final linear layer that parameterizes $\mu$ and $\log \sigma^2$ of the approximate posterior.

| **Encoder** | | |
|---|---|---|
| *Type* | *Size/Ch.* | *Notes* |
| Input: $\mathbf{x}$ | 3 | |
| Conv $5 \times 5$ | | Stride 2, Padding 2 |
| LeakyReLU | | |
| | | |
| Residual Block | 64 | 2 Conv layers |
| Residual Block | 64 | 2 Conv layers |
| Conv $1 \times 1$ | 128 | |
| AvgPool2D | | Kernel size 2, Stride 2 |
| | | |
| Residual Block | 128 | 2 Conv layers |
| Residual Block | 128 | 2 Conv layers |
| AvgPool2D | | Kernel size 2, Stride 2 |
| | | |
| Residual Block | 128 | 2 Conv layers |
| Residual Block | 128 | 2 Conv layers |
| Conv $1 \times 1$ | 256 | |
| | | |
| Residual Block | 256 | 2 Conv layers |
| Residual Block | 256 | 2 Conv layers |
| | | |
| Flatten | | |
| LeakyReLU | | |
| Linear | 512 | |
| LeakyReLU | | |
| LayerNorm | | |

Table 5: Structure of the decoder for the vanilla VAE.

| **Vanilla Decoder** | | |
|---|---|---|
| *Type* | *Size/Ch.* | *Notes* |
| Input: $\mathbf{z}$ | $64\times$ num. slots | |
| LeakyReLU | | |
| Linear | 512 | |
| LeakyReLU | | |
| Unflatten | | |
| | | |
| Residual Block | 256 | 2 Conv layers |
| Residual Block | 256 | 2 Conv layers |
| Conv $1 \times 1$ | 128 | |
| Interpolation | | Scale 2 |
| | | |
| Residual Block | 128 | 2 Conv layers |
| Residual Block | 128 | 2 Conv layers |
| Interpolation | | Scale 2 |
| | | |
| Residual Block | 128 | 2 Conv layers |
| Residual Block | 128 | 2 Conv layers |
| Conv $1 \times 1$ | 64 | |
| Interpolation | | Scale 2 |
| | | |
| Residual Block | 64 | 2 Conv layers |
| Residual Block | 64 | 2 Conv layers |
| Interpolation | | Scale 2 |
| | | |
| LeakyReLU | | |
| Conv $5 \times 5$ | Image channels | Stride 1, Padding 2 |

Table 6: Structure of the decoder for the broadcast VAE. One less Interpolation is required, because the final image size for this architecture is $64$ and the broadcasting is to a feature map of size $8$.

| **Broadcast Decoder** | | |
|---|---|---|
| *Type* | *Size/Ch.* | *Notes* |
| Input: $\mathbf{z}$ | $64\times$ num. slots | |
| Broadcast | $64\times$ num. slots $+2$ | Broadcast dim. $8$ |
| | | |
| Residual Block | $256$ | 2 Conv layers |
| Residual Block | $256$ | 2 Conv layers |
| Conv $1 \times 1$ | $128$ | |
| | | |
| Residual Block | $128$ | 2 Conv layers |
| Residual Block | $128$ | 2 Conv layers |
| Interpolation | | Scale $2$ |
| | | |
| Residual Block | $128$ | 2 Conv layers |
| Residual Block | $128$ | 2 Conv layers |
| Conv $1 \times 1$ | $64$ | |
| Interpolation | | Scale $2$ |
| | | |
| Residual Block | $64$ | 2 Conv layers |
| Residual Block | $64$ | 2 Conv layers |
| | | |
| LeakyReLU | | |
| Conv $5 \times 5$ | Image channels | Stride $1$, Padding $2$ |

## B. Datasets

We collected 5 existing multi-object datasets and converted them into a common format. *Multi-dSprites*, *Objects Room* and *Tetrominoes* are from DeepMind's Multi-Object Datasets collection, under the Apache 2.0 license (Kabra et al., 2019). *CLEVR* was originally proposed by Johnson et al. (2017), with segmentation masks introduced by Kabra et al. (2019). *Shapestacks* was proposed by Groth et al. (2018) under the GPL 3.0 license. Details on these datasets are provided in the following subsections. See Fig. 10 for sample images and ground-truth segmentation masks for these datasets. In Table 7, we report dataset splits, number of foreground and background objects, and number of slots used when training object-centric models.

### B.1. CLEVR

This dataset consists of $128 \times 128$ images of 3D scenes with up to 10 objects, possibly occluding each other. Objects can have different colors (8 in total), materials (rubber or metal), shapes (sphere, cylinder, cube), sizes (small or large), x and y positions, and rotations. Objects can be occluded by others. On average, 6.2 objects are visible. As in previous work (Greff et al., 2019; Locatello et al., 2020), we learn object-centric representations on the CLEVR6 variant, which contains at most 6 objects. There are $100\,000$ samples in the full dataset, and $53\,483$ in the CLEVR6 variant (at most 6 objects). The CLEVR dataset has been cropped and resized according to the procedure detailed originally by Burgess et al. (2019).

Each object is annotated with the following properties:

- `color` (categorical): 8 colors:
  - Red. RGB: `[173, 35, 35]`
  - Cyan. RGB: `[41, 208, 208]`
  - Green. RGB: `[29, 105, 20]`
  - Blue. RGB: `[42, 75, 215]`
  - Brown. RGB: `[129, 74, 25]`
  - Gray. RGB: `[87, 87, 87]`
  - Purple. RGB: `[129, 38, 192]`
  - Yellow. RGB: `[255, 238, 51]`
- `material` (categorical): The material of the object: rubber or metal.
- `shape` (categorical): The shape of the object: sphere, cylinder or cube.
- `size` (categorical): The size of the object: small or large.
- `x` (numerical): The x coordinate in 3D space.
- `y` (numerical): The y coordinate in 3D space.

### B.2. Multi-dSprites

This dataset is based on the *dSprites* dataset (Matthey et al., 2017). Following previous work (Greff et al., 2019; Locatello et al., 2020), we use the Multi-dSprites variant with colored sprites on a grayscale background. Each scene has 2–5 objects with random shapes (ellipse, square, heart), sizes (6 discrete values in $[0.5, 1]$), x and y position, orientation, and color (randomly sampled in HSV space). Objects can occlude each other. The intensity of the uniform grayscale background is randomly sampled in each image. Images have size $64 \times 64$.

Each object is annotated with the following properties:

- `color` (numerical): 3-dimensional RGB color vector.
- `scale` (numerical): Scaling of the object, 6 uniformly spaced values between 0.5 and 1.
- `shape` (categorical): The shape type of the object (ellipse, heart, square).
- `x` (numerical): Horizontal position between 0 and 1.
- `y` (numerical): Vertical position between 0 and 1.

Table 7: Dataset splits, number of foreground and background objects, and number of slots used when training object-centric models.

| Dataset Name | Train Size | Validation Size | Test Size | Background Objects | Foreground Objects | Slots |
|---|---|---|---|---|---|---|
| CLEVR6 | 49 483 | 2000 | 2000 | 1 | 3–6 | 7* |
| Multi-dSprites | 90 000 | 5000 | 5000 | 1 | 2–5 | 6* |
| Objects Room | 90 000 | 5000 | 5000 | 4 | 1–3 | 7* |
| Shapestacks | 90 000 | 5000 | 5000 | 1 | 2–6 | 7* |
| Tetrominoes | 90 000 | 5000 | 5000 | 1 | 3 | 4† |

*In SPACE we use 69 slots: 5 background slots, and a grid of $8 \times 8$ foreground slots.
†In SPACE we use 21 slots: 5 background slots, and a grid of $4 \times 4$ foreground slots.

### B.3. Objects Room

This dataset was originally introduced by Eslami et al. (2018) and consists of $64 \times 64$ images of 3D scenes with up to three objects. Since this dataset includes masks but no labels for the object properties, we can use it only to evaluate segmentation performance.

### B.4. Shapestacks

This dataset consists of $64 \times 64$ images of 3D scenes where objects are stacked to form a tower. Each scene is available under different camera views. Object properties are shape (cube, cylinder, sphere), color (6 possible values), size (numerical) and ordinal position in the stack.

Each object is annotated with the following properties:

- `shape` (categorical): shape of the object: cylinder, sphere or cuboid.
- `color` (categorical): 6 colors:
  - Blue. RGB:`[0, 0, 255]`
  - Green. RGB:`[0, 255, 0]`
  - Cyan. RGB:`[0, 255, 255]`
  - Red. RGB:`[255, 0, 0]`
  - Purple. RGB:`[255, 0, 255]`
  - Yellow. RGB:`[255, 255, 0]`

### B.5. Tetrominoes

This dataset consists of $32 \times 32$ images (cropped from the original $35 \times 35$ for simplicity) of 3D-textured tetris pieces placed on a black background. There are always 3 objects in a scene, and no occlusions. Objects have different shapes (19 in total), colors (6 fully saturated colors), x and y position.

Each object is annotated with the following properties:

- `shape` (categorical): 19 shapes:
  - Horizontal I piece.
  - Vertical I piece.
  - L piece pointing downward.
  - J piece pointing upward.
  - L piece pointing upward.
  - J piece pointing downward.
  - L piece pointing left.

  - J piece pointing left.
  - J piece pointing right.
  - L piece pointing right.
  - Horizontal Z piece.
  - Horizontal S piece.
  - Vertical Z piece.
  - Vertical S piece.
  - T piece pointing upward.
  - T piece pointing downward.
  - T piece pointing left.
  - T piece pointing right.
  - O piece.
- `color` (categorical): 6 colors:
  - Blue. RGB:`[0, 0, 255]`
  - Green. RGB:`[0, 255, 0]`
  - Cyan. RGB:`[0, 255, 255]`
  - Red. RGB:`[255, 0, 0]`
  - Purple. RGB:`[255, 0, 255]`
  - Yellow. RGB:`[255, 255, 0]`
- `x` (numerical): Horizontal position.
- `y` (numerical): Vertical position.

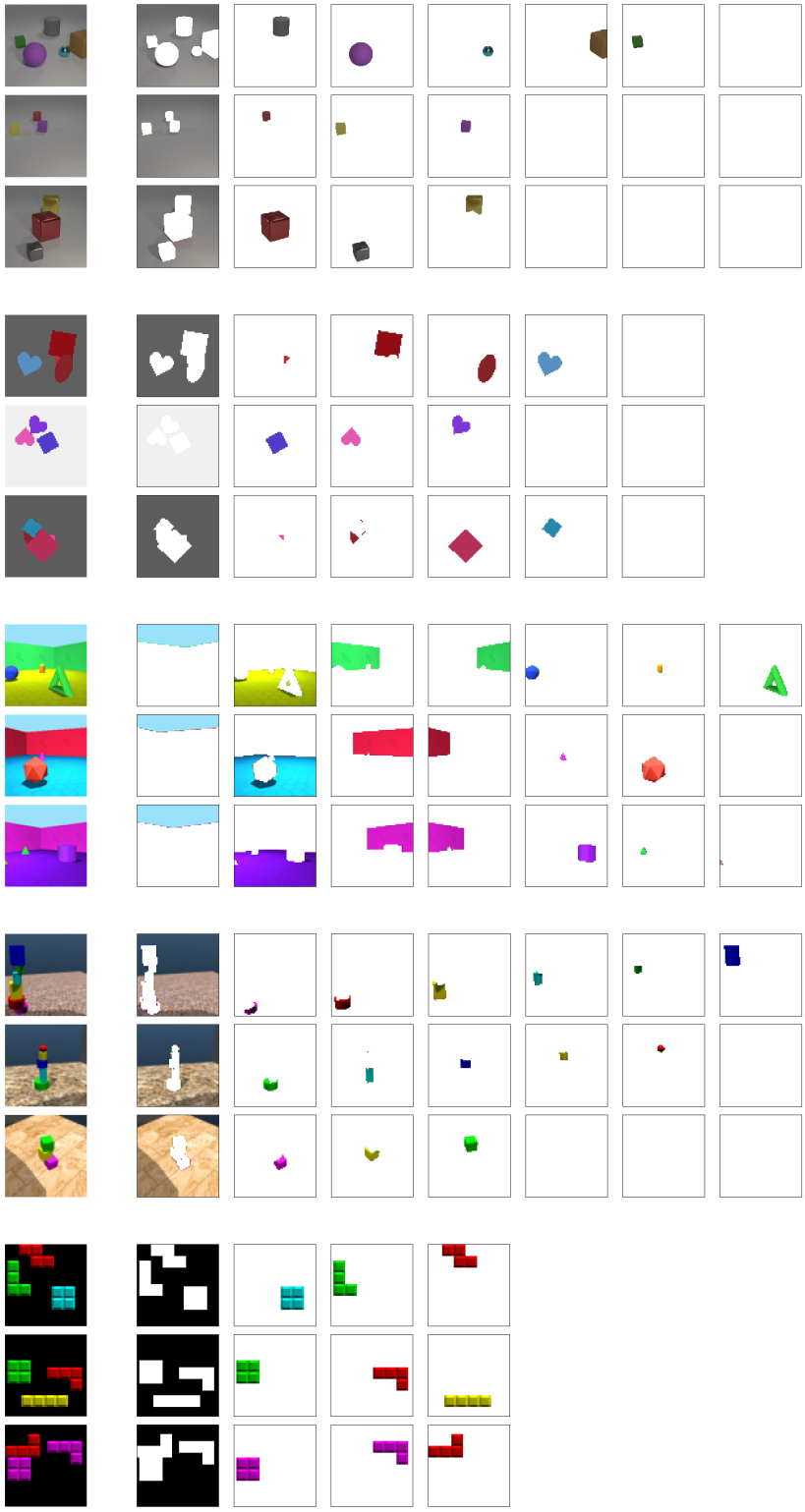

Figure 10: Examples of images from the datasets considered in this work. The leftmost column represents the original image, the other columns show all the objects in the scene according to the ground-truth segmentation masks. Top to bottom: CLEVR6, Multi-dSprites, Objects Room, Shapestacks, Tetrominoes.

# C. Evaluations

In this section, we discuss in more detail the chosen reconstruction and segmentation metrics (Appendix C.1), provide implementation details on the downstream property prediction task (Appendix C.2), and more closely examine the distribution shifts considered in this study (Appendix C.3).

## C.1. Reconstruction and segmentation metrics

**Mean reconstruction error.** Since all models in this study are autoencoders, we can use the reconstruction error to This is potentially an informative metric as it should roughly indicate the amount and accuracy of information captured by the models and present in the representations. All models include some form of reconstruction term in their losses, but they may take different forms. We then choose to evaluate the reconstruction error with the mean squared error (MSE), defined for an image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ as follows:

$$\text{MSE}\left(\mathbf{x}, \hat{\mathbf{x}}\right) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \frac{1}{D}\sum_{i=1}^{D}(x_i - \hat{x}_i)^2 \tag{1}$$

where for simplicity we assume a vector representation of $\mathbf{x}$ and $\hat{\mathbf{x}}$, both with dimension $D$ equal to the number of pixels times the number of color channels.

**Adjusted Rand Index (ARI).** The Adjusted Rand Index (ARI) (Hubert & Arabi, 1985) measures the similarity between two partitions of a set (or clusterings). Interpreting segmentation as clustering of pixels, the ARI can be used to measure the degree of similarity between two sets of segmentation masks. Segmentation accuracy is then assessed by comparing ground-truth and predicted masks. The expected value of the ARI on random clustering is 0, and the maximum value is 1 (identical clusterings up to label permutation). As in prior work (Burgess et al., 2019; Engelcke et al., 2020b; Locatello et al., 2020), we only consider the ground-truth masks of foreground objects when computing the ARI. Below, we define the Rand Index and the Adjusted Rand Index in more detail.

The Rand Index is a symmetric measure of the similarity between two partitions of a set (Hubert & Arabi, 1985; Rand, 1971; Wagner & Wagner, 2007). It is inspired by traditional classification metrics that compare the number of correctly and incorrectly classified elements. The Rand Index is defined as follows: Let $S$ be a set of $n$ elements, and let $A = \{A_1, \ldots, A_{n_A}\}$ and $B = \{B_1, \ldots, B_{n_B}\}$ be partitions of $S$. Furthermore, let us introduce the following quantities:

- $m_{11}$: number of pairs of elements that are in the same subset in both $A$ and $B$,
- $m_{00}$: number of pairs of elements that are in different subsets in both $A$ and $B$,
- $m_{10}$: number of pairs of elements that are in the same subset in $A$ and in different subsets in $B$,
- $m_{01}$: number of pairs of elements that are in different subsets in $A$ and in the same subset in $B$.

The Rand Index is then given by:

$$\text{RI}(A, B) = \frac{m_{11} + m_{00}}{m_{11} + m_{00} + m_{10} + m_{01}} = \frac{2(m_{11} + m_{00})}{n(n-1)} \tag{2}$$

and quantifies the number of elements that have been correctly classified over the total number of elements.

The Rand Index ranges from 0 (no pair classified in the same way under $A$ and $B$) to 1 ($A$ and $B$ are identical up to a permutation). However, the result is strongly dependent on the number of clusters and on the number of elements in each cluster. If we fix $n_A$, $n_B$, and the proportion of elements in each subset of the two partitions, then the Rand Index will increase as $n$ increases, and even converge to 1 in some cases (Fowlkes & Mallows, 1983). The expected value of a random clustering also depends on the number of clusters and on the number of elements $n$.

The Adjusted Rand Index (ARI) (Hubert & Arabi, 1985) addresses this issue by normalizing the Rand Index such that, with a random clustering, the metric will be 0 in expectation. Given the same conditions as above, let $n_{i,j} = |A_i \cap B_j|$,

$a_i = |A_i|$, and $b_i = |B_i|$, with $i = 1, \dots, n_A$ and $i = 1, \dots, n_B$. The ARI is then defined as:

$$\mathrm{ARI}(A, B) = \frac{\sum_{i,j} \binom{n_{i,j}}{2} - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}} \tag{3}$$

which is 0 in expectation for random clusterings, and 1 for perfectly matching partitions (up to a permutation). Note that the ARI can be negative.

**Segmentation covering metrics.** Segmentation Covering (SC) (Arbelaez et al., 2010) uses the intersection over union (IOU) between pairs of segmentation masks from the sets $A$ and $B$. How the segmentation masks are matched depends on whether we are considering the covering of $B$ by $A$ (denoted by $A \to B$) or vice versa ($B \to A$). We use the slightly modified definition by Engelcke et al. (2020b):

$$\mathrm{SC}(A \to B) = \frac{1}{\sum_{R_B \in B} |R_B|} \sum_{R_B \in B} |R_B| \max_{R_A \in A} \mathrm{IOU}(R_A, R_B) \,, \tag{4}$$

where $|R|$ denotes the number of pixels belonging to mask $R$, and the intersection over union is defined as:

$$\mathrm{IOU}(R_A, R_B) = \frac{|R_A \cap R_B|}{|R_A \cup R_B|} \,. \tag{5}$$

While standard (weighted) segmentation covering weights the IOU by the size of the ground truth mask, mean (or unweighted) segmentation covering (mSC) (Engelcke et al., 2020b) gives the same importance to masks of different size:

$$\mathrm{mSC}(A \to B) = \frac{1}{|B|} \sum_{R_B \in B} \max_{R_A \in A} \mathrm{IOU}(R_A, R_B) \,, \tag{6}$$

where $|B|$ denotes the number of non-empty masks in $B$. Since a high SC score can still be attained when small objects are not segmented correctly, mSC is considered to be a more meaningful and robust metric across different datasets (Engelcke et al., 2020b).

Note that neither SC nor mSC are symmetric: Following Engelcke et al. (2020b), we consider $A$ to be the predicted segmentation masks and $B$ the ground-truth masks of the foreground objects. As observed by Engelcke et al. (2020b), both SC and mSC penalize over-segmentation (segmenting one object into separate slots), unlike the ARI. Both SC and mSC take values in $[0, 1]$.

### C.2. Downstream property prediction

Here we start by briefly summarizing the downstream property prediction task presented in the main text, and then provide additional details on the models and evaluation protocol.

**Overview of the property prediction task.** As outlined in Section 3, we evaluate scene representations by training downstream models to predict ground-truth object properties from the representations. Exploiting the fact that object slots share a common representational format, a single downstream model $f$ can be used to predict the properties of each object independently: for each slot representation $\mathbf{z}_k$ we predict a vector of object properties $\hat{\mathbf{y}}_k = f(\mathbf{z}_k)$. This vector represents predictions for *all* properties of an object. We then match each slot's prediction to a corresponding ground-truth object using *mask matching* or *loss matching* (see main text). In non-slotted models such as the VAE baselines considered in this study, we do not have access to separate object representations $\{\mathbf{z}_k\}_{k=1}^K$. Therefore, the downstream model $f$ in this case takes as input the overall distributed representation $\mathbf{z}$, which is a flat vector, and outputs a prediction of *all objects at once*: $\hat{\mathbf{y}} = f(\mathbf{z})$. This is then split into $K$ vectors, which are matched to ground-truth objects with either *loss matching* or *deterministic matching* (see main text).

Table 8: Architecture of the downstream MLP models for property prediction. The third and fourth items are repeated 0 or more times, depending on the required number of hidden layers.

| Layer type | Input size | Output size | |
|---|---|---|---|
| Linear | $d$ or $d \cdot K$ | 256 | |
| LeakyReLU(0.01) | 256 | 256 | |
| Linear | 256 | 256 | } repeated 0 or |
| LeakyReLU(0.01) | 256 | 256 | } more times |
| Linear | 256 | $P$ or $P \cdot K$ | |

**Implementation details.** We use 4 different downstream models: a linear model, and MLPs with up to 3 hidden layers of size 256 each. Let $P$ be the size of the ground-truth property vector, which includes all numerical and categorical[5] properties according to an order specified by the dataset. We denote by $K$ be the number of slots and $d$ the dimensionality of a slot representation $\mathbf{z}_k$ in object-centric models. Note that we must include in $\mathbf{z}_k$ *all* representations related to a slot, possibly including different latent variables that are explicitly responsible for modeling, e.g., the location, appearance, or presence of an object. The downstream model $f$ has input size $d$ and output size $P$, and is applied in parallel (with shared weights) to all slots. In non-slotted models, we always define the dimensionality of the distributed representation $\mathbf{z}$ in terms of $K$ for fair comparison with slot-based models, hence we can write the latent dimensionality of such models as $d \cdot K$. In this case, the input and output sizes of the downstream model ($d$ and $P$, respectively) are multiplied by $K$, and we apply this model only once, to the entire scene representation. The linear downstream model is implemented as a linear layer. MLP models (with at least one hidden layer) have hidden size 256 and LeakyReLU nonlinearities, as shown in Table 8.

**Data splits.** Let $\mathcal{D}_s$ be a source dataset and $\mathcal{D}_t$ a target dataset. When doing in-distribution evaluation, we train and test the downstream model without distribution shifts, so we simply have $\mathcal{D}_s = \mathcal{D}_t$. Given a representation function $r$, and a matching strategy to match the slots with ground-truth objects, we consider:

- a train split of $10\,000$ images from $\mathcal{D}_s$ ,
- a validation split of $1000$ images from $\mathcal{D}_s$ ,
- a test split of $2000$ images from $\mathcal{D}_t$ .

The test split only contains images that were not used when training the upstream unsupervised models.

**Training.** We then train the downstream model to predict $\hat{\mathbf{y}}$ from $\mathbf{z} = r(\mathbf{x})$ using the Adam optimizer with an initial learning rate of 1e-3 and a batch size of 64, for a maximum of $6000$ steps. The learning rate is halved every $2000$ steps. We perform early stopping as follows: We use the validation set to compute the (in-distribution) validation loss every $250$ training steps—if the loss does not decrease by more than $0.01$ for 3 evaluations (750 steps), training is interrupted. In this stage, the representation for each image is fixed, i.e. the representation function $r$ is never updated. The loss is computed independently for each object property, and is a sum of MSE and cross-entropy terms, depending on whether an object property is numerical or categorical.

**Downstream training and evaluation under distribution shifts.** As mentioned earlier, when doing in-distribution evaluation we simply have $\mathcal{D}_s = \mathcal{D}_t$. In the general case, we may for example train on the original Multi-dSprites dataset, and test on the Multi-dSprites variant that has an unseen shape or an occlusion. In the special case in which we allow retraining of the downstream model (see Sections 4.3 and 4.4), we still have $\mathcal{D}_s = \mathcal{D}_t$, but they are both OOD with respect to the original "clean" dataset used for training the unsupervised models.

Under distribution shifts, the representations $r(\mathbf{x})$ might be inaccurate, which might bias our downstream results. Although there is no perfect solution to this issue, we attempt to reduce as much as possible the potential effect of distribution shifts on the training and evaluation of downstream models. When distribution shifts affect global scene properties, there is no alternative but to train and evaluate the models as usual. When distribution shifts affect single objects, however, we can assume that the representations of the ID objects are not as severely affected by the shift, and only use these for training

[5]Here we use the one-hot representation of categorical properties.

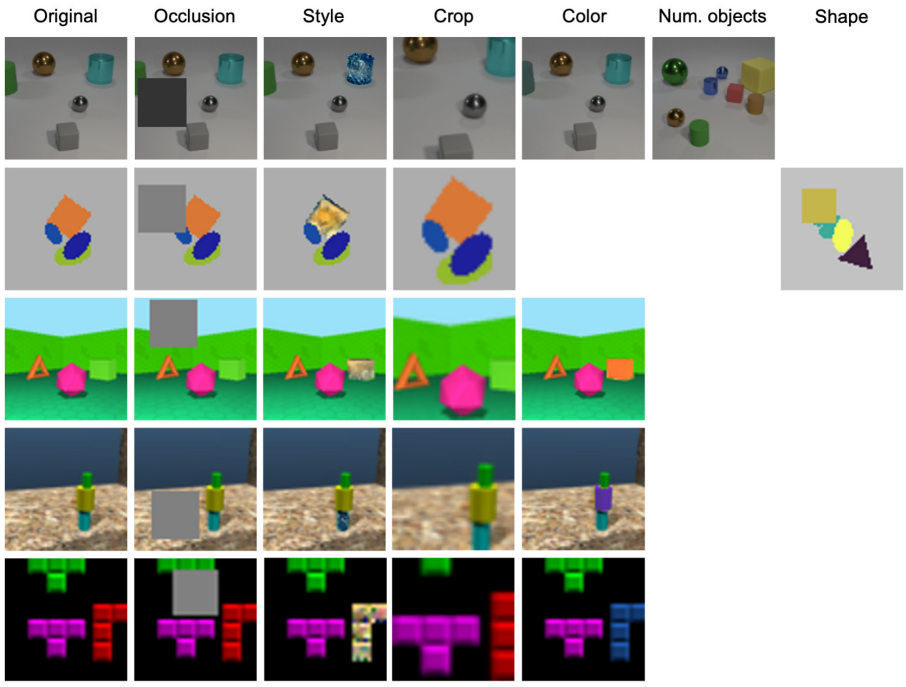


Figure 11: Distribution shifts applied to the different datasets to test generalization.

downstream models.

Here we consider the case where the test dataset $\mathcal{D}_t$ has an object-level distribution shift, and the training dataset $\mathcal{D}_s$ is either the original "clean" dataset or the same as $\mathcal{D}_t$. At **train time**, we ignore OOD objects (if any) both when matching slots with objects and when training downstream property prediction models. Note that, when the training dataset $\mathcal{D}_s$ is the original "clean" dataset, the downstream models are always trained as usual because there are no OOD objects. At **test time**, there are a few cases depending on the matching strategy:

- When using mask matching, we consider *all* objects for matching, and evaluate the downstream models on all objects. We then report test results on ID and OOD objects separately.
- When using loss matching, we cannot match all ground-truth objects, since the OOD objects might have OOD categorical properties (in our setup, the downstream models cannot predict classes that were not seen during training). Therefore, we resort to a two-step matching approach: we first match slots to all objects using the prediction loss computed only on the properties that are ID for *all* objects. We then keep only the matches for OOD objects, and repeat the usual loss matching with the remaining slots and objects, using all properties. The OOD objects are thus matched in a relatively fair way, while the matching of the ID objects can be refined at a later step using all available properties.
- When using deterministic matching, we cannot exactly follow the two-step matching strategy presented above. Instead, we modify the lexicographic order to give a higher weight to OOD features of OOD objects, so the corresponding objects are pushed down in the order while maintaining the order given by more significant (according to the order) properties. Note that the downstream model in this case might be at a disadvantage if it is trained on a dataset with object-level distribution shifts: the model is now trained to predict only ID objects, so at test time there will be one more target object on average.

### C.3. Distribution shifts for OOD evaluation

Here we present more in detail the distribution shifts we apply to images in order to test OOD generalization in different scenarios. Examples are shown in Fig. 11.

**Occlusion.** A gray square is placed on top of the scene. The position is determined by picking 5 locations uniformly at random (such that the entire square is in the image) and selecting the one that occludes less (in terms of total area) of the

foreground objects. The size of the occlusion is $(\lfloor 0.4 \cdot H \rfloor, \lfloor 0.4 \cdot W \rfloor)$ with $H$ and $W$ the height and width of the image, respectively. Occluded objects have their mask updated to reflect the occlusion. The occlusion is categorized as background (or first background object in case there are multiple background objects such as in Objects Room). The RGB color of the square is $[0.2, 0.2, 0.2]$ for CLEVR and $[0.5, 0.5, 0.5]$ for all other datasets.

**Object color.** An object is selected uniformly at random and its color is changed by randomly adjusting its brightness, contrast, saturation, and hue, using torchvision's `ColorJitter` transform with arguments $[0.5, 0.5, 0.5, 0.5]$ for the four above-mentioned parameters. This transformation is not performed on Multi-dSprites, since the object colors in this dataset cover the entire RGB color space. The `color` and `material` properties (when relevant) are not used in downstream tasks.

**Crop.** The image and mask are cropped at the center and resized to match their original size. The crop size is $(\lfloor \frac{2}{3} H \rfloor, \lfloor \frac{2}{3} W \rfloor)$ with $H$ and $W$ the original height and width of the image, respectively. When resizing, we use bilinear interpolation for the image and nearest neighbor for the mask.

**Object style.** We implement style transfer based on Gatys et al. (2016) and on the PyTorch tutorial by Jacq & Herring (2021). The first 100k samples in all datasets are converted using as style image *The Great Wave off Kanagawa* from Hokusai's series *Thirty-six Views of Mount Fuji*. The style is applied only to one foreground object using the object masks. The `color` and `material` properties (when relevant) are not used in downstream tasks.

**Object shape.** For the Multi-dSprites dataset, a triangle is placed on the scene with properties sampled according to the same distributions defined by the Multi-dSprites dataset. This is performed only on the images where at most 4 objects are present, to mimic changing the shape of an existing object. The depth of the triangle in the object stack is selected uniformly at random as an integer in $[1, 5]$. All objects from the selected depth and upwards are moved up by one level to place the new shape underneath them. The objects masks are adjusted accordingly for both the added shape and the objects below it.

# D. Additional results

In this section, we report additional quantitative results and show qualitative performance on all datasets for a selection of object-centric models and VAE baselines.

## D.1. Performance in the training distribution

Fig. 12 shows the distributions of the reconstruction MSE and all the segmentation metrics, broken down by dataset and model. The relationship between these metrics is also shown in scatter plots in Fig. 13. As discussed in Section 4.1, we observe that segmentation covering metrics are correlated with the ARI only in some cases, and the models are ranked very differently depending on the chosen segmentation metric. In particular, we observe here that Slot Attention achieves a high ARI score and significantly lower (m)SC scores on CLEVR, Multi-dSprites, and Tetrominoes. This is because Slot Attention on these datasets tends to model the background across many slots (see Appendix D.3), which is penalized by the denominator of the IOU in the (m)SC scores (see Eqs. (4) to (6) in Appendix C.1). This behavior should not have a major effect on downstream performance, which is confirmed by the strong and consistent correlation between ARI and downstream performance (see also Section 4.2 and Fig. 16).

Fig. 14 shows an overview of downstream factor prediction performance on all labeled datasets (one per column), using as downstream predictors a linear model or an MLP with up to 3 hidden layers (one model per row). The MLP1 results are also shown in Section 4.2 (Fig. 4). We report results separately for each object-centric model and for each ground-truth object property. The metrics used here are accuracy for categorical attributes and $R^2$ for numerical attributes. We generally observe consistent trends across downstream models.

In Fig. 15, we show the same results in a different way, to directly compare downstream models (here the median baselines for slot-based and distributed representations are shown as horizontal lines on top of the relevant bars). Using larger downstream models tends to slightly improve test performance but, interestingly, in many cases the effect is negligible. There are however a few cases in which using a larger model significantly boosts test performance in object property prediction. In some cases it seems sufficient to use a small MLP with one hidden layer instead of a linear model (e.g., color prediction in CLEVR with Slot Attention, shape prediction in CLEVR with MONet and Slot Attention, color prediction in Tetrominoes with MONet, GENESIS, and SPACE, or location prediction in Multi-dSprites with SPACE), while in other cases we get further gains by using even larger models (e.g., shape prediction in Multi-dSprites with SPACE, and shape prediction in Tetrominoes with all models except Slot Attention which already achieves a perfect score with a linear model).

Results for VAEs are generally less interpretable because the performance is often too close to the naive baseline. However, in some cases using deeper downstream models has clear benefits: e.g., shape prediction in Tetrominoes and color prediction in Shapestacks improve from baseline level when using a linear model to a relatively high accuracy when using one or two hidden layers. In other cases, a linear model already works relatively well even from distributed representations—although significantly worse than object-centric representations—and using deeper downstream models is not beneficial (e.g., color and size prediction in CLEVR). In many other cases, larger downstream models do not seem to be sufficient to improve performance from VAE representations, confirming that often the relevant information may not be easily accessible and suggesting that object-centric representations may be generally beneficial.

In Fig. 16 we show the Spearman rank correlations between evaluation metrics and downstream performance with all considered combinations of slot matching (loss- and mask-based) and downstream model (linear, MLP with 1, 2, or 3 hidden layers). The trends are broadly consistent in all combinations, except that correlations with ARI tend to be stronger (perhaps unsurprisingly) when using mask matching, and when using larger downstream models.

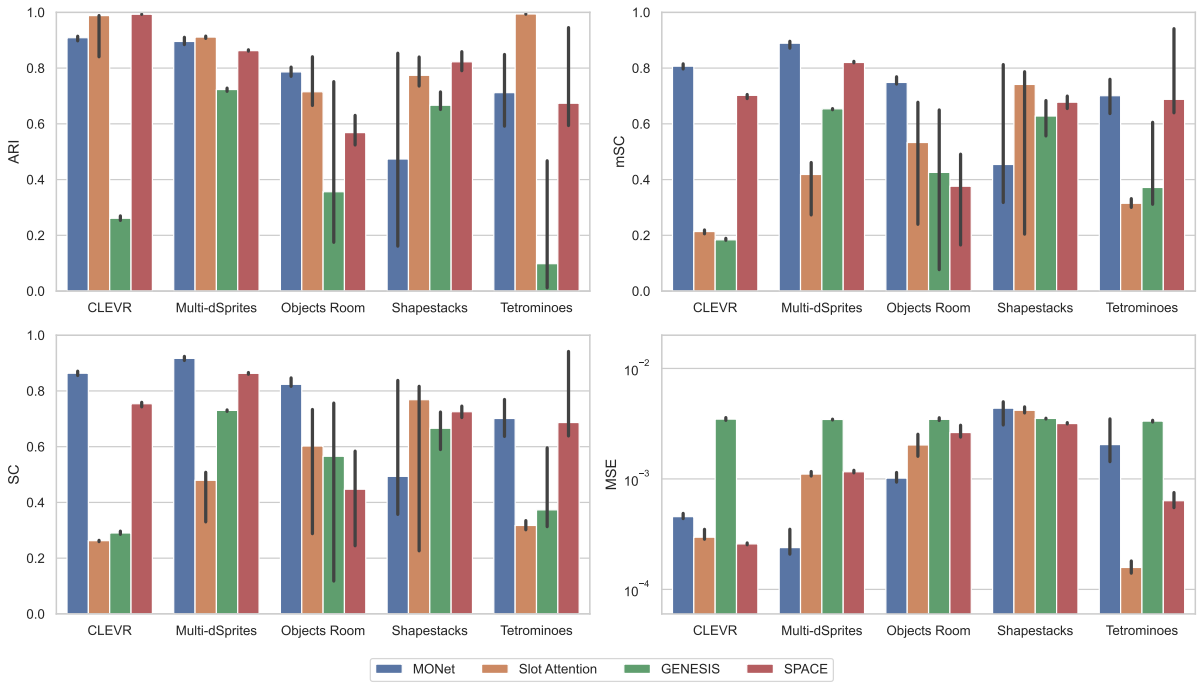


Figure 12: Overview of segmentation metrics (ARI ↑, mSC ↑, SC ↑) and reconstruction MSE (↓) in distribution (test set of 2000 images). The bars show medians and 95% confidence intervals with 10 random seeds.

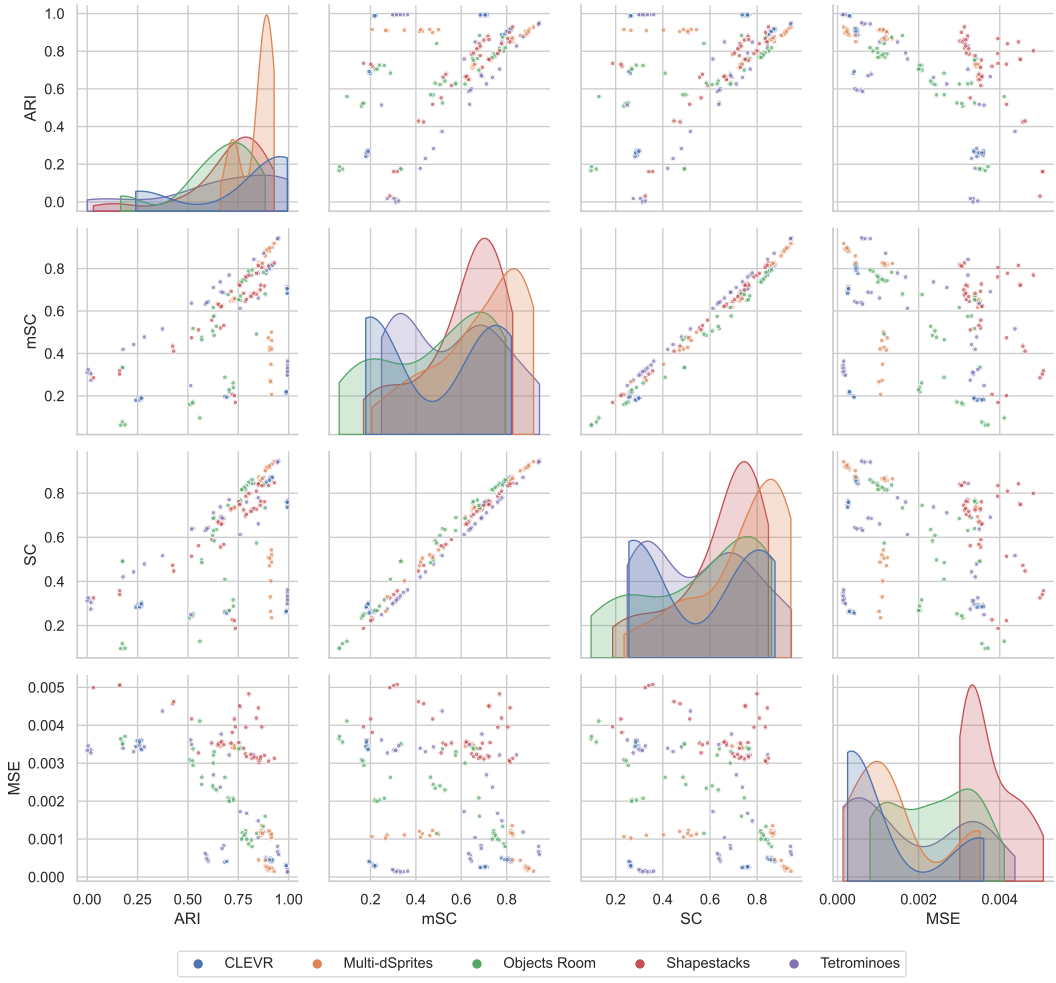


Figure 13: Scatter plots between metrics over all 200 object-centric models, color-coded by dataset. Diagonal plots: kernel density estimation (KDE) of the quantities on the x-axes.

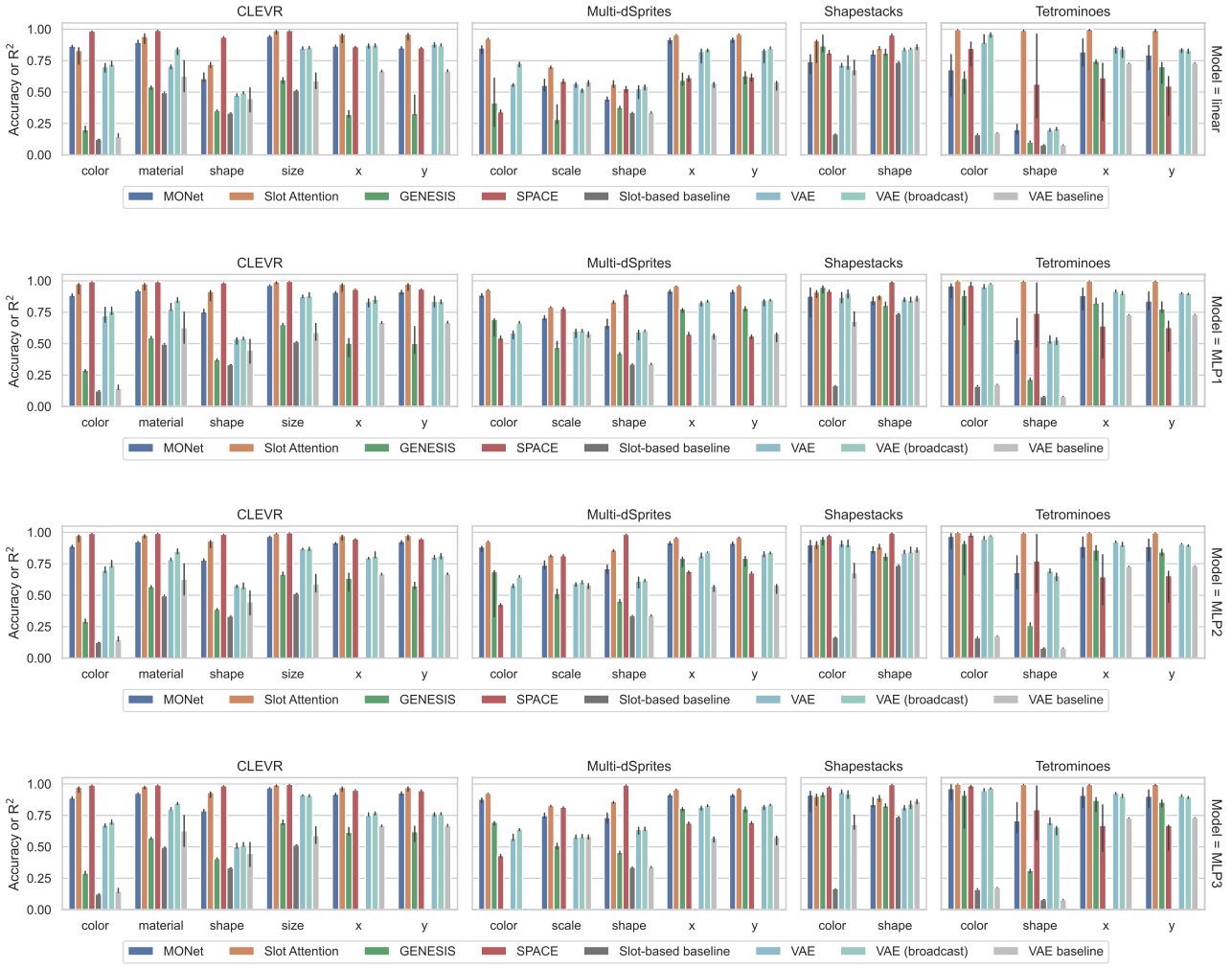


Figure 14: Overview of downstream performance in the training distribution (test set of 2000 images) for object-centric models and VAEs, with respective baselines. The metrics on the y-axes are accuracy (↑) for categorical properties and $R^2$ (↑) for numerical features. Each row show results for a different downstream prediction model. From top to bottom: linear, MLP with 1, 2, and 3 hidden layers (see annotation on the right). We use loss matching (see Section 3) for all models. The bars show medians and 95% confidence intervals with 10 random seeds.

Figure 15: Comparing property prediction performance of different downstream models (linear, MLP with 1 to 3 hidden layers), using loss matching (see Section 3). Results on a test set of 2000 images in the training distribution of the upstream unsupervised models. Each plot shows the test performance on one feature of a dataset. We show results for all object-centric models and VAEs, and indicate the baseline (see Section 3) with a horizontal line (not visible when the baseline is 0). The metrics on the y-axes are accuracy (↑) for categorical properties and $R^2$ (↑) for numerical features. The bars show medians and 95% confidence intervals with 10 random seeds.

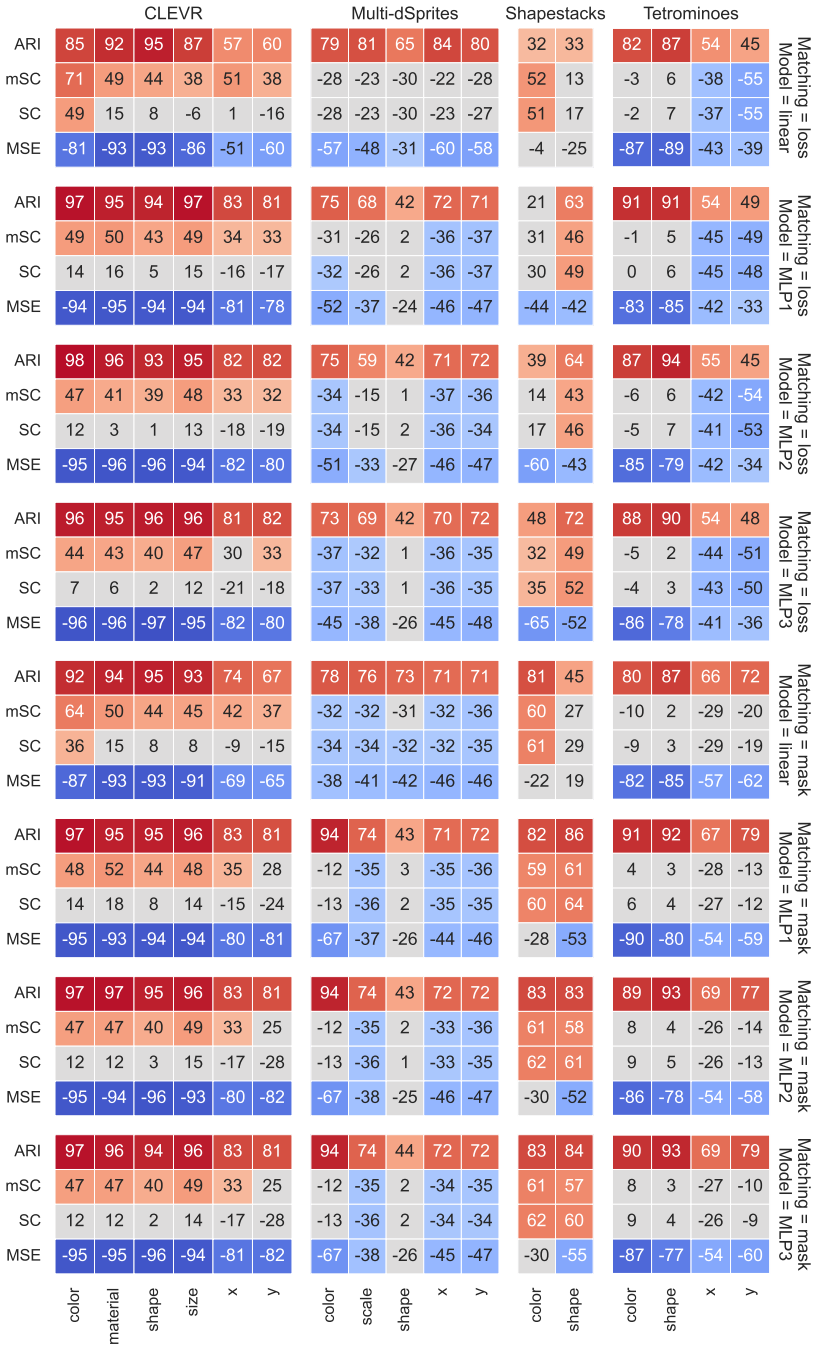


Figure 16: Spearman rank correlations between evaluation metrics and downstream performance with all considered combinations of slot matching (loss- and mask-based) and downstream model (linear, MLP with 1, 2, or 3 hidden layers). The correlations are color-coded only when $p<0.05$.

## D.2. Performance under distribution shifts

### D.2.1. Segmentation and reconstruction

In Fig. 17, we report the distributions of the reconstruction MSE and segmentation metrics in scenarios where one object is OOD. Results are split by dataset, model, and type of distribution shift. As discussed in Section 4.3, the SC and mSC scores show a compatible but less pronounced trend, while the MSE more closely mirrors ARI. Notably, in many cases when we alter object style or color, the reconstruction MSE increases significantly while the ARI is only mildly affected. This suggests that the models are still capable of separating the objects but, unsurprisingly, they fail at reconstructing them properly as they have features that were never encountered during training.

Fig. 18 shows analogous results when the distribution shift affects global scene properties. Here we observe that segmentation performance is relatively robust to occlusion although the MSE increases significantly (as expected, the occlusion cannot be reconstructed properly). Segmentation metrics are also robust to increasing the number of objects in CLEVR—here the MSE also increases, but to a lesser extent, especially for SPACE.

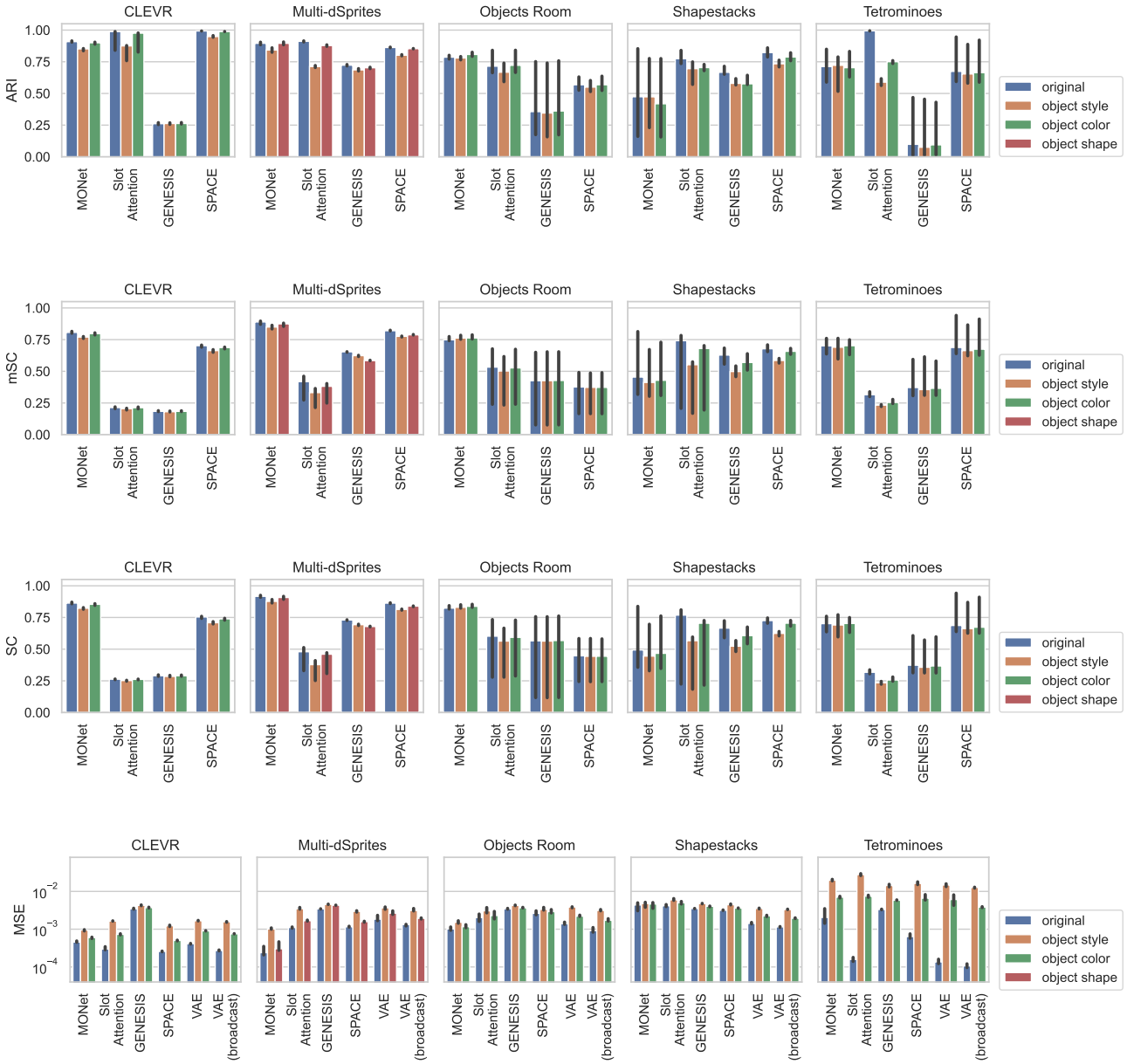


Figure 17: Overview of segmentation metrics (ARI ↑, mSC ↑, SC ↑) and reconstruction MSE (↓) on OOD dataset variants where **one object** undergoes distribution shift (test set of 2000 images). The bars show medians and 95% confidence intervals with 10 random seeds.

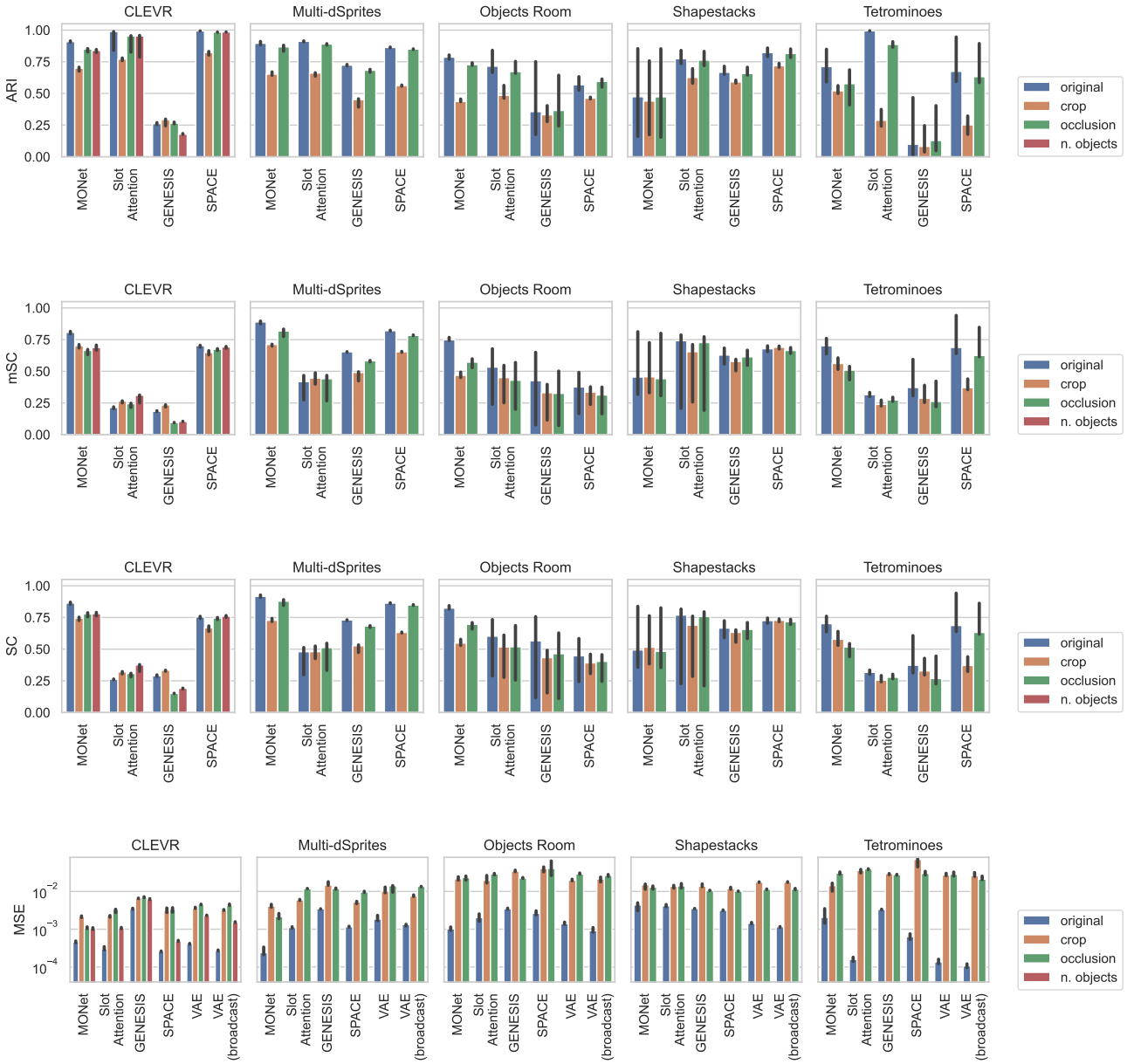


Figure 18: Overview of segmentation metrics (ARI ↑, mSC ↑, SC ↑) and reconstruction MSE (↓) on OOD dataset variants where **global properties** of the scene are altered (test set of 2000 images). The bars show medians and 95% confidence intervals with 10 random seeds.

#### D.2.2. DOWNSTREAM PERFORMANCE

In Figs. 19 to 34, we show the relationship between ID and OOD downstream prediction performance for the same model, dataset, downstream predictor, and object property. Assume a pretrained unsupervised object discovery model is given, and a downstream model is trained from said model's representations to predict object properties. These plots answer the following question: given that the downstream model predicts a particular object property (e.g., size in CLEVR) with a certain accuracy (on average over all objects in all test images), how well is it going to predict the same property when the scene undergoes one of the possible distribution shifts considered in this study? And in case the distribution shift only affects one object, how well is it going to predict that property in the ID objects as opposed to the OOD objects? These 16 figures show all combinations of the following 4 factors (hierarchically in this order): object-centric/distributed representations; loss/mask matching for object-centric representations or loss/deterministic for distributed representations; without/with retraining of the downstream model after the distribution shift has occurred; single-object/global distribution shifts. In each figure, we show results for each of the 4 downstream models considered in this study (linear, and MLP with up to 3 hidden layers). For each of these, we show splits in terms of ID/OOD objects (when applicable), dataset, upstream model, type of distribution shift.

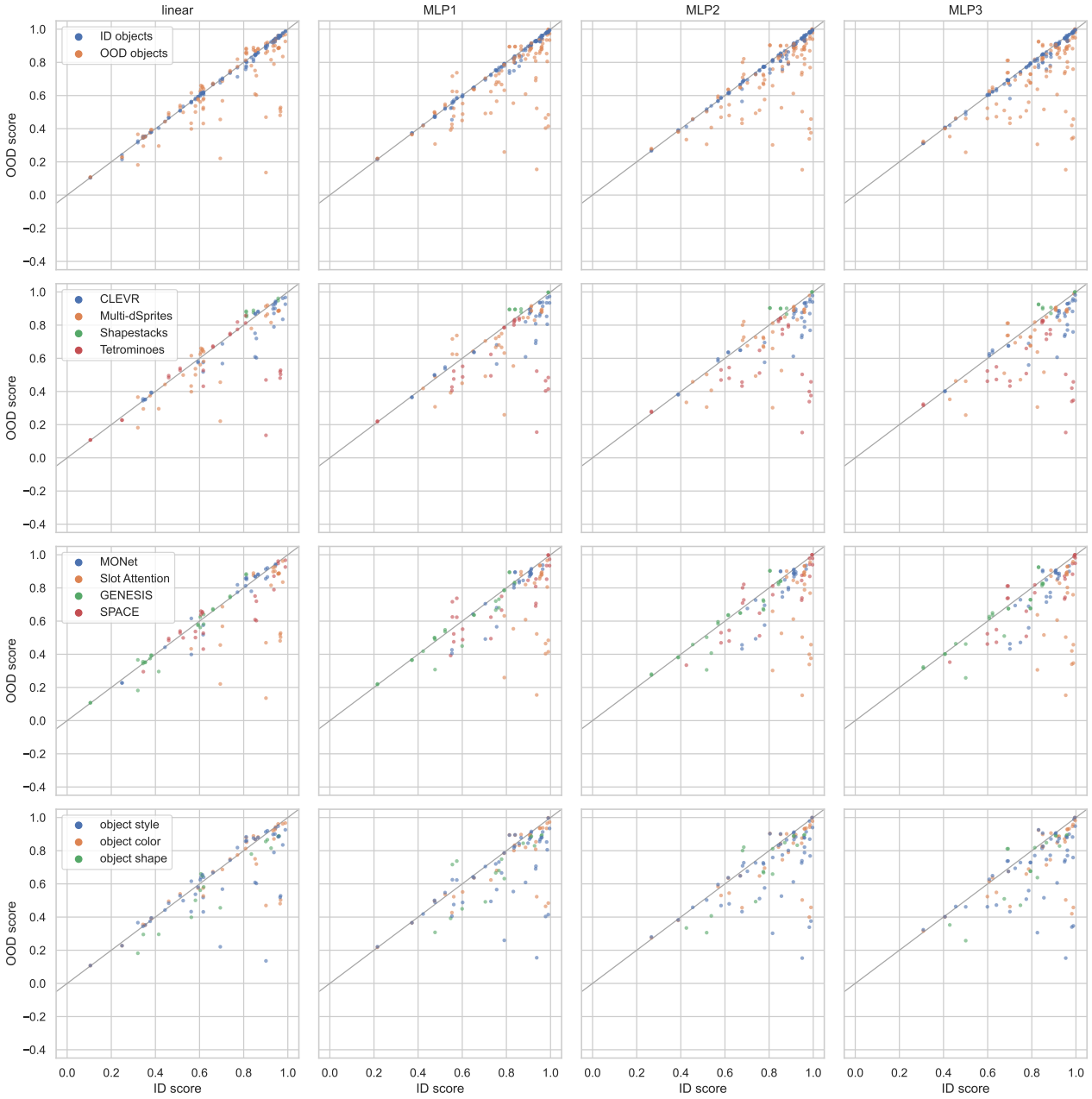


Figure 19: Generalization of **object-centric representations** in downstream prediction, using **loss matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

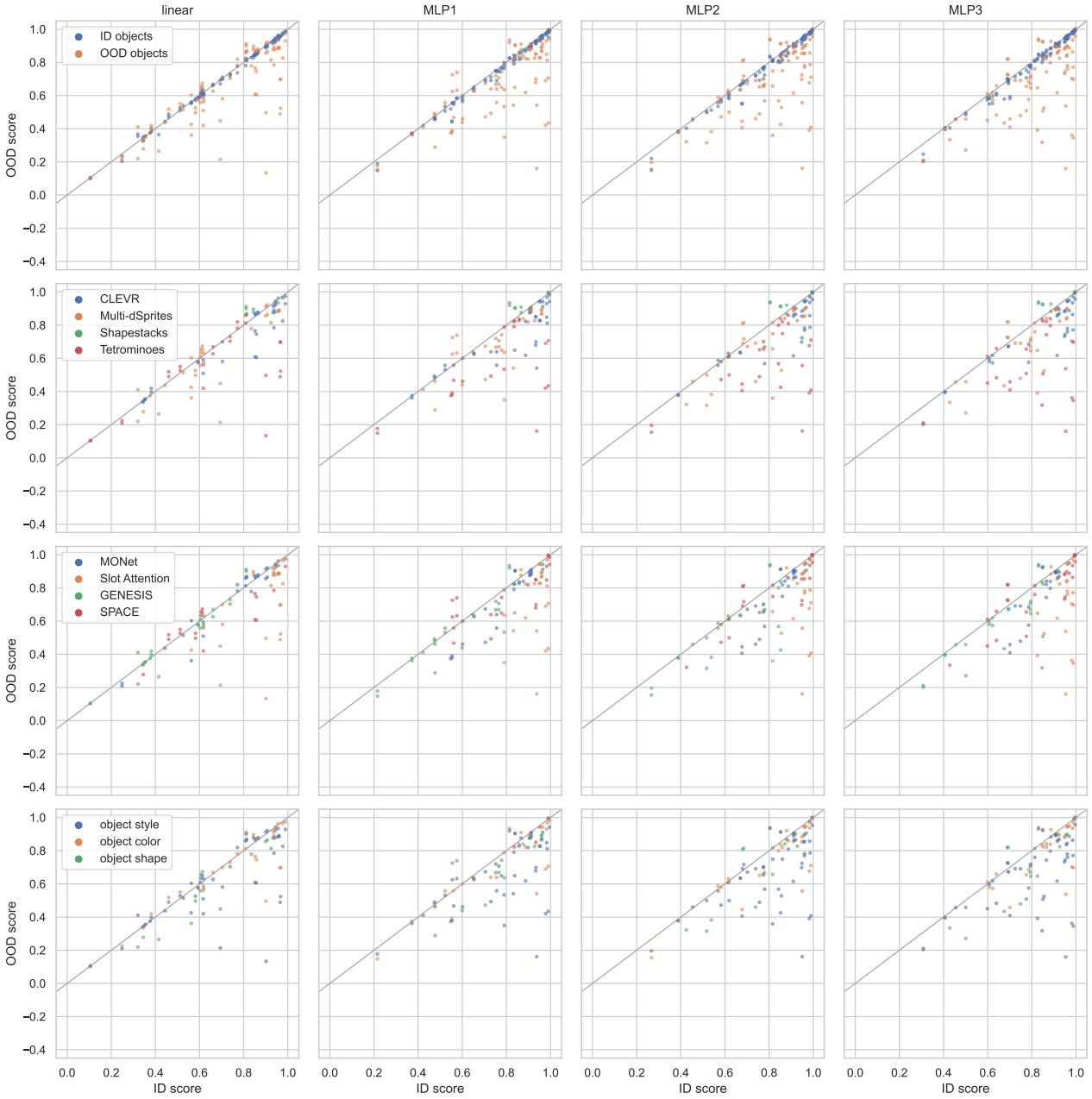


Figure 20: Generalization of **object-centric representations** in downstream prediction, using **loss matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

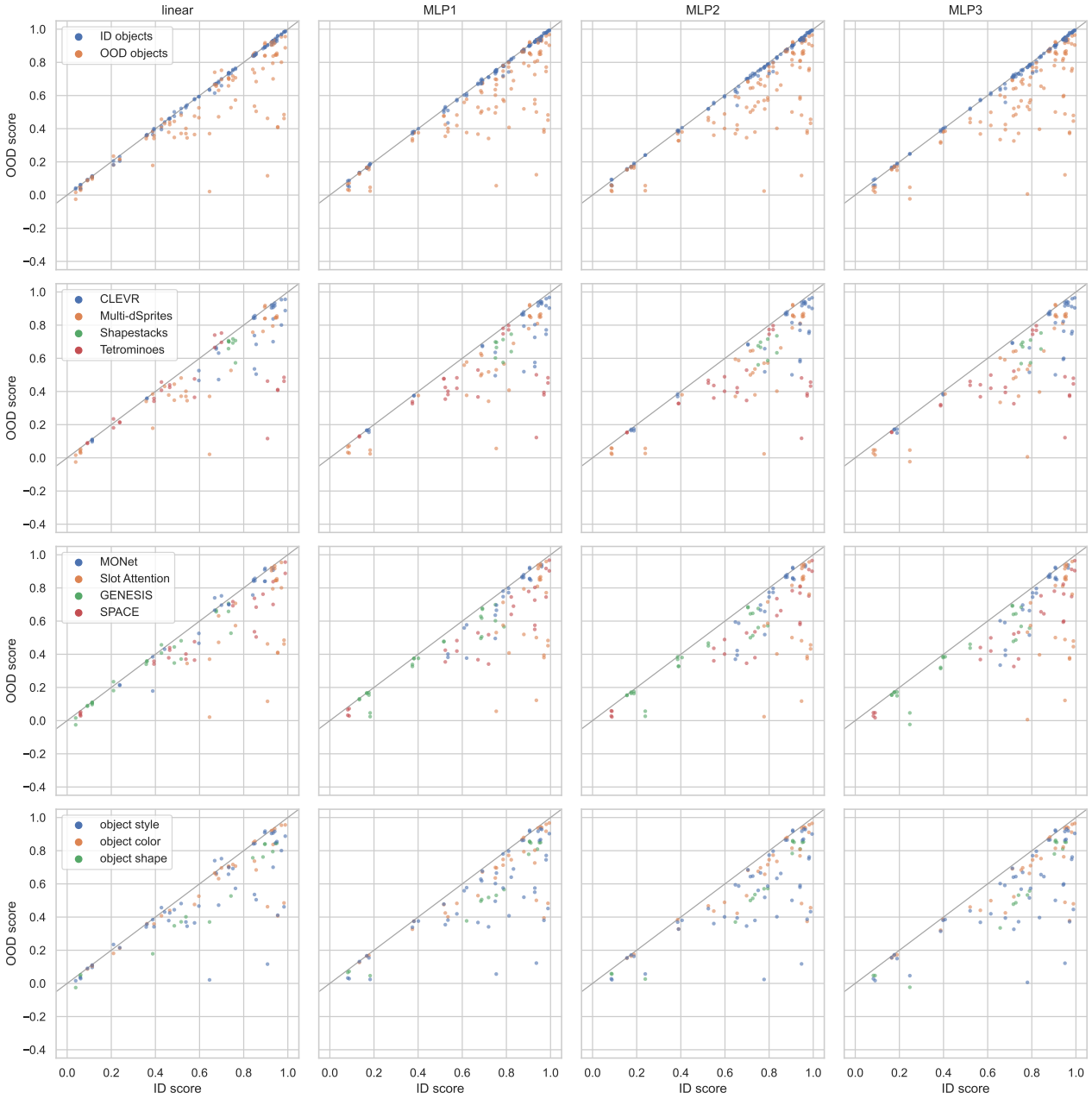


Figure 21: Generalization of **object-centric representations** in downstream prediction, using **mask matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

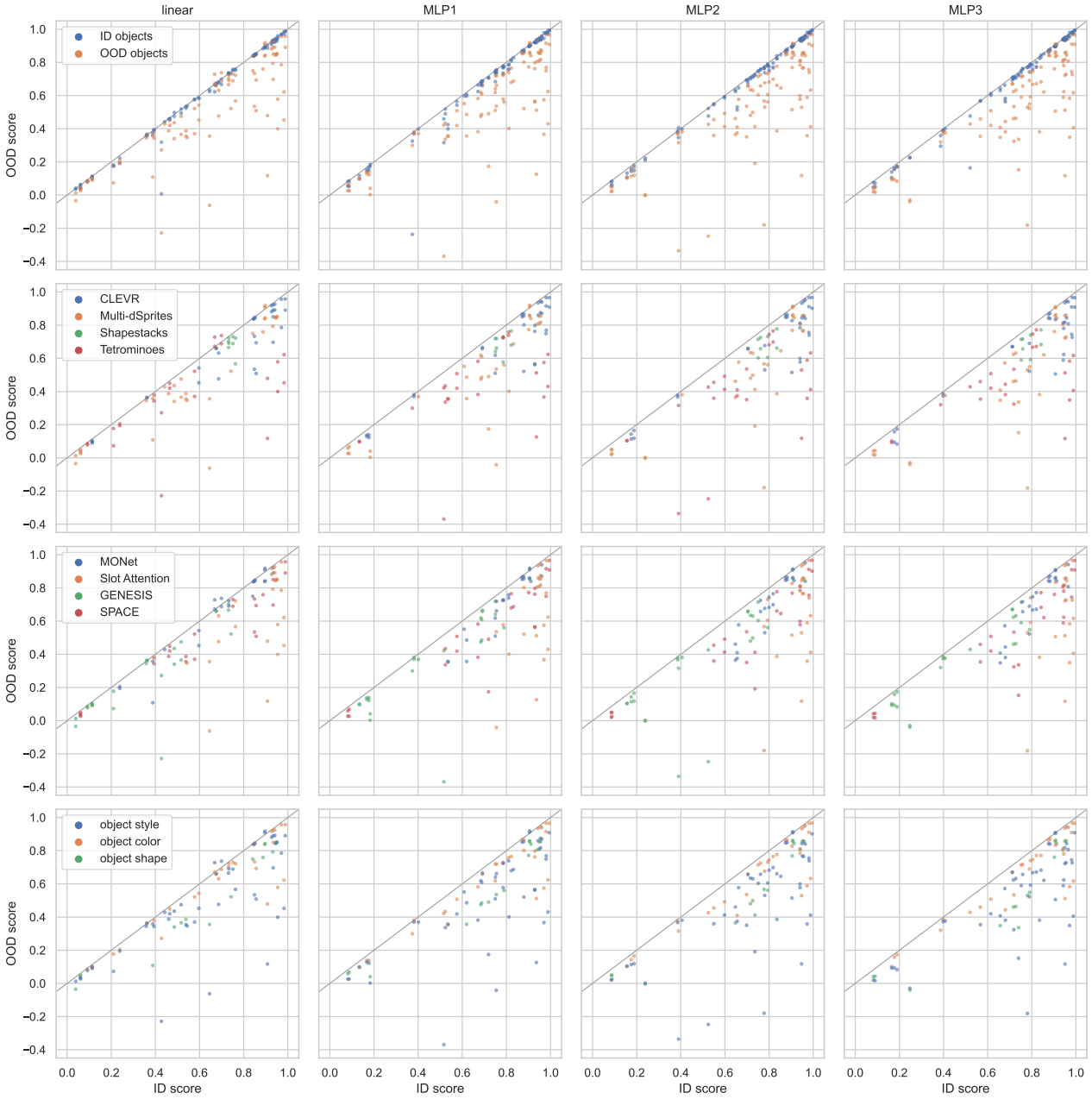


Figure 22: Generalization of **object-centric representations** in downstream prediction, using **mask matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

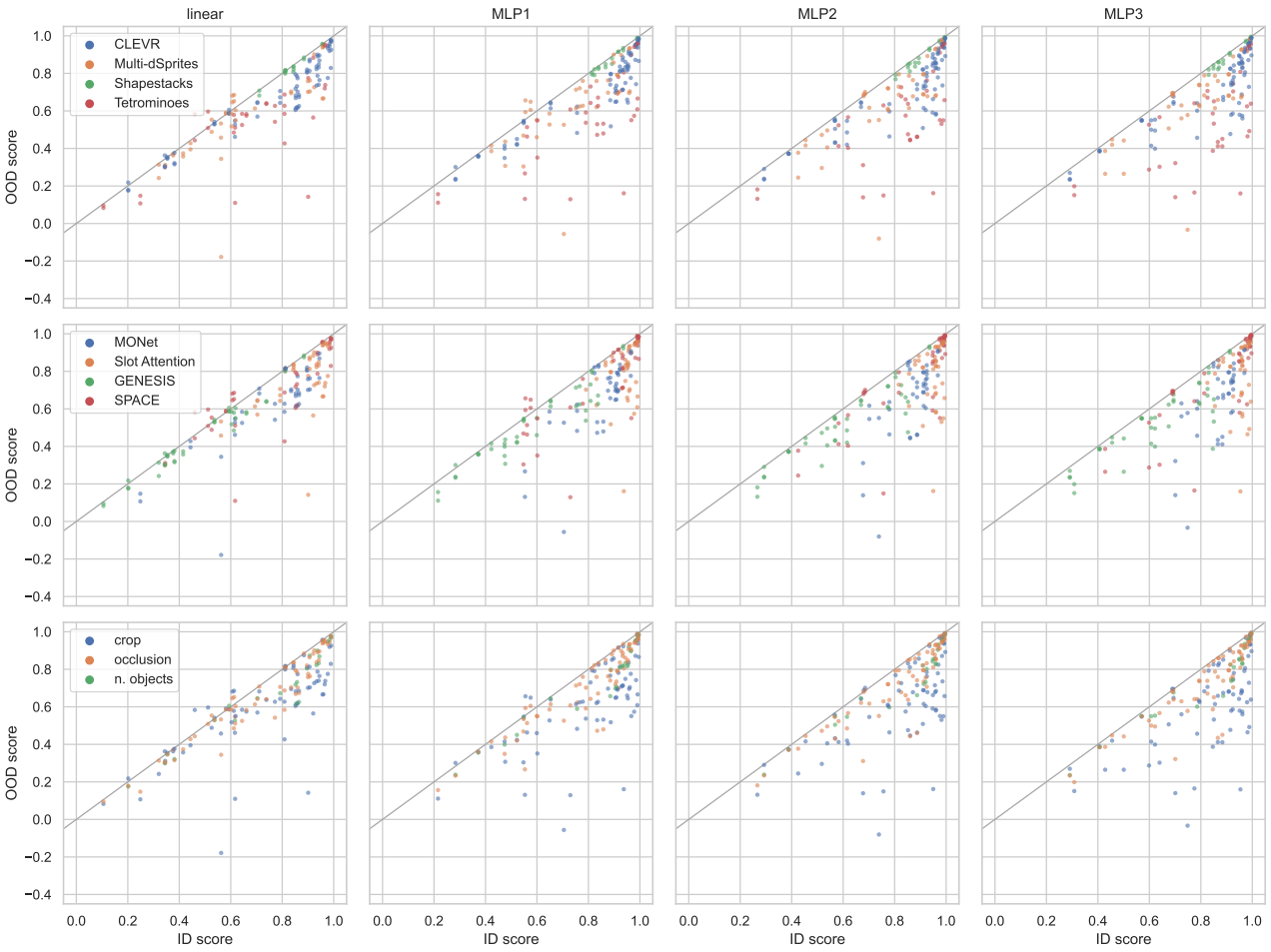


Figure 23: Generalization of **object-centric representations** in downstream prediction, using **loss matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

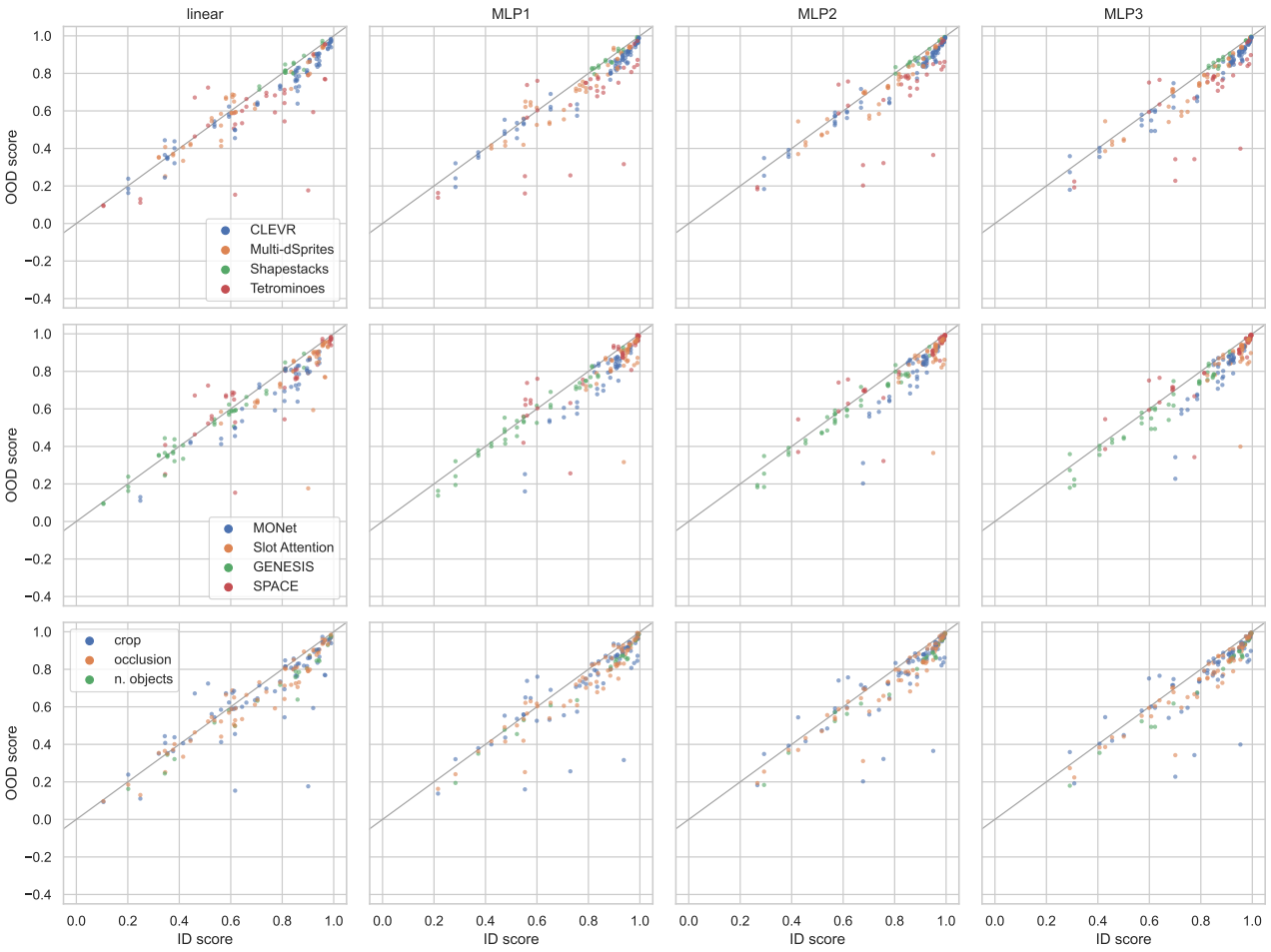


Figure 24: Generalization of **object-centric representations** in downstream prediction, using **loss matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

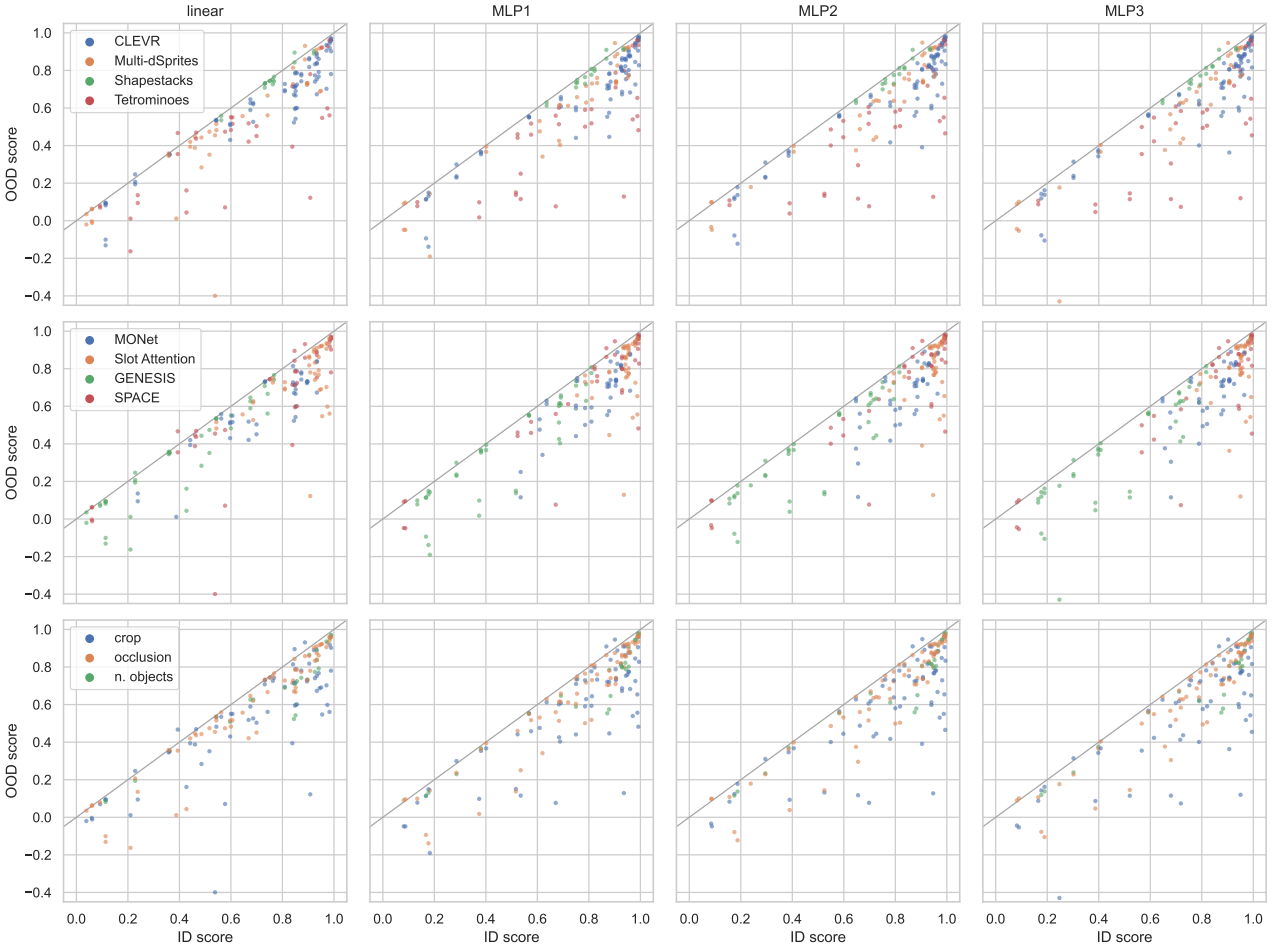


Figure 25: Generalization of **object-centric representations** in downstream prediction, using **mask matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

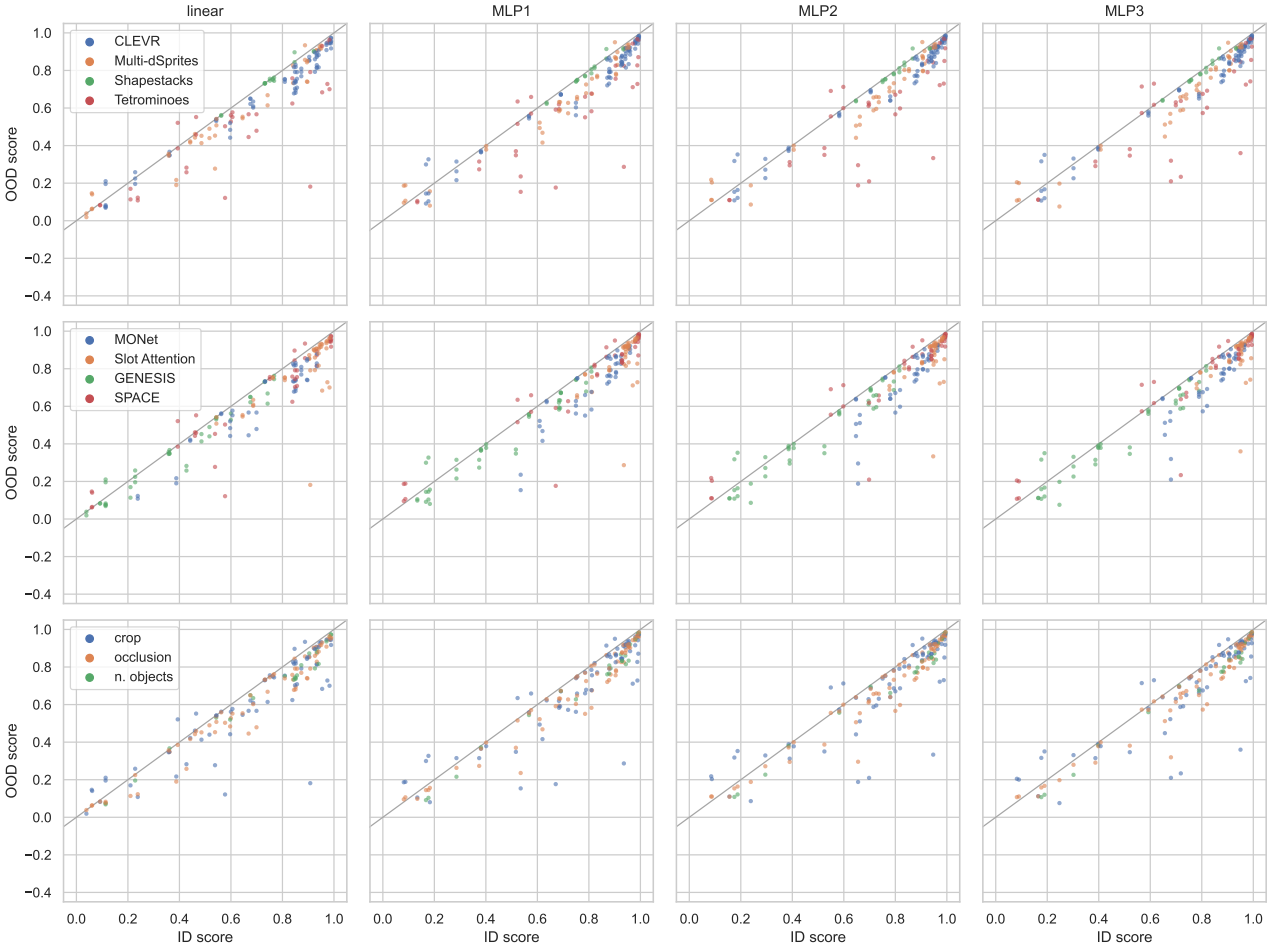


Figure 26: Generalization of **object-centric representations** in downstream prediction, using **mask matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

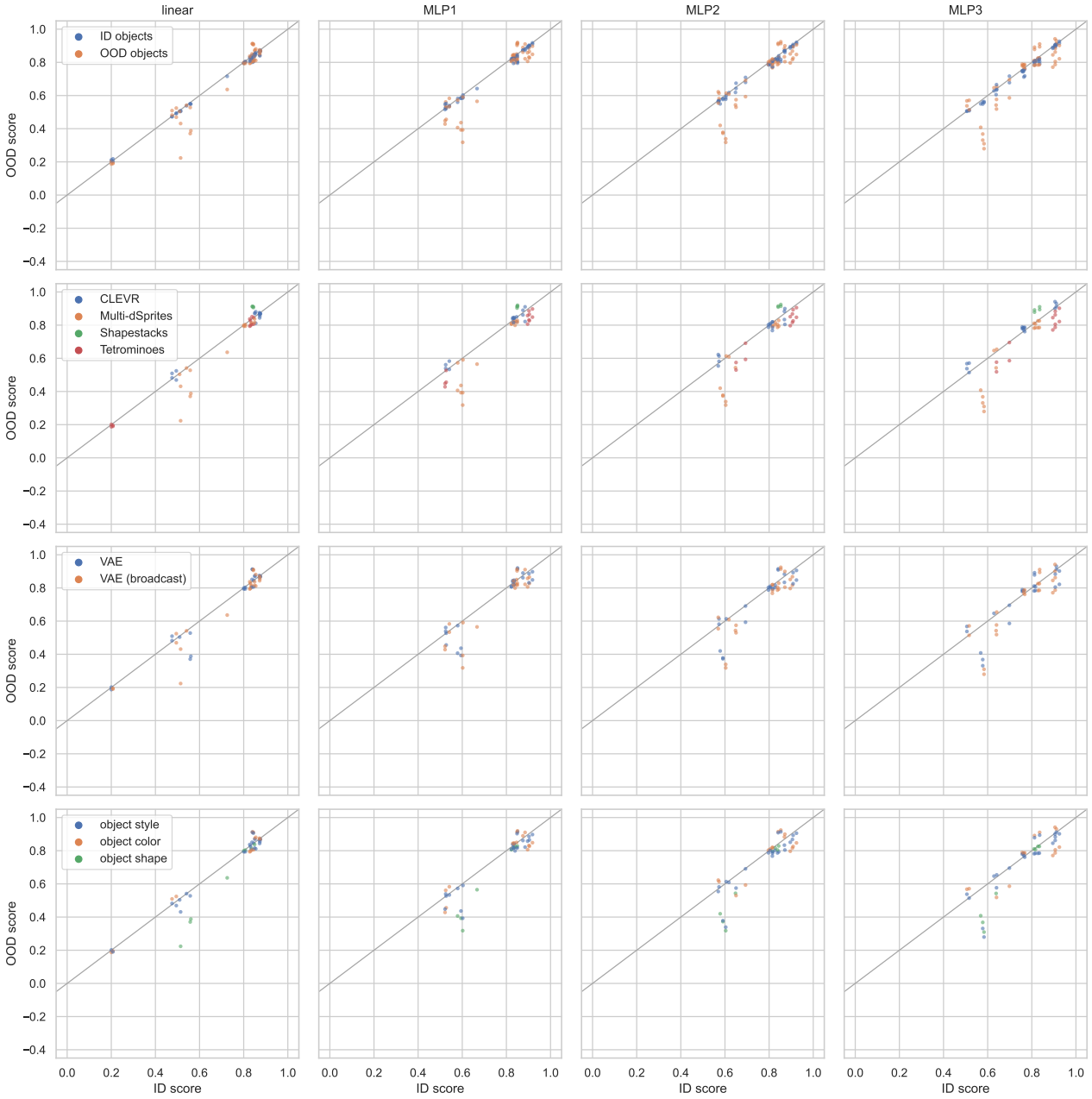


Figure 27: Generalization of **distributed representations** in downstream prediction, using **loss matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

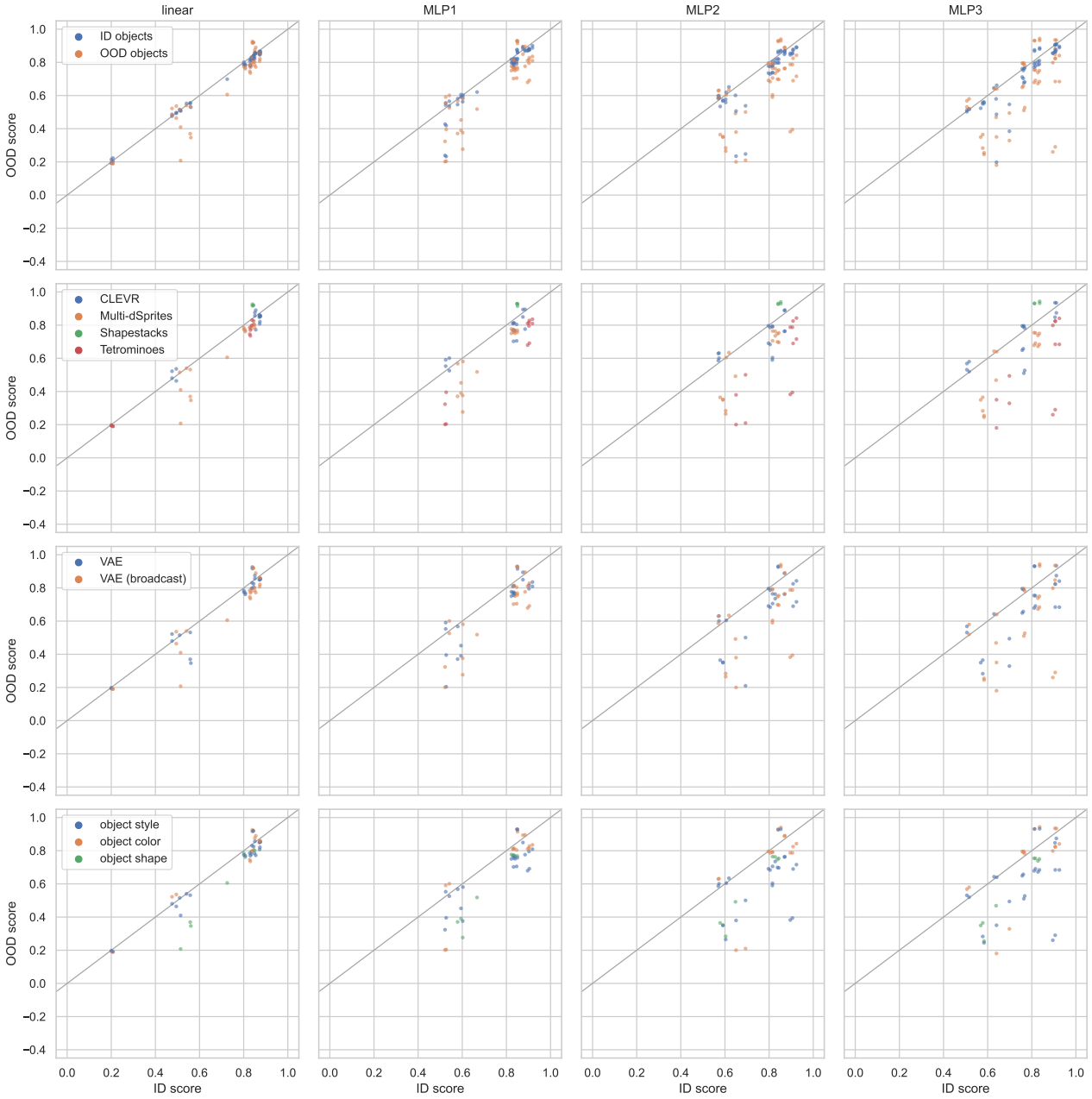


Figure 28: Generalization of **distributed representations** in downstream prediction, using **loss matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

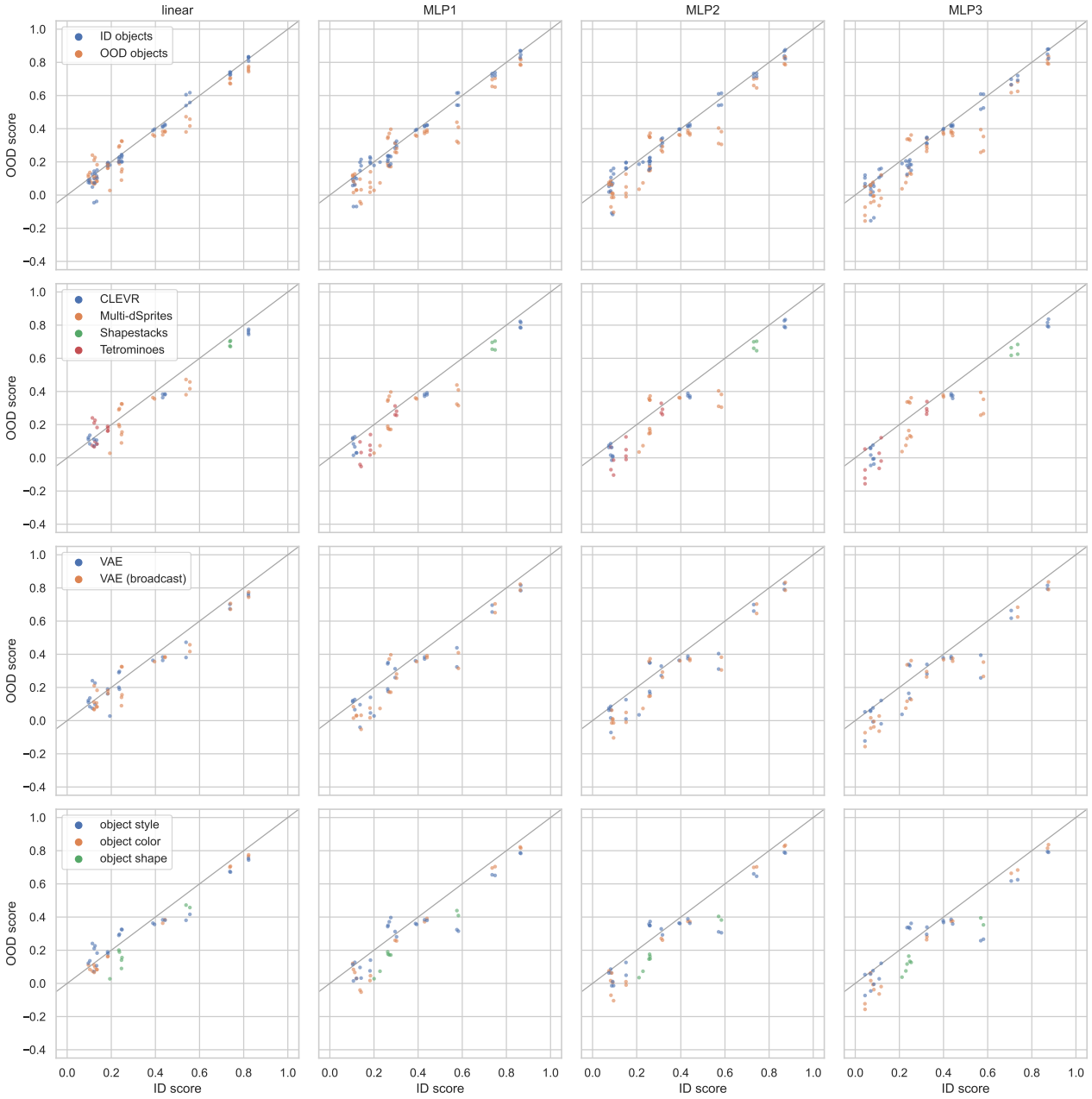


Figure 29: Generalization of **distributed representations** in downstream prediction, using **deterministic matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

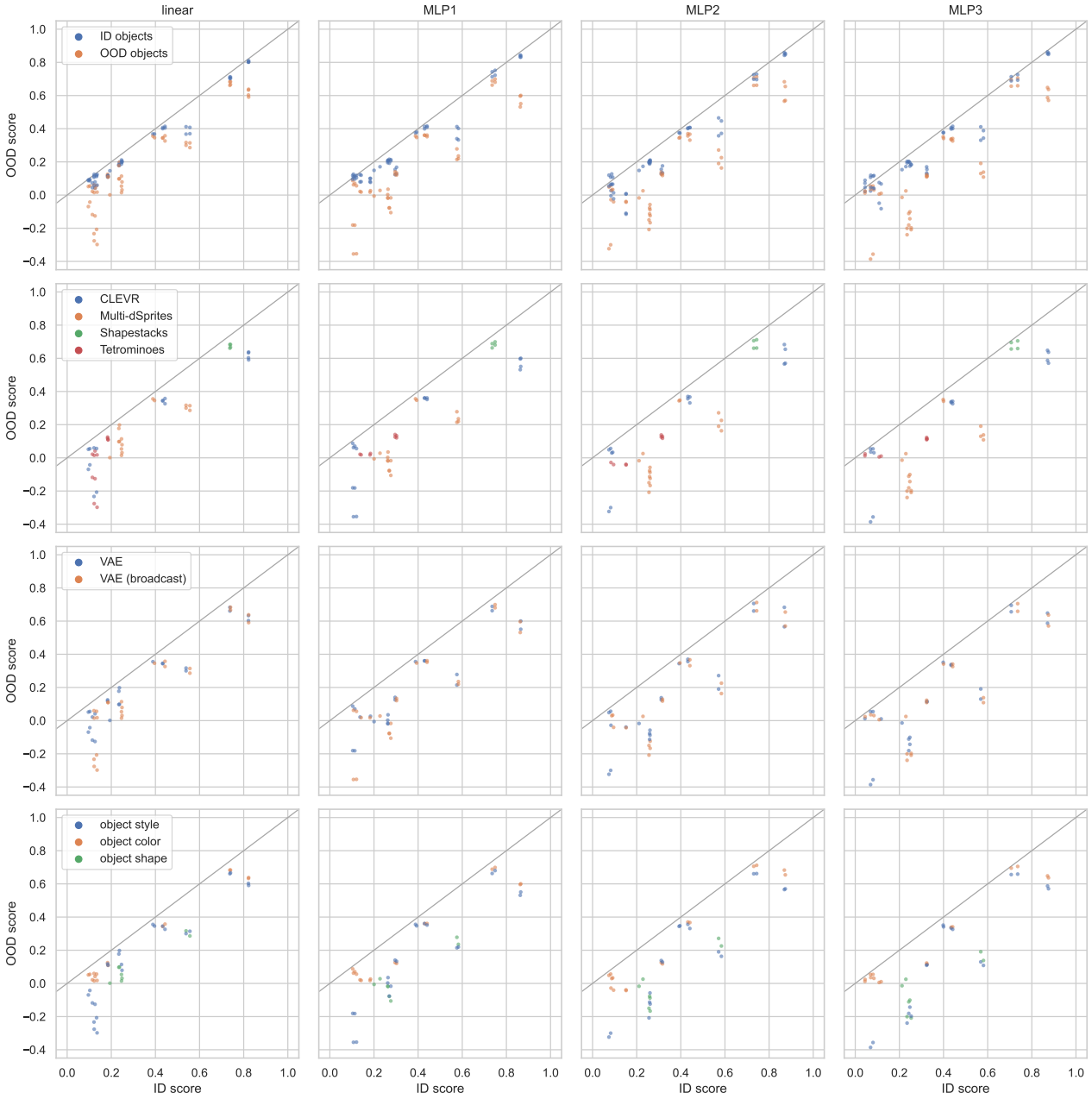


Figure 30: Generalization of **distributed representations** in downstream prediction, using **deterministic matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **one object**. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, one type of distribution shift, and either ID or OOD objects. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts and to ID/OOD objects. In the top row, we separately report (color-coded) the performance over ID and OOD objects. In the following rows, we only show OOD objects and split according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

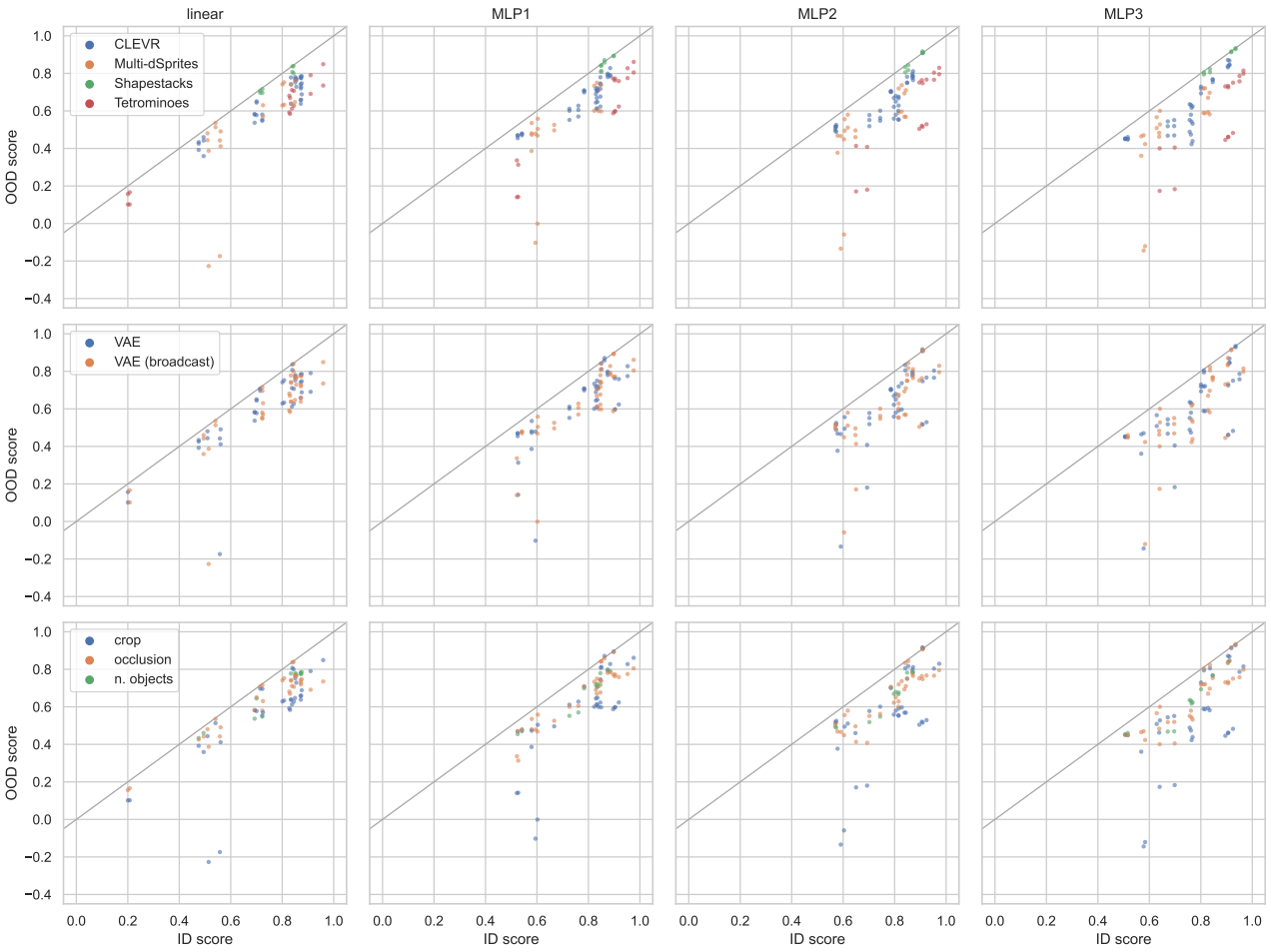


Figure 31: Generalization of **distributed representations** in downstream prediction, using **loss matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

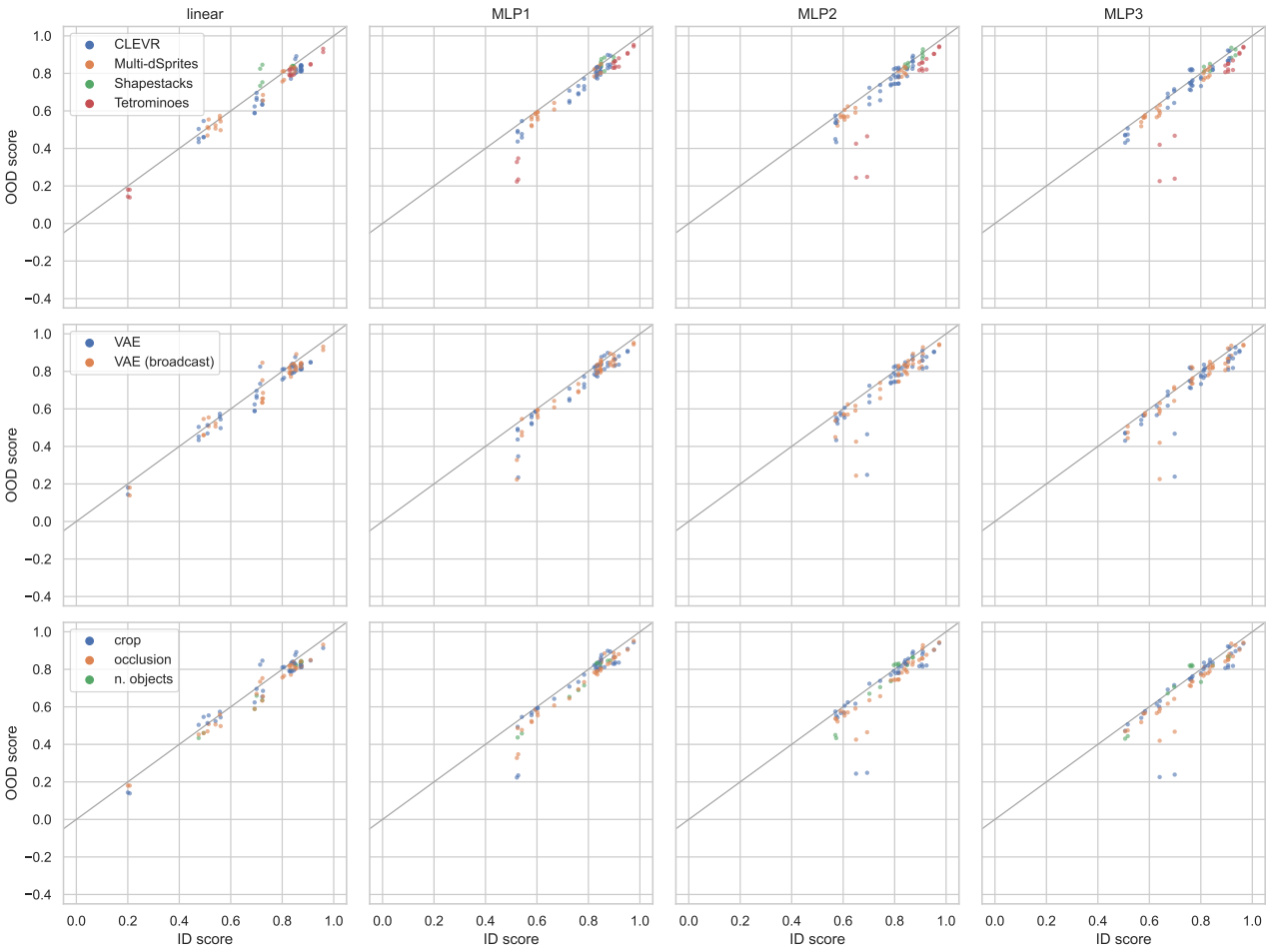


Figure 32: Generalization of **distributed representations** in downstream prediction, using **loss matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

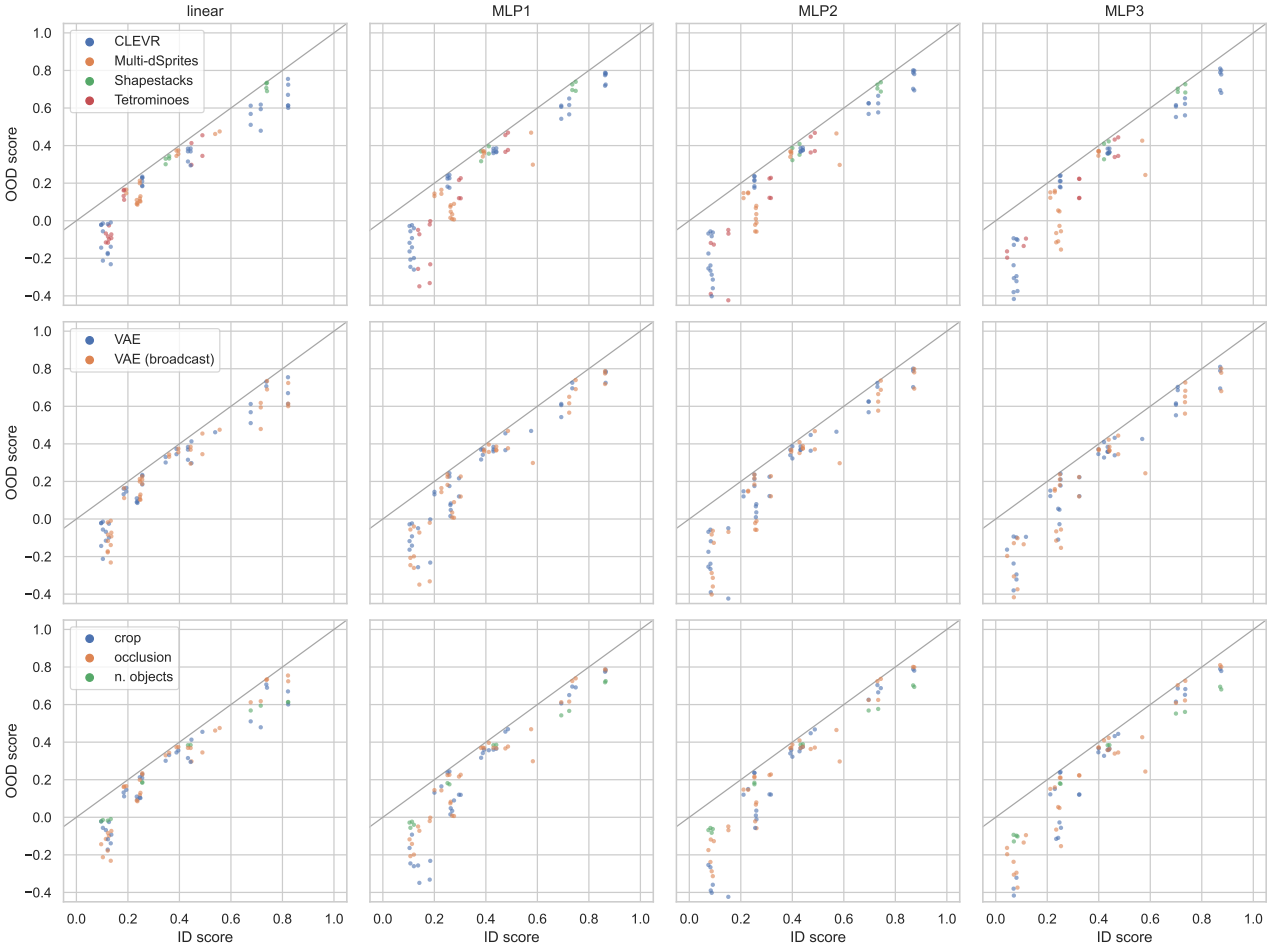


Figure 33: Generalization of **distributed representations** in downstream prediction, using **deterministic matching** and **without retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

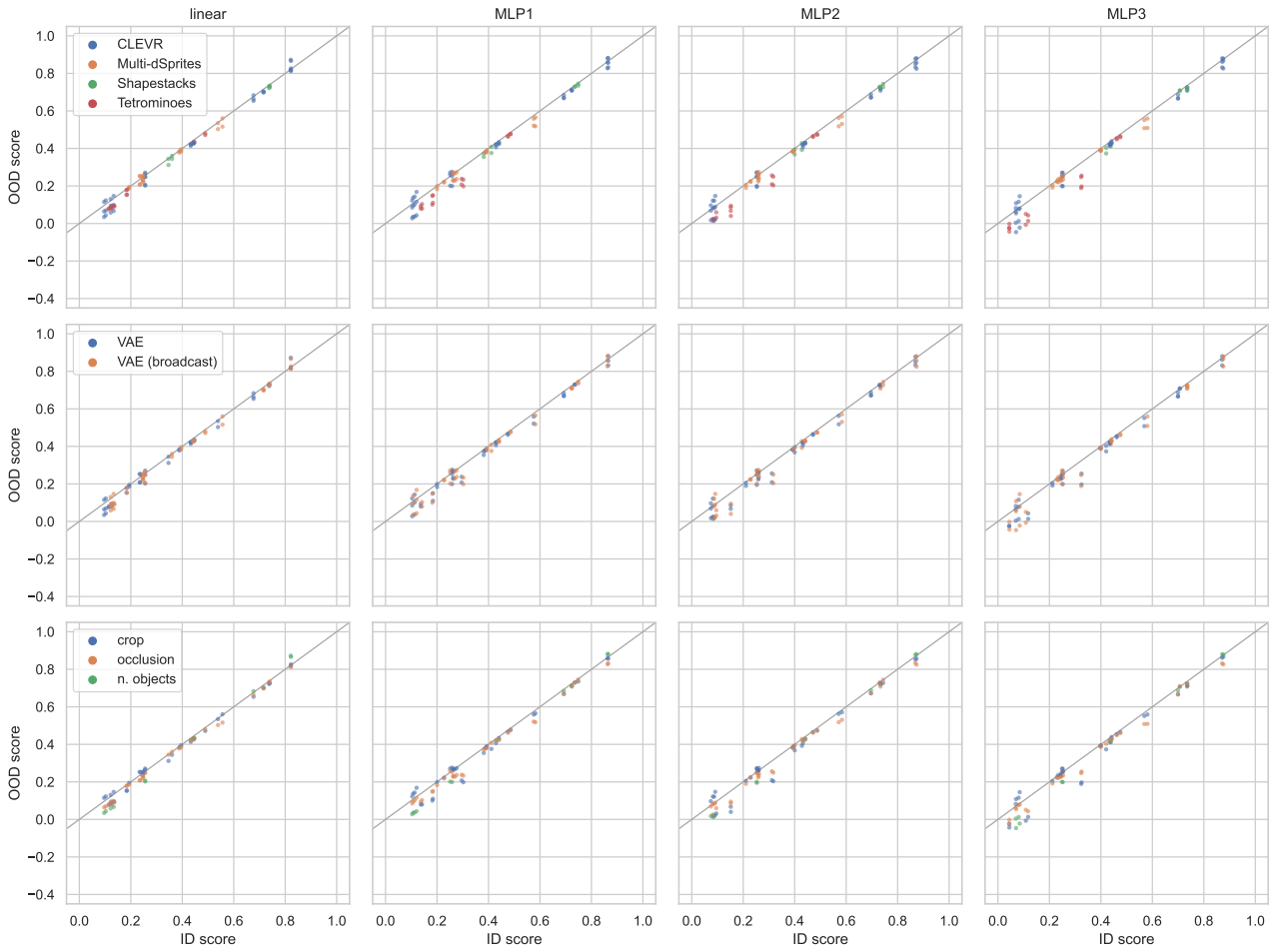


Figure 34: Generalization of **distributed representations** in downstream prediction, using **deterministic matching** and **retraining** the downstream model after the distribution shift. Here the distribution shift affects **global properties** of the scene. On the x-axis: prediction performance (accuracy or $R^2$) for one object property on one dataset, averaged over all objects, on the original training set of the unsupervised object discovery model. On the y-axis: the same metric in OOD scenarios. Each data point corresponds to one representation model (e.g., MONet), one dataset, one object property, and one type of distribution shift. For each x (performance on one object feature in the training distribution, averaged over objects in a scene and over random seeds of the object-centric models) there are multiple y's, corresponding to different distribution shifts. In each row, we color-code the data according to dataset, model, or type of distribution shift. Each column shows analogous results for each of the 4 considered downstream models for property prediction (linear, and MLPs with up to 3 hidden layers).

### D.3. Qualitative results

In Figs. 35 to 39 we show the reconstruction and segmentation performance of a selection of object-centric models on a random subset of held-out test images, for all 5 datasets. We select one object-centric model per type (MONet, Slot Attention, GENESIS, and SPACE) based on the ARI score on the validation set. The images we show were not used for model selection. For each model we show the following:

- *Input and reconstructed images*.
- *Ground-truth and inferred segmentation maps*. Here we use a set of 8 colors and assign each object (or slot) to a color. If there are more than 8 slots, we loop over the 8 colors again (this does not happen here, except in SPACE, where it is not an issue in practice). Rather than taking hard masks, we treat the masks as "soft", such that a pixel's color is a weighted mean of the 8 colors according to the masks. This is evident in Slot Attention, which typically splits the background smoothly across slots (consistently with the qualitative results shown in Locatello et al. (2020)). For clarity, we match (with the Hungarian algorithm) the colors of the ground-truth and predicted masks using the cosine distance (1 minus the cosine similarity) between masks.
- *Slot-wise reconstructions*. Each column corresponds to a slot in the object-centric representation of the model. Here we show the entire slot reconstruction with the inferred slot mask as alpha (transparency) channel. The overall reconstruction is the sum of these images. Since SPACE has in total up to 69 slots in our experiments ($K = 5$ background slots, and a grid of foreground slots of size $G \times G$ with $G = 8$), it is impractical to show all slots here. We choose instead to show the 10 most salient slots, selected according to the average mask value over the image. This number is sufficient as most slots are unused. When selecting slots this way, the selected slots are shown in their original order (in SPACE, the background slots are appended to the foreground slots).

For completeness, in Fig. 40 we show inputs and reconstructions for one VAE baseline per type (convolutional and broadcast decoder), selected using the reconstruction MSE on the validation set.

Finally, Figs. 41 to 45 show input–reconstruction pairs for each dataset, model type, and distribution shift. Note that the comparison is not necessarily fair, since object-centric models were chosen using the validation ARI on the training distribution, while VAEs were chosen in a similar way but using the MSE. However, these qualitative results can still be highly informative. We report some examples:

- Most object-centric models are relatively robust to shifts affecting a single object, as discussed in the main text based on quantitative results.
- On the other hand, they are often not robust to global shifts, especially when cropping and enlarging the scene.
- MONet achieves relatively good reconstructions even out of distribution, probably because images are segmented mostly based on color. This was suggested by Papa et al. (2022), where the models are trained on objects with style transfer. However, we conjecture the behavior may be the same in our case, and that the argument should also apply to other distribution shifts, as seen by the relatively accurate reconstructions under both single-object and global distribution shifts. Note that, while reconstructions are potentially more accurate than for other models, this does not mean that MONet has segmented the object correctly.
- Although its ARI score does not decrease significantly, Slot Attention may not always handle more objects than in the training distribution, even when the number of slots in the model is increased. This is consistent with the results reported by Locatello et al. (2020), and increasing the number of Slot Attention iterations at test time seems to be a promising approach (Locatello et al., 2020, Fig. 2).
- VAEs seem to be relatively good at generalizing to a greater number of objects in CLEVR. In particular, they reconstruct images with the correct number of objects, although a few details of the objects may not be inferred correctly (e.g. an object may be reconstructed with the wrong size, color, or shape). This is surprising, since VAEs do not have any inductive bias for this, and the fact that the encoder is OOD (i.e., the encoder input is OOD w.r.t. the distribution used to train the encoder itself) might lead us to expect poor generalization capabilities, as discussed by Dittadi et al. (2021) and Träuble et al. (2022) in the "OOD2" case. On the other hand, some object-centric models are remarkably robust to this shift (in particular SPACE, as confirmed by the ARI in Fig. 8).

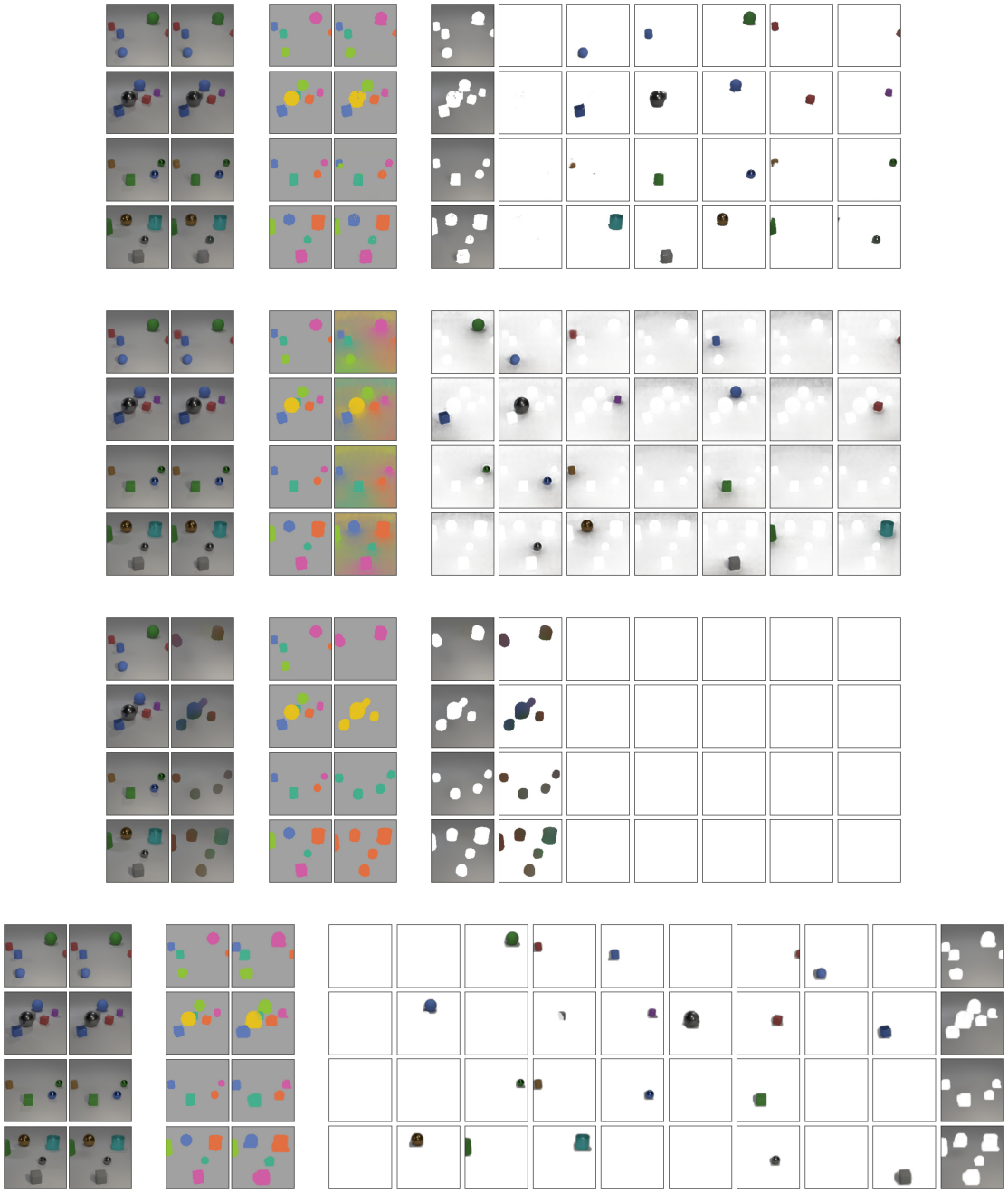

Figure 35: **Reconstruction and segmentation** of 4 random images from the held-out test set of **CLEVR6**. Top to bottom: MONet, Slot Attention, GENESIS, SPACE. Left to right: input, reconstruction, ground-truth masks, predicted (soft) masks, slot-wise reconstructions (masked with the predicted masks). As explained in the text, for SPACE we select the 10 most salient slots using the predicted masks. As explained in the text, for SPACE we select the 10 most salient slots using the predicted masks. For each model type, we visualize the specific model with the highest ARI score in the *validation* set. The images shown here are from the *test* set and were not used for model selection.

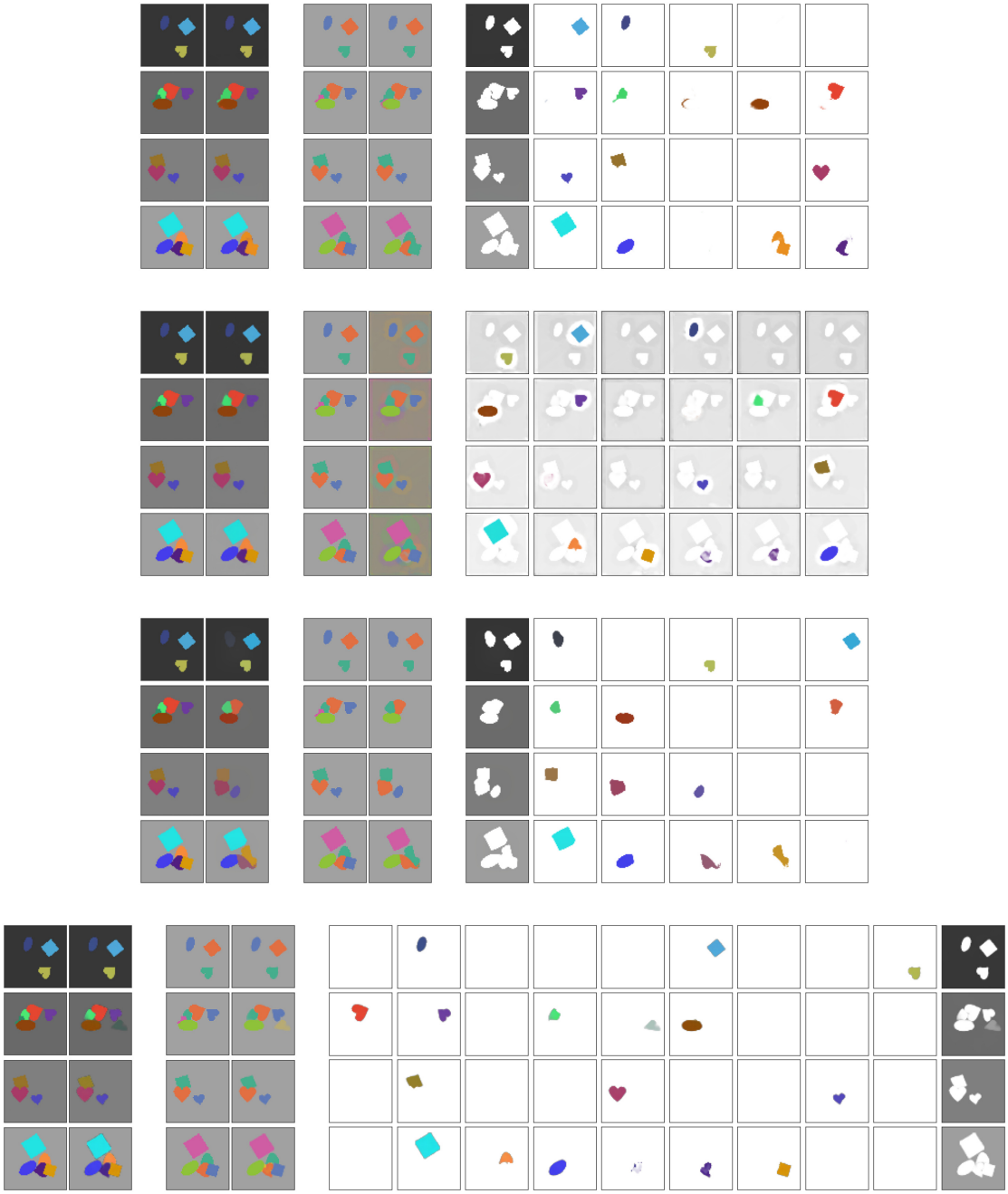

Figure 36: **Reconstruction and segmentation** of 4 random images from the held-out test set of **Multi-dSprites**. Top to bottom: MONet, Slot Attention, GENESIS, SPACE. Left to right: input, reconstruction, ground-truth masks, predicted (soft) masks, slot-wise reconstructions (masked with the predicted masks). As explained in the text, for SPACE we select the 10 most salient slots using the predicted masks. For each model type, we visualize the specific model with the highest ARI score in the *validation* set. The images shown here are from the *test* set and were not used for model selection.

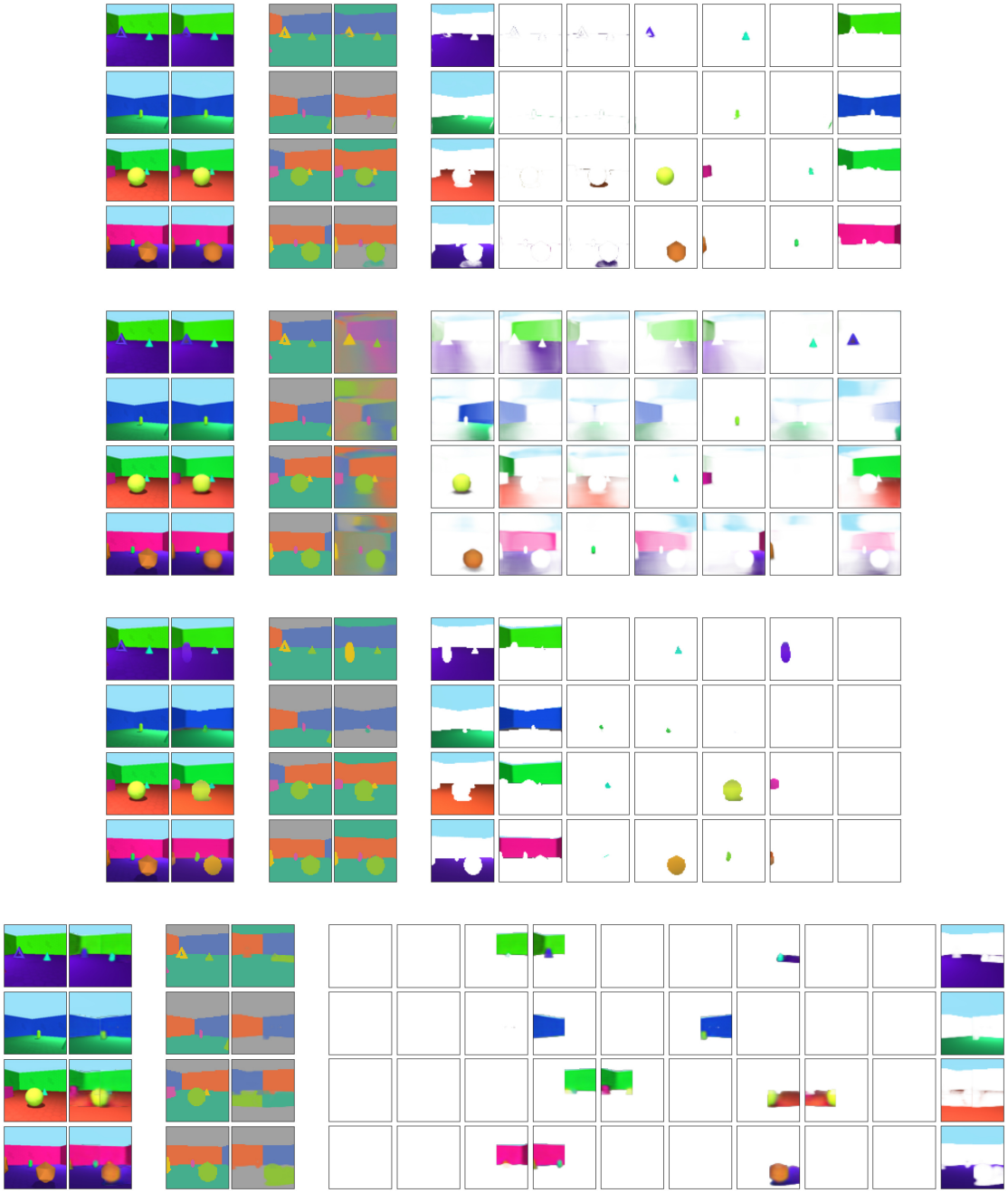

Figure 37: **Reconstruction and segmentation** of 4 random images from the held-out test set of **Objects Room**. Top to bottom: MONet, Slot Attention, GENESIS, SPACE. Left to right: input, reconstruction, ground-truth masks, predicted (soft) masks, slot-wise reconstructions (masked with the predicted masks). As explained in the text, for SPACE we select the 10 most salient slots using the predicted masks. For each model type, we visualize the specific model with the highest ARI score in the *validation* set. The images shown here are from the *test* set and were not used for model selection.

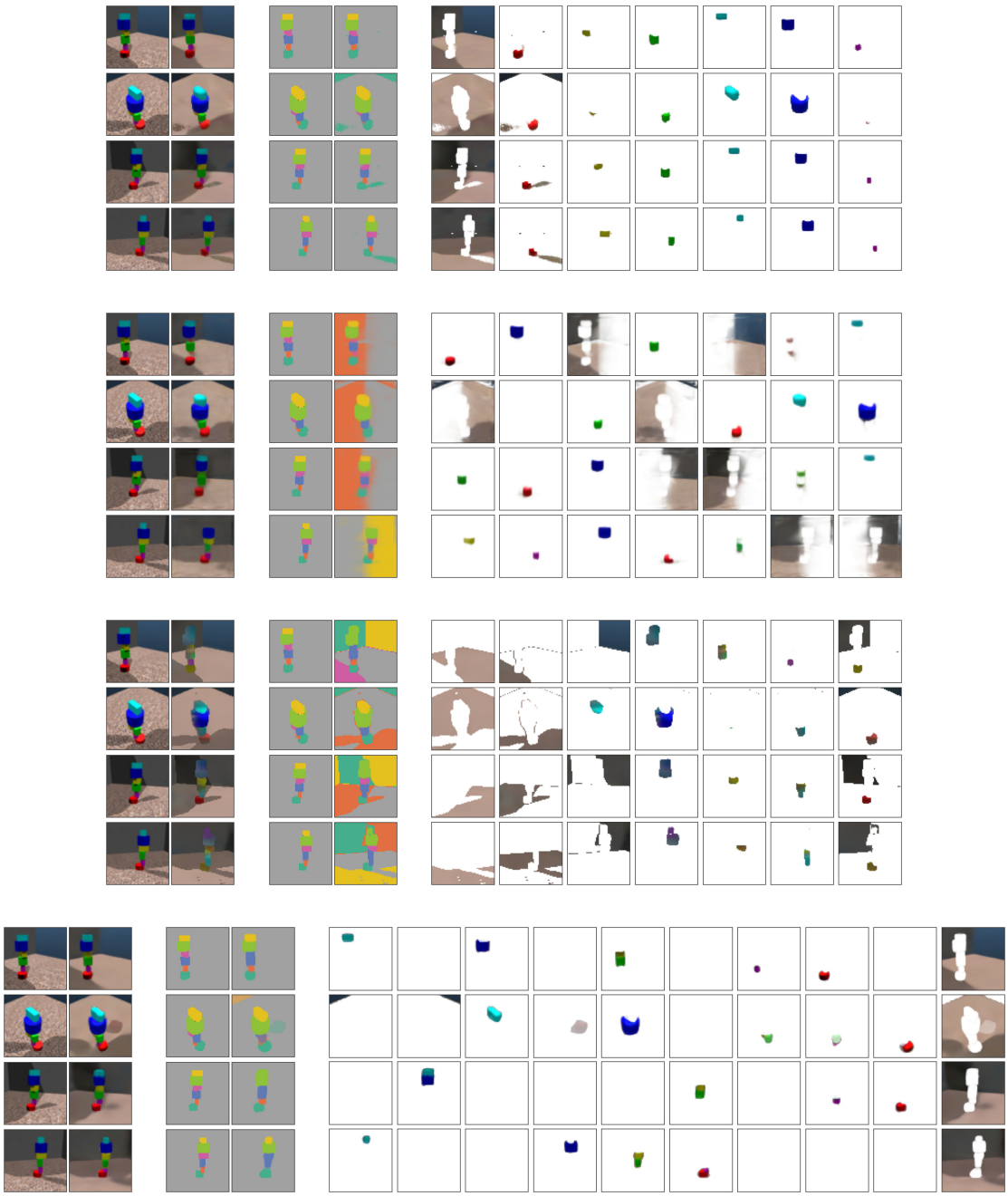

Figure 38: **Reconstruction and segmentation** of 4 random images from the held-out test set of **Shapestacks**. Top to bottom: MONet, Slot Attention, GENESIS, SPACE. Left to right: input, reconstruction, ground-truth masks, predicted (soft) masks, slot-wise reconstructions (masked with the predicted masks). As explained in the text, for SPACE we select the 10 most salient slots using the predicted masks. For each model type, we visualize the specific model with the highest ARI score in the *validation* set. The images shown here are from the *test* set and were not used for model selection.

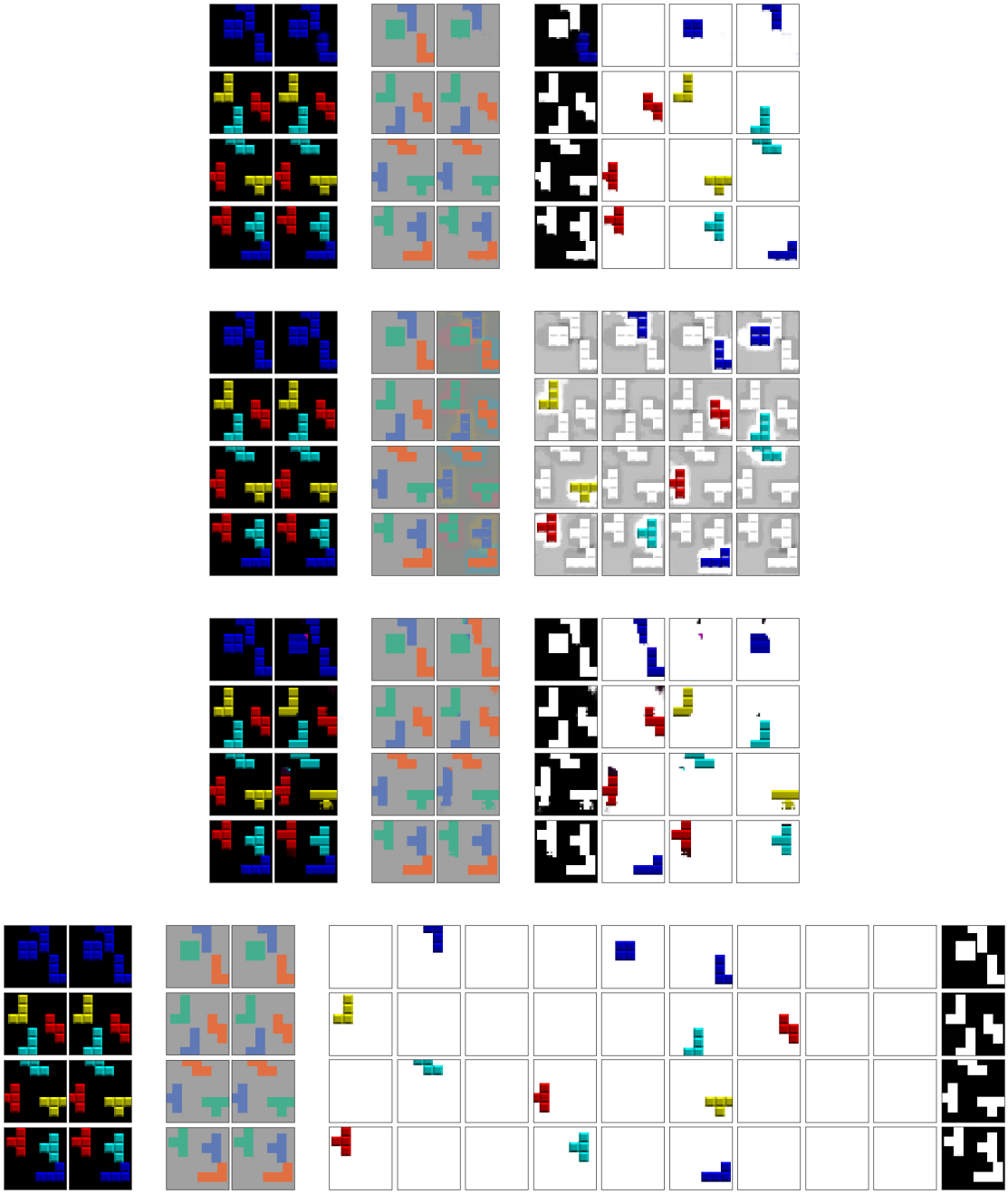

Figure 39: **Reconstruction and segmentation** of 4 random images from the held-out test set of **Tetrominoes**. Top to bottom: MONet, Slot Attention, GENESIS, SPACE. Left to right: input, reconstruction, ground-truth masks, predicted (soft) masks, slot-wise reconstructions (masked with the predicted masks). As explained in the text, for SPACE we select the 10 most salient slots using the predicted masks. For each model type, we visualize the specific model with the highest ARI score in the *validation* set. The images shown here are from the *test* set and were not used for model selection.

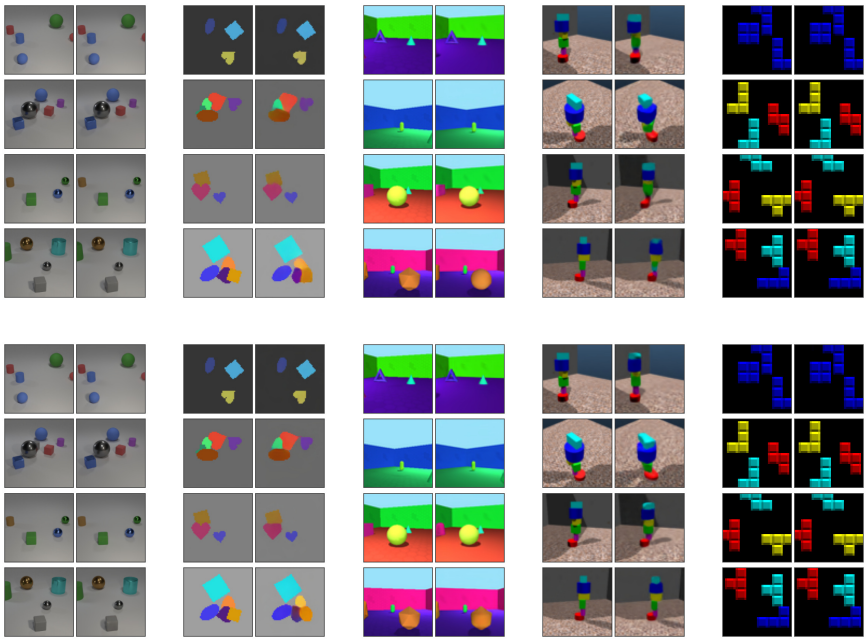

Figure 40: **Input–reconstruction pairs** of 4 random images from the held-out *test* set of all 5 datasets, for the VAE model with convolutional (top) and broadcast (bottom) decoder. Each VAE type was trained with 5 random seeds, and for each type we show here the model with the lowest MSE on the *validation* set. The images shown here are from the *test* set and were not used for model selection. For each image, we show the input on the left and the reconstruction on the right. As these are not slot-based models, segmentation masks and slot-wise reconstructions are not available.

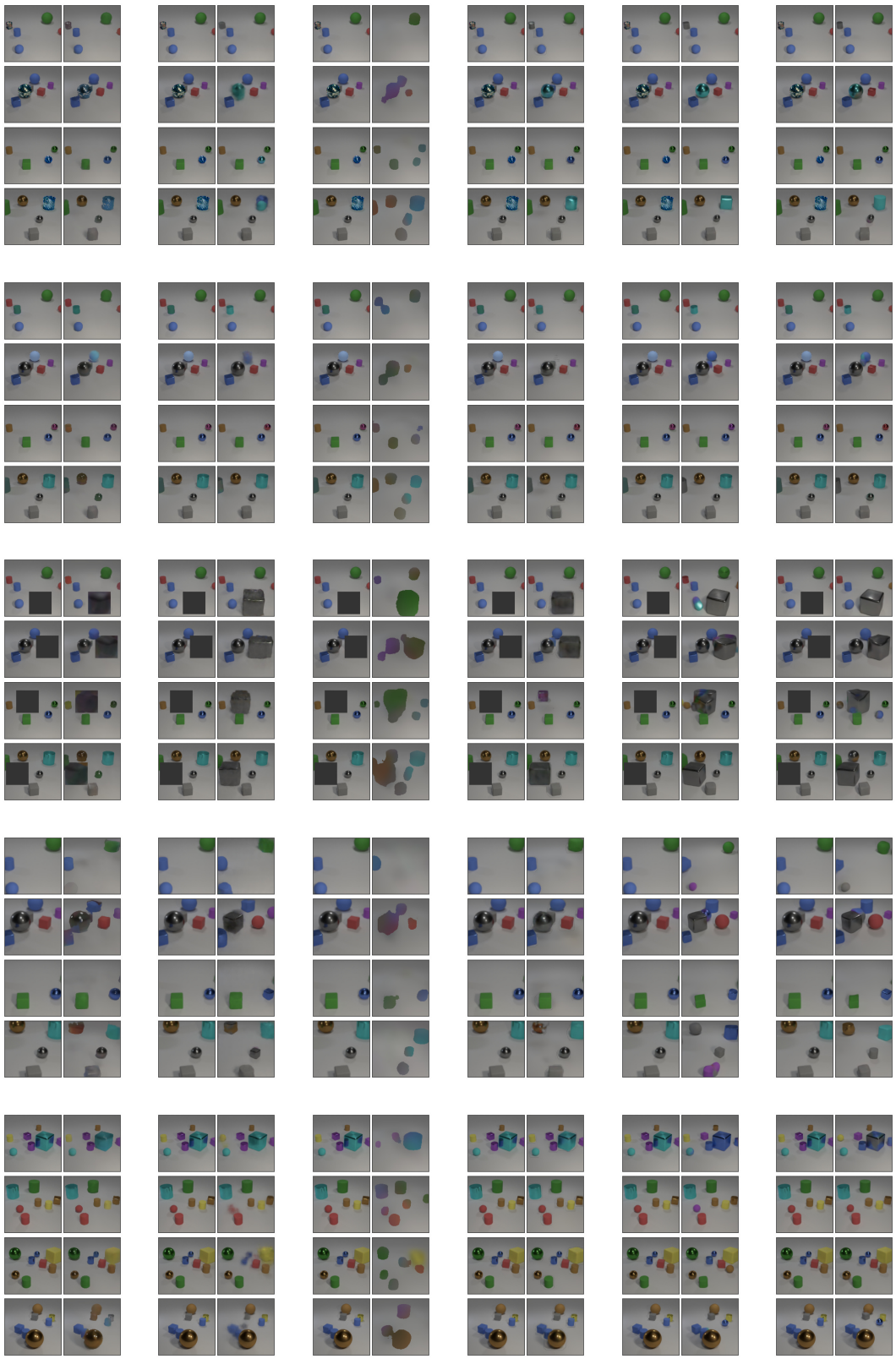

Figure 41: **Inputs and reconstructions for OOD images in CLEVR.** Columns from left to right: MONet, Slot Attention, GENESIS, SPACE, convolutional decoder VAE, broadcast decoder VAE. Rows from top to bottom: object style, object color, occlusion, crop, number of objects.

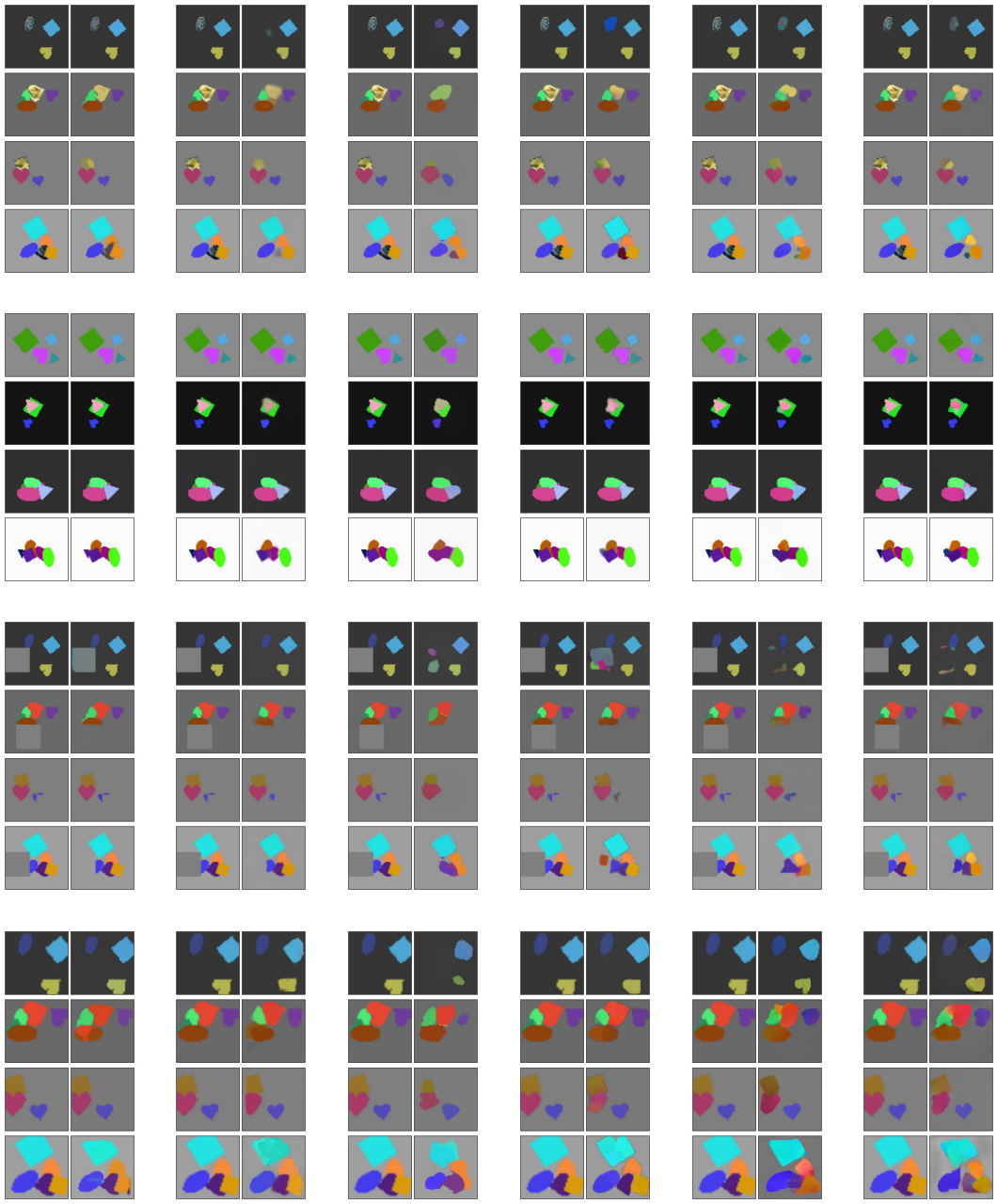

Figure 42: **Inputs and reconstructions for OOD images in Multi-dSprites.** Columns from left to right: MONet, Slot Attention, GENESIS, SPACE, convolutional decoder VAE, broadcast decoder VAE. Rows from top to bottom: object style, object shape, occlusion, crop.

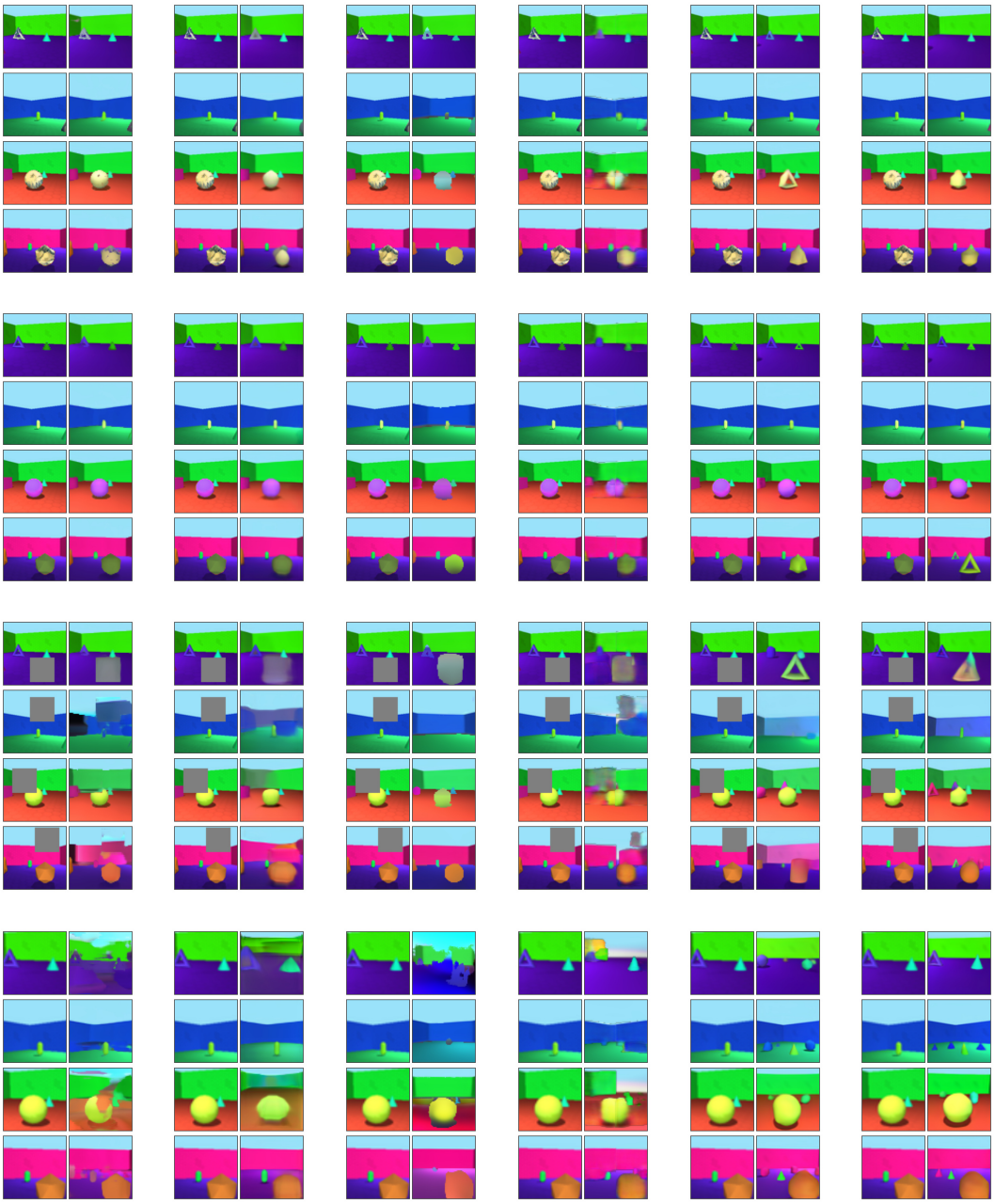

Figure 43: **Inputs and reconstructions for OOD images in Objects Room.** Columns from left to right: MONet, Slot Attention, GENESIS, SPACE, convolutional decoder VAE, broadcast decoder VAE. Rows from top to bottom: object style, object color, occlusion, crop.

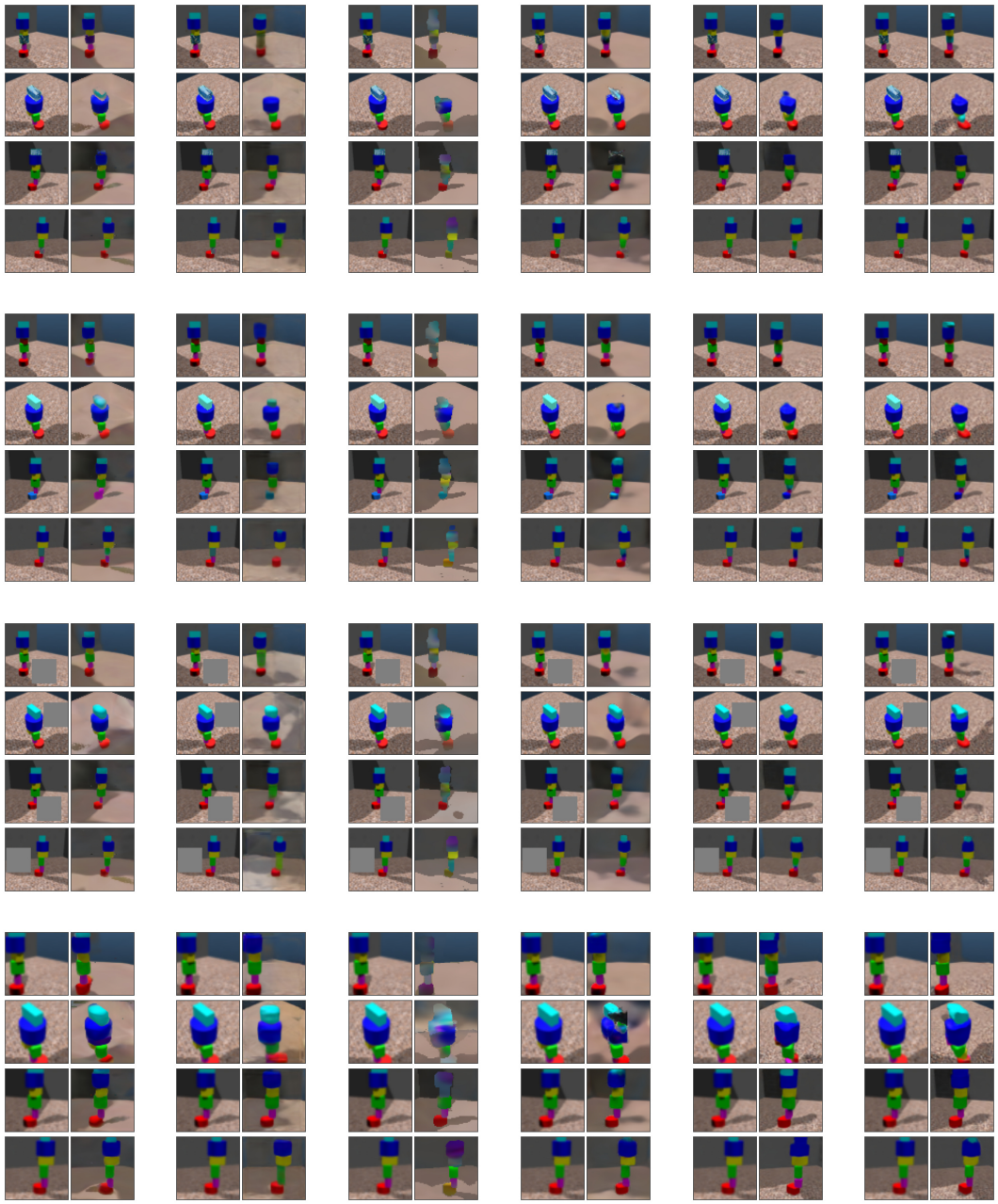

Figure 44: **Inputs and reconstructions for OOD images in Shapestacks.** Columns from left to right: MONet, Slot Attention, GENESIS, SPACE, convolutional decoder VAE, broadcast decoder VAE. Rows from top to bottom: object style, object color, occlusion, crop.

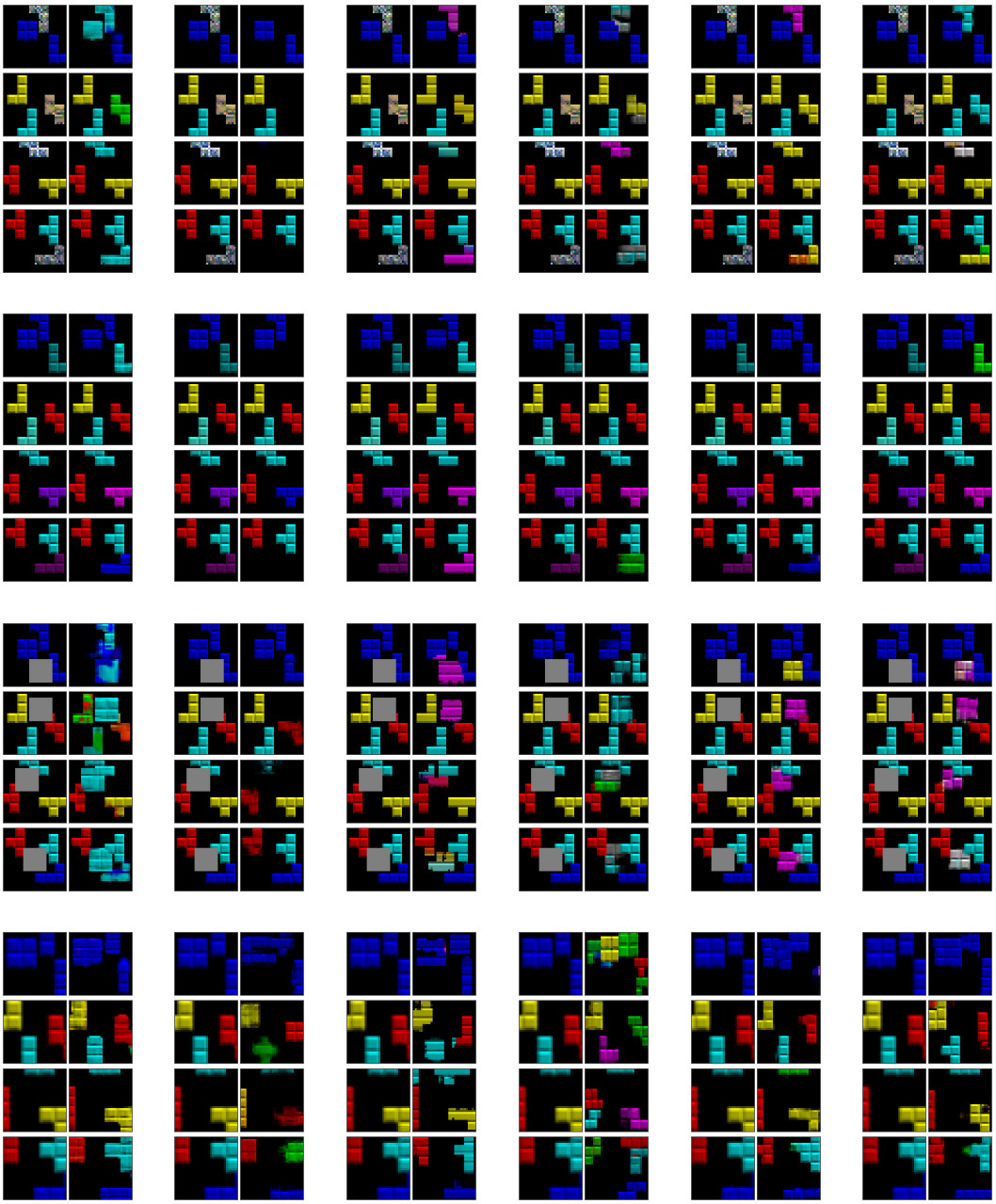

Figure 45: **Inputs and reconstructions for OOD images in Tetrominoes.** Columns from left to right: MONet, Slot Attention, GENESIS, SPACE, convolutional decoder VAE, broadcast decoder VAE. Rows from top to bottom: object style, object color, occlusion, crop.