Unlocking Slot Attention by Changing Optimal Transport Costs

Yan Zhang [*1] **David W. Zhang** [*2] **Simon Lacoste-Julien** [1 3 4] **Gertjan J. Burghouts** [5] **Cees G. M. Snoek** [2]

Abstract

Slot attention is a powerful method for object-centric modeling in images and videos. However, its set-equivariance limits its ability to handle videos with a dynamic number of objects because it cannot break ties. To overcome this limitation, we first establish a connection between slot attention and optimal transport. Based on this new perspective we propose **MESH** (Minimize Entropy of Sinkhorn): a cross-attention module that combines the tiebreaking properties of unregularized optimal transport with the speed of regularized optimal transport. We evaluate slot attention using MESH on multiple object-centric learning benchmarks and find significant improvements over slot attention in every setting.

1. Introduction

Suppose we have an image containing two cats and one dog. Given a query like [cat, cat, dog], our task is to provide instance-specific information for each query element, such as their positions in the image. When constructing a neural network to solve this problem, *cross-attention* is a natural choice to relate the queries to our image (Vaswani et al., 2017; Wei et al., 2020). With such a model, the dog can be located perfectly, but our two queries for the cats inevitably end up with an undesirable result: the *average* of the two cats' positions. The problem is that with our model, multiple copies of the same query element *must* have the same result (Zhang et al., 2022); it is impossible to receive different answers for the same query, even if the context makes it obvious what is desired.

Models that rely on cross-attention, such as slot attention (Locatello et al., 2020), can run into this issue when trying to extract objects from images and other data modalities. This has especially been a problem in the video domain due to occlusions and new objects appearing (Kipf et al., 2022). For example, Wu et al. (2022) observe in experiments that when there are two objects but five "slots" (equivalent to queries in the previous example), the three slots that do not have a specific object to model become nearly identical. Once that has happened, the "cat problem" from before applies: these very similar slots (queries) must receive essentially the same information. Consequently, if multiple new objects appear in the video, these three slots can—at best—all bind to only a single object, which is clearly undesirable.

This issue is present because cross-attention is set-equivariant, a property traditionally considered desirable. However, Zhang et al. (2022) show that set-equivariance is too restrictive when applied to *multisets*: sets with repeated elements allowed, which are prevalent in deep learning (background in Section 2). This manifests itself in two related problems:

1. **Soft assignments.** The model tends to mix several inputs into each slot (query) rather than making a hard decision of one slot corresponding to exactly one input. This leads to difficulties when the information from each input must be kept distinct.

2. **Lack of tiebreaking.** Similar slots are processed similarly, so they will likely contain similar information. Multiple similar slots prefer to capture an average of the relevant inputs rather than each slot capturing a different input, which leads to the problem described earlier where two cats cannot be localized individually.

To avoid these issues, a property called *exclusive multiset-equivariance* is necessary (Zhang et al., 2022). So far, only models from the Deep Set Prediction Networks family (Zhang et al., 2019; 2022) are known to have this property. However, they lack the object-centric inductive bias useful for object-centric learning tasks where the set-equivariant slot attention (Locatello et al., 2020) shines. Fortunately, introducing even a single exclusively multiset-equivariant module in a model is enough to give the entire model this property. In this paper, we develop a module that enhances cross-attention in order to make the object-centric slot attention exclusively multiset-equivariant, thereby addressing the problems of soft assignments and tiebreaking.

[*]Equal contribution [1]Samsung - SAIT AI Lab, Montreal [2]University of Amsterdam [3]Mila, Université de Montreal [4]Canada CIFAR AI Chair [5]TNO. Correspondence to: Yan Zhang <yan@cyan.zone>, David W. Zhang <w.d.zhang@uva.nl>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

In particular, we will look towards the field of optimal transport for inspiration, which is a natural fit for attention models (Sander et al., 2022). This is because optimal transport concerns itself with computing the "best" assignment from one set to another *given some pairwise costs* (Villani, 2009), while cross-attention *learns* exactly such costs. This optimal transport perspective is especially useful because some optimal transport algorithms are able to break ties, which makes them relevant for multiset-equivariance. Unfortunately, these algorithms tend to be slow and difficult to parallelize. On the other hand, solutions to the entropy-regularized optimal transport problem (Cuturi, 2013) are fast but are unable to break ties. We aim to combine the best of both worlds.

Contributions.

1. We establish that slot attention *already* uses an approximation of regularized optimal transport (Section 3). We use this to motivate variants of slot attention where either the approximation, or both the approximation and regularization are removed—with the latter being exclusively multiset-equivariant. However, this comes with a significant speed penalty and a lack of gradients, which can hinder learning.
2. To avoid these issues, we introduce the **MESH** idea: minimize the entropy of Sinkhorn (Section 4). It combines the benefits of our two proposed slot attention variants: speed, gradients, and exclusive multiset-equivariance. We also show why this is more effective for reducing entropy and maintaining useful gradients than Sinkhorn alone.
3. We evaluate our method in slot attention (SA-MESH[1]) on two object detection and two unsupervised object discovery tasks (Section 5). We find that our optimal transport-based variants generally outperform slot attention. Crucially, SA-MESH almost always has the best results—often by a significant margin.

2. Background

Multisets are generalizations of sets by allowing repetitions of elements. In deep learning, sets and multisets are represented as $\mathbb{R}^{n\times c}$ matrices with n being the number of elements and c the feature dimension per element. The uniqueness property of sets is rarely enforced in deep learning, so most models should be thought of as operating on multisets rather than sets (Zhang et al., 2022). These models must then be careful to not rely on the arbitrary order of the n elements. To *guarantee* this, they should satisfy certain equivariances.

[1]Samesh is an Indian name meaning "lord of equality", fitting for our method addressing a problem with equal elements.

2.1. Permutation equivariances

The standard definition of permutation-equivariant (**set-equivariant**) functions f states that a permutation of the input $\boldsymbol{X}$ should result in the same permutation of the output (Zaheer et al., 2017). With Π as the space of $n \times n$ permutation matrices:

$$\begin{aligned} &\forall \boldsymbol{X} \in \mathbb{R}^{n\times c}, \forall \boldsymbol{P} \in \Pi : \\ &\quad f(\boldsymbol{P}\boldsymbol{X}) = \boldsymbol{P} f(\boldsymbol{X}). \end{aligned} \tag{1}$$

However, this means that a set-equivariant function must always produce the same result when there are equal inputs (Zhang et al., 2022): $f([\boldsymbol{a}, \boldsymbol{a}]) = [\boldsymbol{c}, \boldsymbol{d}]$ is not possible for $\boldsymbol{c} \neq \boldsymbol{d}$. Zhang et al. (2022) therefore introduce a more appropriate equivariance for multisets, **multiset-equivariance**:

$$\begin{aligned} &\forall \boldsymbol{X} \in \mathbb{R}^{n\times c}, \forall \boldsymbol{P}_1 \in \Pi, \exists \boldsymbol{P}_2 \in \Pi : \\ &f(\boldsymbol{P}_1\boldsymbol{X}) = \boldsymbol{P}_2 f(\boldsymbol{X}) \ \wedge \ \boldsymbol{P}_1\boldsymbol{X} = \boldsymbol{P}_2\boldsymbol{X}. \end{aligned} \tag{2}$$

It states that when there are interchangeable elements in $\boldsymbol{X}$ (so $\boldsymbol{P}_1\boldsymbol{X} = \boldsymbol{P}_2\boldsymbol{X}$ for $\boldsymbol{P}_1 \neq \boldsymbol{P}_2$), then there are *multiple* permutations of the output that are valid for achieving equivariance. This relaxation of set-equivariance makes the tiebreaking in $f([\boldsymbol{a}, \boldsymbol{a}]) = [\boldsymbol{c}, \boldsymbol{d}]$ possible.

While this property specifies what happens with equal elements, the continuity of most machine learning models suggests that the primary benefit in practice is with *similar* elements (Zhang et al., 2022): similar elements no longer have to result in similar outputs. All set-equivariant models are also multiset-equivariant, which means that only models that are *exclusively multiset-equivariant* (multiset-equivariant, but not set-equivariant) are capable of tiebreaking. Unfortunately, most operations in the multiset learning literature are set-equivariant and thus unable to break ties.

2.2. Slot attention

Slot attention (SA) (Locatello et al., 2020) can be used to abstract the contents of an image into a multiset of "slots". Each slot can be thought of as a "container" that combines related information from the input into a vector. The model learns how to route information from the input into these slots—in object-centric learning, these slots often learn to represent individual objects. It does this by alternating two steps: 1. cross-attention between the multisets of input features and slot features to compute updates for each slot, and 2. utilizing a GRU (Cho et al., 2014) to apply the computed updates to the corresponding slots.

The input features are represented by a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times c}$, and the slots are randomly initialized as a matrix $\boldsymbol{Z}^{(0)} \in \mathbb{R}^{m\times d}$, with m being the number of slots and d the number of dimensions per slot. Cross-attention in slot attention utilizes the standard key-query-value mechanism to compute

updates for each slot. The computed updates are then applied to the corresponding slots using a GRU update. The whole procedure is repeated for a fixed number of times, which is referred to as the number of slot attention iterations.

$$\boldsymbol{Q}^{(l)} = \boldsymbol{Z}^{(l)}\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \tag{3}$$

$$\boldsymbol{A}^{(l)} = \texttt{normalize}(\mathrm{softmax}(\boldsymbol{Q}^{(l)}\boldsymbol{K}^\top)) \tag{4}$$

$$\boldsymbol{Z}^{(l+1)} = \mathrm{GRU}(\boldsymbol{Z}^{(l)}, \boldsymbol{A}^{(l)}\boldsymbol{V}). \tag{5}$$

In Locatello et al. (2020), softmax forces each sum over the m slots to be 1 while `normalize` forces each sum over the n input elements to be 1. All operations used in slot attention are set-equivariant with respect to the slots, which makes the model set-equivariant (Locatello et al., 2020). In this paper, we introduce a module to make slot attention exclusively multiset-equivariant, which allows it to break ties between similar slots.

Note that similar slots can arise due to a multitude of reasons, such as the ones we pointed out in the introduction; it is not necessary for the objects in the input to be similar. This is also why the random initialization of slots in slot attention is not sufficient for tiebreaking, because the slots can become similar *after* the slot updates, after which they can no longer be separated easily.

3. Connecting slot attention to optimal transport

Optimal transport (Villani, 2009) identifies the most cost-effective method for redistributing mass between two distributions. This process typically involves sampling from both distributions, calculating the pairwise distances between the samples, and applying an optimal transport algorithm to determine the transport map that minimizes the total cost. In the context of cross-attention, we are comparing two multisets containing the input and slot features rather than two distributions, but the same principles of computing pairwise distances and finding an optimal transport map still apply. By making this connection between slot attention and optimal transport more explicit, we can leverage the powerful algorithms of optimal transport to enhance the performance of slot attention.

Entropy-regularized optimal transport. A critical component when applying cross-attention in slot attention is the normalization of the attention matrix. Equation 4 first exponentiates all the entries, then normalizes one dimension of the attention matrix to sum to 1, then the other dimension to sum to 1. This sequence of operations is also known as applying the Sinkhorn algorithm for a single step. The Sinkhorn algorithm solves the entropy-regularized optimal transport problem (Cuturi, 2013) by repeatedly alternating these two normalizations, which results in a doubly stochastic matrix at convergence. We can therefore think of Equation 4 as approximating this entropy-regularized optimal transport (by using only one Sinkhorn iteration) to determine how the information from the input should be associated with the slots. This connection between transformers (which use a slightly different normalization) and optimal transport has also been made by Sander et al. (2022).

A simple extension is thus to consider slot attention using more than one Sinkhorn iteration, which we will refer to as **SA-SH**. In this variant, instead of using a similarity score, a distance function d (e.g. L2 norm[2]) is used to compute the attention matrix as follows:

$$C_{ij} = d(\boldsymbol{Q}_i, \boldsymbol{K}_j) \tag{6}$$

$$\boldsymbol{A} = \mathrm{sinkhorn}(\boldsymbol{C}). \tag{7}$$

Although the proposed extension properly computes the optimal transport map, it remains limited by its set-equivariance and is unable to perform tiebreaking. One additional detail is that the Sinkhorn algorithm must handle non-square matrices since the number of slots is usually much smaller than the number of inputs; naïvely applying the algorithm on such matrices does not converge. We describe the details of how to handle this in Appendix A as they are not important to the following discussion.

Unregularized optimal transport. By using the full Sinkhorn algorithm, SA-SH replaces the usual attention matrix with the transport map of the regularized optimal transport problem. This raises the question whether other optimal transport algorithms can be used in the context of slot attention too. A benefit of optimal transport without regularization is that it can be *exclusively multiset-equivariant*: many algorithms naturally include tiebreaking, which leads to low entropy solutions. By replacing entropy-regularized optimal transport with unregularized optimal transport, we can make slot attention exclusively multiset-equivariant to avoid the issues we pointed out in Section 1.

We thus propose the **SA-EMD** (Earth Mover's Distance) variant, wherein we use the EMD algorithm (Bonneel et al., 2011) that is part of the POT package (Flamary et al., 2021). The EMD algorithm provides a sparse solution to the unregularized optimal transport problem and has the ability to break ties. However, using the EMD algorithm also presents some challenges. The gradients of unregularized optimal transport problems are piecewise constant, which prevents learning of the cost matrix $\boldsymbol{C}$. Thus, we need to estimate gradients, for which many different techniques exist (Gould et al., 2016; Fung et al., 2022; Bai et al., 2019; Pogančić et al., 2020). In practice, we observe that the gradient of the Sinkhorn operator is also a good descent direction for EMD.

[2] Sander et al. (2022) point out that for Sinkhorn, this is equivalent to the usual (negative) dot product used in cross-attention.

We find that we obtain the best empirical results through:

$$\boldsymbol{A} = \mathrm{emd}(\boldsymbol{C}) + \mathrm{sinkhorn}(\boldsymbol{C}). \tag{8}$$

Another issue is that the EMD algorithm is relatively slow. The standard solvers use a network simplex algorithm (a variant of the simplex algorithm for graphs), which is difficult to parallelize efficiently on GPUs. Can we get the benefits of unregularized optimal transport with its exclusive multiset-equivariance, while still being fast and differentiable like entropy-regularized optimal transport?

4. MESH: minimizing the entropy of Sinkhorn

Previously, we focused on different ways of turning costs $\boldsymbol{C}$ into a transport map $\boldsymbol{A}$. Now, we turn our attention to *changing the costs themselves*, followed by using the computational efficiency of the Sinkhorn algorithm to compute the transport map for these modified costs. A key difference between unregularized optimal transport and entropy-regularized optimal transport is the entropy in the resulting transport map. *The idea is to change the costs in such a way that even after entropy-regularized optimal transport, the entropy remains low.* This objective allows the tiebreaking necessary for exclusive multiset-equivariance. To implement this idea, we aim to find a new cost $\boldsymbol{C}'$ that minimizes the entropy $H(P) = -\sum_{i,j} P_{ij} \log P_{ij}$.

$$\mathrm{MESH}(\boldsymbol{C}) = \underset{\boldsymbol{C}' \in \mathcal{V}(C)}{\arg\min}\, H(\mathrm{sinkhorn}(\boldsymbol{C}')) \tag{9}$$

$$\boldsymbol{A} = \mathrm{sinkhorn}(\mathrm{MESH}(\boldsymbol{C})). \tag{10}$$

$\mathcal{V}(\boldsymbol{C})$ denotes a neighborhood around $\boldsymbol{C}$, which is implicitly defined by how we implement the $\arg\min$. We refer to this variant of slot attention as **SA-MESH** (Minimize Entropy of Sinkhorn). Equation 9 changes the cost so that the resulting transport map has low entropy (preferring 0s and 1s) despite the entropy regularization. The final transport map is calculated from this new cost matrix efficiently using the Sinkhorn algorithm. In summary, the costs are changed to make the transport map look more like the output of unregularized optimal transport.

To solve this optimization problem, we propose to use gradient descent for a small, fixed number of steps starting from the original cost matrix $\boldsymbol{C}$. The last point ensures that the new cost matrix remains close to $\boldsymbol{C}$ so that the new optimal transport problem will be similar to the original one. Differentiating through this can be done with standard automatic differentiation. While the process described so far does not feature any tiebreaking explicitly (and would in fact struggle to minimize entropy successfully when exactly equal rows or columns are present in $\boldsymbol{C}$), we can simply add a small amount of noise at the start of the optimization: $\boldsymbol{C}'^{(0)} = \boldsymbol{C} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(0, 10^{-6})$. Another important detail is to normalize the gradient to have a fixed norm: the small amount of noise is amplified (to break ties) only when slots are similar. This is done without having to resort to large learning rates, which would impact the stability of optimization. We thus propose to repeat the following for T steps with a learning rate λ:

$$\boldsymbol{C}'^{(t+1)} = \boldsymbol{C}'^{(t)} - \lambda \frac{\nabla_{\boldsymbol{C}'^{(t)}} \boldsymbol{A}'^{(t)}}{\left\| \nabla_{\boldsymbol{C}'^{(t)}} \boldsymbol{A}'^{(t)} \right\|} \tag{11}$$

$$\text{with } \boldsymbol{A}'^{(t)} = H(\mathrm{sinkhorn}(\boldsymbol{C}'^{(t)})). \tag{12}$$

Note that exact minimization is not needed, nor necessarily desirable. A critical aspect is that $\boldsymbol{C}'^{(T)}$ is still related to the initial $\boldsymbol{C}$. For example, if $\boldsymbol{C}$ is square, then $\boldsymbol{C}' = 10^{23}\boldsymbol{I}$ would always "successfully" minimize Equation 9, but also solve a problem unrelated to $\boldsymbol{C}$ and provide no signal to learn from. This is why we initialize $\boldsymbol{C}'^{(0)}$ as $\boldsymbol{C}$ with a small amount of noise. Note that if this noise is too high (e.g. $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(0, 1)$), the solution can again lose correspondence with the actual cost matrix $\boldsymbol{C}$ that we are trying to compute the transport map for. We develop a more complicated version that explicitly enforces $\boldsymbol{C}$ and $\boldsymbol{C}'$ to be similar (Appendix B) but find that it is not necessary in practice as long as the noise in the initialization is reasonably low.

4.1. Properties

Equivariance. SA-MESH is now exclusively multiset-equivariant: it is multiset-equivariant because all the individual operations are multiset-equivariant, but it is not set-equivariant because equal slots will no longer receive the same transport plans due to the tiebreaking from the noise and subsequent optimization (see proof in Appendix C). This gives our method more representational power than standard SA and SA-SH because it is no longer restricted by set-equivariance.

A useful side-effect is that random initialization of slots is no longer necessary. Since ties can be broken by this entropy minimization, it is no problem to initialize all slots to be the same vector. In standard slot attention, the amount of noise in $\boldsymbol{Z}^{(0)}$ needs to be just right: too low, and the set-equivariant model has difficulties breaking ties between these similar slots (Wu et al., 2022); too high, and the model can become unreliable from the noisiness (Kipf et al., 2022). Furthermore, randomly initializing the slots does not prevent them from collapsing to the same values in a later iteration—such as when there are fewer objects to model than slots. In contrast, a tiny amount of noise in $\boldsymbol{C}'$ (as long as it is above machine precision after $\mathrm{sinkhorn}$) is sufficient for tiebreaking and avoiding collapse, and any other effect of the noise can be optimized away through the gradient descent.

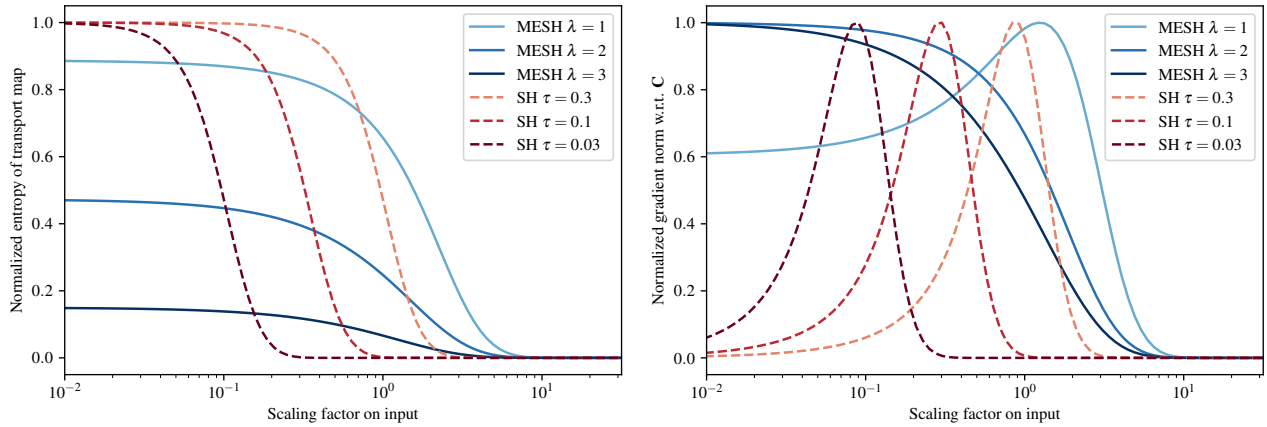


Figure 1. MESH reduces entropy while maintaining reasonable gradients across a large range of attention values, in particular when attention is uncertain (low scaling factors). Meanwhile, Sinkhorn (SH) provides nontrivial gradients in a much smaller range with entropy reduction being ineffective when the scaling is too low. Entropy of transport map (left) and corresponding gradient norm (right) when varying the scaling factor on the input. The entropy is normalized to have 1 as maximum entropy. The gradient norm of each method is normalized to have a maximum of 1, we show the unnormalized gradient norms in Appendix E.

Speed. SA-MESH offers a significant improvement in computation time when compared to SA-EMD. This is because it only requires the evaluation of the Sinkhorn algorithm for a small number of optimization steps (in our experiments, we found little improvement above four steps). While it may not be as fast as SA-SH, its exclusive multiset-equivariance can speed up learning by making the model more powerful. In many cases, the computation required for SA-MESH is outweighed by other components of the model such as the image processing part that creates the input multiset for SA. Since each optimization step in MESH needs to solve a similar optimal transport problem, we can reuse the result from the previous iterations to improve the efficiency, as we describe in Appendix D.

Gradients. Another benefit over SA-EMD is that gradient computation is simple since we can fully rely on automatic differentiation instead of having to manually estimate gradients for the black-box EMD solver in SA-EMD. We find experimentally that we do not even need to differentiate the gradient updates in Equation 11 themselves; the gradients of $\boldsymbol{C}'^{(T)}$ can simply be passed along to $\boldsymbol{C}$ in a straight-through manner (Bengio et al., 2013) without reduction in performance. SA-MESH also has benefits in terms of the "quality" of gradients over SA-SH, which we explain in the following.

4.2. Comparison to changing temperature

An alternative for breaking ties is to add noise to the cost matrix (like SA-MESH), but then simply reduce the temperature of the Sinkhorn algorithm, which corresponds to reducing the amount of entropy regularization (Cuturi, 2013). For sufficiently low temperatures, this should also be able to map equal inputs to different slots. How does this much simpler approach compare to MESH, which minimizes entropy by gradient descent?

Sinkhorn with low temperatures can be thought of as analogous to softmax with low temperatures. As the temperature decreases, the behavior of softmax becomes more similar to an argmax, but the gradients become ill-behaved as a result. Similarly, with Sinkhorn, low temperatures may result in gradients that make it difficult or impossible to learn a good cost matrix. We now make this notion more concrete and show that MESH can reduce entropy *while maintaining "good" gradients* for a much larger range of inputs.

In Figure 1, we have the following set-up. Starting from a 10×10 identity matrix, we scale it by a varying factor to obtain the cost matrix, then apply either Sinkhorn (for different temperatures τ) or MESH (for different learning rates λ). We then measure the entropy of the resulting transport map (left), as well as the norm of the gradient of this entropy with respect to the cost matrix (right). Gradient norms close to zero slow down learning because the gradients provide little to no information on how to learn the cost matrix.

When the scaling factor is large (e.g. >10, attention is confident), the behaviors are similar (low entropy and small gradients). When the scaling factor is small (e.g. <0.1, attention is not confident), MESH is still always able to reduce entropy while maintaining reasonable gradients. In contrast, Sinkhorn only has nontrivial gradients in a relatively small range; outside of this range, learning is difficult because the gradient norms become close to 0. Thus, a trade-off has to be made for Sinkhorn, which is not necessary for MESH: either τ is high and low scaling factors result in virtually zero gradients (with no reduction to the entropy), or τ is low and higher scaling factors result in virtually zero gradients

Table 1. Random objects detection, measured in RMSE divided by standard deviation σ of random objects (lower is better). An always-predict-zeros baseline has a normalized RMSE of 1. Median over 5 random seeds.

Model	$\sigma = 1.0$	$\sigma = 0.1$	$\sigma = 0.01$
SA	0.44	0.53	0.65
SA-SH	0.27	0.32	0.41
SA-EMD	**0.23**	0.52	1.04
SA-MESH	0.24	**0.27**	**0.31**

instead. The MESH learning rate λ can be used to control the shape of how entropy is reduced, while the Sinkhorn temperature τ only changes the location and does not increase the range of nontrivial gradients. In summary, MESH effectively reduces entropy while maintaining well-behaved gradients for a large range of inputs.

5. Experiments

We now experimentally evaluate SA with our optimal transport variants, with a particular focus on comparing SA to SA-MESH. We open-source all of our code https://github.com/davzha/MESH and provide extra experimental details in Appendix F.

5.1. Random objects detection

First, we evaluate the effect of the different SA variants in a simplified object detection setting. The aim is to assess a model's ability to detect and distinguish similar objects in a controlled setting. Given a multiset containing k random 32d vectors sampled from $\mathcal{N}(0, \sigma^2 \boldsymbol{I})$ and h zero vectors, the goal is to copy only the k random vectors into the k slots. In the context of object detection on images, this can be thought of as detecting k different objects, each of which occupies exactly one position in a feature map.

In this setting, we want the object information to be preserved as accurately as possible. To vary the difficulty of this task, we change the standard deviation σ of the random vectors to be copied and measure the error relative to this σ. A lower standard deviation corresponds to a harder task because the elements become more similar to each other and to the background zero vectors.

Results. Table 1 shows our results for $k = 5$ objects and $h = 100$ background elements for varying difficulties σ. First, we see that SA-SH and SA-MESH can detect the objects on the hardest setting of $\sigma = 0.01$ more accurately than SA on $\sigma = 1$. This demonstrates the general value of our proposed optimal transport perspective in the context of attention. SA-MESH with its lower entropy outperforms SA-SH: the SA-MESH performance for a specific σ is roughly equivalent to SA-SH at a σ ten times higher. This shows the benefits of reducing entropies in the transport map through MESH, which helps objects stay distinct from each other. Meanwhile, SA-EMD (also with low entropy transport maps) performs similarly to SA-MESH on $\sigma = 1$, but degenerates to the always-predict-zeros baseline on $\sigma = 0.01$. We attribute this to the inherently imprecise gradient estimation leading to learning problems.

5.2. CLEVR property prediction

Next, we test SA and our proposed variants on a more realistic object detection task. CLEVR (Johnson et al., 2017) is a synthetic dataset containing images with up to ten objects in a 3d scene. Each object is sampled with varying sizes, materials, shapes, and colors. The task is to predict the multiset of objects with their properties and 3d position. Following Zhang et al. (2019), we evaluate using average precision (AP) at different distance thresholds for the 3d coordinates of the predicted objects.

Results. Table 2 shows that SA-MESH achieves the best SA results (and state-of-the-art results on some metrics) while only increasing run time by a small amount. These results are followed by SA-EMD, which has slightly worse results (likely due to the inherently imprecise gradient estimation) but takes over three times longer to train. Again, we see that all three optimal transport-based methods greatly

Table 2. CLEVR property prediction, average precision (AP) in % (mean ± standard deviation) over 5 random seeds, higher is better. SA-MESH improves over all other SA variants at only a small computational cost. Note that the exclusively multiset-equivariant iDSPN is not object-centric, so it is not fully comparable. SA (original) results are copied from Locatello et al. (2020), iDSPN results from Zhang et al. (2022). Models with † use the improvement by Chang et al. (2022), see Appendix G for our results without †.

Model	AP_∞	AP_1	$AP_{0.5}$	$AP_{0.25}$	$AP_{0.125}$	$AP_{0.0625}$	Train time
iDSPN (Zhang et al., 2022)	$98.8_{\pm 0.5}$	$98.5_{\pm 0.6}$	$98.2_{\pm 0.6}$	$95.8_{\pm 0.7}$	$76.9_{\pm 2.5}$	$32.3_{\pm 3.9}$	—
SA (original) (Locatello et al., 2020)	$94.3_{\pm 1.1}$	$86.7_{\pm 1.4}$	$56.0_{\pm 3.6}$	$10.8_{\pm 1.7}$	$0.9_{\pm 0.2}$	—	—
SA†	$94.3_{\pm 0.4}$	$85.7_{\pm 1.6}$	$77.2_{\pm 1.5}$	$53.1_{\pm 2.7}$	$16.7_{\pm 1.8}$	$4.0_{\pm 0.7}$	2.2 h
SA-SH†	$98.9_{\pm 0.2}$	$97.7_{\pm 0.5}$	$95.2_{\pm 0.9}$	$83.3_{\pm 0.8}$	$38.5_{\pm 2.0}$	$10.0_{\pm 1.4}$	2.3 h
SA-EMD†	$99.3_{\pm 0.3}$	$98.1_{\pm 0.4}$	$95.9_{\pm 0.8}$	$85.8_{\pm 1.1}$	$42.0_{\pm 2.0}$	$11.4_{\pm 1.3}$	9.3 h
SA-MESH†	$\mathbf{99.4}_{\pm 0.1}$	$\mathbf{99.2}_{\pm 0.2}$	$\mathbf{98.9}_{\pm 0.2}$	$\mathbf{91.1}_{\pm 1.1}$	$\mathbf{47.6}_{\pm 0.8}$	$\mathbf{12.5}_{\pm 0.4}$	2.4 h

outperform the baseline SA, which validates the benefits of the optimal transport perspective in attention.

We believe that there are two reasons why SA-SH performs much better than SA, even though both are set-equivariant. Keep in mind that SA is equivalent to SA-SH with a single Sinkhorn iteration. Performing more Sinkhorn iterations gives a more accurate transport map, which can be interpreted as fully resolving the "competition" between slots for inputs. This competition interpretation is what motivated the normalizations (1-step Sinkhorn) in (Locatello et al., 2020). The other factor is that SA-SH, in order to converge for rectangular cost matrices, requires the use of learned marginals (see Appendix A). These can assist with weighting down the importance of background pixels or unused slots, which makes it easier for the model to learn the $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ matrices of slot attention.

Note that we only provide the results for iDSPN (which is also exclusively multiset-equivariant) for context; this is not supposed to be a direct comparison due to the significant difference in approach. In general, slot attention through the use of attention has the benefit of not needing to compress the input into a single vector (global scene representation) like iDSPN. The resulting object-centric inductive bias and the relative simplicity have allowed for wider adoption and success of SA over iDSPN (Kipf et al., 2022; Hu et al., 2020; Li et al., 2021; Sajjadi et al., 2022), which makes our improvements to SA meaningful, even if in this case some metrics are worse than iDSPN.

5.3. Unsupervised object discovery on images

In this task, the objective is to discover objects without the supervision of what the objects are. We follow Locatello et al. (2020) and set up an image reconstruction task with slots as the latent bottleneck using SA. These slots are individually decoded into object-specific images, each comprising the RGB color channels and an alpha mask. These object-specific images are then combined to form the final reconstructed image. To evaluate the performance, we compare the per-slot alpha masks to the actual object segmentation masks. The goal is thus for image reconstruction with a multiset bottleneck to lead to a decomposition of the scene into individual objects, with each object being modeled by a distinct slot.

We evaluate on the Multi-dSprites dataset, which is the only benchmark presented by Locatello et al. (2020) that still presented a challenge (possibly due to the presence of highly overlapping objects). Additionally, we test on ClevrTex (Karazija et al., 2021), a synthetic dataset similar to CLEVR that introduces the added challenge of different textures. In line with prior work (Kipf et al., 2022), we evaluate the Foreground Adjusted Rand Index (FG-ARI) and Foreground mean Intersection over Union (FG-mIoU).

Table 3. Object discovery on images results in Multi-dSprites in % (mean ± standard deviation) over 5 random seeds, higher is better. SA-MESH outperforms the other models and has lower variance.

Model	FG-ARI	FG-mIoU
SA (Locatello et al., 2020)	$91.3_{\pm 0.3}$	—
SA	$92.2_{\pm 0.5}$	$24.3_{\pm 5.4}$
SA-SH	$87.2_{\pm 1.8}$	$84.0_{\pm 3.1}$
SA-MESH	$\mathbf{95.6}_{\pm \mathbf{0.2}}$	$\mathbf{86.2}_{\pm \mathbf{2.2}}$

Table 4. Object discovery on images results in ClevrTex in % (mean ± standard deviation) over 5 random seeds, higher is better. SA-MESH outperforms the other models and has lower variance.

Model	FG-ARI	FG-mIoU
SA	$52.8_{\pm 14.9}$	$26.3_{\pm 14.9}$
SA-SH	$70.8_{\pm 5.8}$	$35.3_{\pm 4.4}$
SA-MESH	$\mathbf{79.0}_{\pm \mathbf{2.6}}$	$\mathbf{43.2}_{\pm \mathbf{4.0}}$

To ensure FG-mIoU is permutation-insensitive, we use the Hungarian algorithm to find the best matching between the masks. See Section F.3 for more details. Note that we no longer test SA-EMD because the larger input size compared to Section 5.2 makes its training time infeasible.

Results. Table 3 and Table 4 show that on both Multi-dSprites and ClevrTex, SA-MESH achieves significantly higher FG-ARI and FG-mIoU compared to all baselines. SA-SH and SA-MESH improve especially in mIoU, which Karazija et al. (2021) argue is a better metric than ARI to evaluate the accuracy of object masks. Appendix H shows two extra ablations on the training setup.

Analysis. In Figure 2, we observe that minimizing the entropy leads to much sparser attention maps for SA-MESH in comparison to SA: SA-MESH only attends to a small part within each object, rather than the whole object as is usually the case for SA. Keep in mind that attention maps are only used to route the required information to each slot, and it is the slots themselves that represent the individual

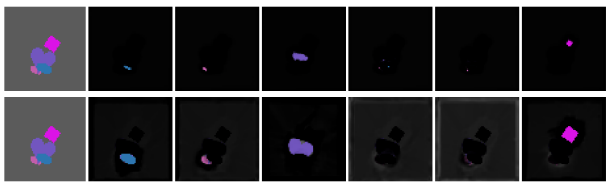


Figure 2. Attention maps (first row) and alpha masks (second row) for each of the six slots in SA-MESH on Multi-dSprites. The attention focuses on only a small region inside each object, but the model still reconstructs the full objects. See Appendix J for more examples.

Table 5. Video object discovery results on CLEVRER-S and CLEVRER-L in % (mean ± standard deviation) over 5 random seeds. SA-MESH outperforms all other models in terms of quality of predicted masks (ARI, mIoU) and achieves high temporal consistency (TC). See Appendix J for example masks.

	CLEVRER-S			CLEVRER-L		
Model	FG-ARI	FG-mIoU	TC	FG-ARI	FG-mIoU	TC
SA	$78.1_{\pm 14.0}$	$16.8_{\pm 9.8}$	$26.3_{\pm 22.5}$	$69.6_{\pm 14.9}$	$12.2_{\pm 6.6}$	$12.8_{\pm 8.7}$
SA fixed noise	$71.0_{\pm 34.0}$	$17.1_{\pm 11.4}$	$42.8_{\pm 19.3}$	$79.4_{\pm 5.9}$	$11.9_{\pm 6.2}$	$18.5_{\pm 13.0}$
SA learned noise	$80.2_{\pm 14.1}$	$13.2_{\pm 4.9}$	$21.3_{\pm 13.5}$	$84.7_{\pm 5.7}$	$10.2_{\pm 1.2}$	$19.4_{\pm 7.7}$
SA-SH	$89.3_{\pm 2.3}$	$10.0_{\pm 2.8}$	$\mathbf{89.7_{\pm 2.4}}$	$82.9_{\pm 1.5}$	$7.0_{\pm 0.2}$	$26.8_{\pm 0.5}$
SA-MESH	$\mathbf{93.8_{\pm 1.0}}$	$\mathbf{44.1_{\pm 7.1}}$	$80.2_{\pm 10.1}$	$\mathbf{92.9_{\pm 2.2}}$	$\mathbf{54.4_{\pm 8.9}}$	$\mathbf{55.4_{\pm 5.7}}$

objects. Attending to a small part of each object is sufficient because the receptive field of the CNN image encoder lets it move information from the edge of an object into the center. The accuracy of the final alpha masks of SA-MESH shows that this happens successfully. In Appendix J, we see that SA-MESH attends less to the background than SA on Multi-dSprites.

Similar to Locatello et al. (2020), we observe that some SA runs fail to separate the individual objects into different slots despite having a low reconstruction error. In those cases, the attention maps divide the image into separate regions, independent of the image content. SA-MESH ensures that the attention maps are sparse, which helps to avoid these kinds of failure modes. Appendix J shows this difference between SA-MESH and SA on ClevrTex: SA-MESH discovers objects successfully in most cases, and usually, only the background is split into spatial regions.

5.4. Unsupervised object discovery on video

As we mentioned in Section 1, Wu et al. (2022) observed issues with SA when applied to videos where multiple objects can enter the scene. To evaluate our method in this scenario, we build two variants of the CLEVRER video dataset (Yi et al., 2019) where the number of visible objects varies over time. We only use two frames from each video: the first frame, and either the 16th frame (short time difference, CLEVRER-S) or the 128th frame (long time difference, CLEVRER-L). We do this to evaluate SA without the dynamics prediction component that Wu et al. (2022) are concerned with. In CLEVRER-S, the total number of objects increases by two or more in 9.5% of the videos, while in CLEVRER-L, this occurs for 68.8% of the videos.

We evaluate FG-ARI and FG-mIoU on the two frames individually, which is then averaged. We also compute a temporal consistency (TC) metric as the fraction of objects that are correctly captured by the same slot (details in Section F.4).

Results. Table 5 shows that SA-MESH outperforms the other models by a significant margin. The only exception is on CLEVRER-S for temporal consistency—we can close this gap simply by reducing the MESH learning rate λ (five run average: 90.0% FG-ARI, 22.7% FG-mIoU, 95.1% TC). The proposal by Wu et al. (2022) of adding noise to slots to prevent them from collapsing helps SA on CLEVRER-L, but to a lesser extent than SA-MESH, which uses exclusive multiset-equivariance to prevent collapse. This is evidenced by the much better FG-mIoU of SA-MESH.

6. Related work

Object-centric learning. Object-centric learning aims to model visual input data in terms of multiple "objects" rather than a global representation or grid of feature vectors. This decomposition can be considered an *abstraction* of the input; reasoning over a small number of objects is intuitively more efficient than over a feature map (Ke et al., 2022; Huang et al., 2020). Scenarios with multiple independent objects are ubiquitous in natural data, so it is desirable to model them well. Karazija et al. (2021) classify object-centric learning methods into three categories: pixel-space approaches which group related pixels together (Greff et al., 2019; Pervez et al., 2022), glimpse approaches which sequentially extract patches from the input (Crawford & Pineau, 2019; Lin et al., 2020; Jiang & Ahn, 2020), and sprite approaches which learn a dictionary of object appearances (Monnier et al., 2021; Smirnov et al., 2021). These are part of the wider research area of factorizing knowledge into smaller, independent parts which can be modeled more easily (Goyal et al., 2019; 2021; Didolkar et al., 2021).

We choose to apply MESH on specifically the pixel-space based slot attention (Locatello et al., 2020) because of its simplicity in approach (cross-attention with GRU updates), the lack of assumptions on what an object is (which makes it a general technique), and its set-equivariance. Since Zhang et al. (2022) show a specific limitation with set-equivariance, there is a clear path towards improvement, namely making it exclusively multiset-equivariant. We accomplish this in this paper, with strong results backing up the benefits of our approach.

Optimal transport. Another approximation of optimal transport can be obtained through the Sliced Wasserstein Distance (Bonneel et al., 2015). It performs tiebreaking through the use of numerical sorting and is thus exclusively multiset-equivariant, but it lacks precise 1-to-1 associations between inputs and slots. This can especially be a problem with varying input sizes. We tried approaches based on this method as a replacement for standard cross-attention but did not obtain any competitive results.

In a similar direction to Sinkhorn, which performs entropy-regularized optimal transport, Blondel et al. (2018) study L2-regularized optimal transport problems. While their solver is faster than unregularized optimal transport and obtains lower entropy solutions than Sinkhorn, similarly to Sinkhorn the convexity of the problem means that it cannot break ties effectively on its own, even with noise. In SA-MESH, we can replace Sinkhorn with this method, but we found that it was too slow comparatively.

7. Discussion

We introduced several variants of slot attention that can break ties between slots which enables better modeling of objects. In particular, MESH is a promising method that enhances cross-attention. As a result, it grants slot attention the property of exclusive multiset-equivariance while maintaining learnability and efficiency.

While minimizing entropy in MESH has a nice symmetry with entropy-regularized optimal transport, it is not clear whether a sparse attention map is always desirable. We try to learn a neural network on the transport map in Appendix I, but find no improvements over simply using the entropy; the derivative of the learned objective ends up with a similar shape to that of the entropy, which suggests that entropy is indeed a reasonable choice to minimize for now. While the experiments on ClevrTex are a small step towards more complicated image data, we do not have any evaluation on real world data, so it is not certain what new problems will present themselves. Recent object-centric learning techniques that are able to scale to real-world scenarios often use more powerful image encoders and decoder architectures with the vanilla slot attention, so it is possible in principle to simply replace SA with SA-MESH in these models. In practice, it is so far uncertain whether any inductive biases introduced by SA-MESH only apply well on simpler synthetic data.

Our experiments show that in certain cases, SA-SH can already provide most of the benefits without the additional complexity of the bi-level optimization in SA-MESH. For example, most of the benefit over SA in Section 5.1 is already obtained with SA-SH, while SA-MESH only provides a small benefit over SA-SH. On the other hand, there are cases like Section 5.4 where SA-MESH greatly outperforms SA-SH. We believe that it is important to gain a better understanding of what situations make one preferable over the other.

A limitation of our experiments is that we only evaluate the MESH idea in the context of slot attention, when in reality it is a more general method. For example, it could be used to enhance self-attention in Transformers. Another example is that Sinkhorn is used by Peña et al. (2022) for merging the weights of two neural networks together. MESH could be used as an alternative in this context to replace the Sinkhorn algorithm. In general, we believe that optimal transport will continue to play an important role in deep learning, with MESH being a way of bringing tiebreaking into the picture without paying the usual speed penalty associated with optimal transport.

Acknowledgements

The work of DWZ is part of the research programme Perspectief EDL with project number P16-25 project 3, which is financed by the Dutch Research Council (NWO) domain Applied and Engineering Sciences (TTW). This research was enabled in part by compute resources provided by Mila (`mila.quebec`), Calcul Québec (`calculquebec.ca`), the Digital Research Alliance of Canada (`alliancecan.ca`), and by support from the Canada CIFAR AI Chair Program. Simon Lacoste-Julien is a CIFAR Associate Fellow in the Learning Machines & Brains program.

References

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Blondel, M., Seguy, V., and Rolet, A. Smooth and sparse optimal transport. In *International Conference on Artificial Intelligence and Statistic (AISTATS)*, 2018.

Bonneel, N., Van De Panne, M., Paris, S., and Heidrich, W. Displacement interpolation using lagrangian mass transport. In *SIGGRAPH Asia*, 2011.

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

Chang, M., Griffiths, T. L., and Levine, S. Object representations as fixed points: Training iterative refinement

algorithms with implicit differentiation. *arXiv preprint arXiv:2207.00787*, 2022.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Crawford, E. and Pineau, J. Spatially invariant unsupervised object detection with convolutional neural networks. In *AAAI Conference on Artificial Intelligence*, 2019.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Didolkar, A., Goyal, A., Ke, N. R., Blundell, C., Beaudoin, P., Heess, N., Mozer, M., and Bengio, Y. Neural production systems: Learning rule-governed visual dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78), 2021.

Fung, S. W., Heaton, H., Li, Q., McKenzie, D., Osher, S., and Yin, W. JFB: Jacobian-free backpropagation for implicit networks. In *AAAI Conference on Artificial Intelligence*, 2022.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent independent mechanisms. In *International Conference on Learning Representations (ICLR)*, 2019.

Goyal, A., Lamb, A., Gampa, P., Beaudoin, P., Levine, S., Blundell, C., Bengio, Y., and Mozer, M. Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. In *International Conference on Learning Representations (ICLR)*, 2021.

Greff, K., Kaufmann, 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 (ICML)*, 2019.

Hu, J., Yang, Y., Chen, C., He, L., and Yu, Z. SAS: Dialogue state tracking via slot attention and slot information sharing. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

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

Jiang, J. and Ahn, S. Generative neurosymbolic machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Karazija, L., Laina, I., and Rupprecht, C. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. *arXiv preprint arXiv:2111.10265*, 2021.

Ke, N. R., Didolkar, A., Mittal, S., Goyal, A., Lajoie, G., Bauer, S., Rezende, D., Bengio, Y., Mozer, M., and Pal, C. Systematic evaluation of causal discovery in visual model based reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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. In *International Conference on Learning Representations (ICLR)*, 2022.

Li, L., Wang, B., Verma, M., Nakashima, Y., Kawasaki, R., and Nagahara, H. Scouter: Slot attention-based classifier for explainable image recognition. In *International Conference on Computer Vision (ICCV)*, 2021.

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 *International Conference on Learning Representations (ICLR)*, 2020.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Monnier, T., Vincent, E., Ponce, J., and Aubry, M. Unsupervised layered image decomposition into object prototypes. In *International Conference on Computer Vision (ICCV)*, 2021.

Pervez, A., Lippe, P., and Gavves, E. Differentiable mathematical programming for object-centric representation learning. *arXiv preprint arXiv:2210.02159*, 2022.

Peyré, G. and Cuturi, M. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6): 355–607, 2019.

Peña, F. A. G., Medeiros, H. R., Dubail, T., Aminbeidokhti, M., Granger, E., and Pedersoli, M. Re-basin via implicit sinkhorn differentiation. *arXiv preprint arXiv:2212.12042*, 2022.

Pogančić, M. V., Paulus, A., Musil, V., Martius, G., and Rolinek, M. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*, 2020.

Sajjadi, M. S. M., Duckworth, D., Mahendran, A., van Steenkiste, S., Pavetić, F., Lučić, M., Guibas, L. J., Greff, K., and Kipf, T. Object scene representation transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Smirnov, D., Gharbi, M., Fisher, M., Guizilini, V., Efros, A., and Solomon, J. M. Marionette: Self-supervised sprite learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Villani, C. *Optimal transport: old and new*, volume 338. Springer, 2009.

Wei, X., Zhang, T., Li, Y., Zhang, Y., and Wu, F. Multi-modality cross attention network for image and sentence matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Wu, Z., Dvornik, N., Greff, K., Kipf, T., and Garg, A. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations (ICLR)*, 2019.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Zhang, Y., Hare, J., and Prügel-Bennett, A. Deep set prediction networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

A. Marginals in Sinkhorn and EMD

As we mention in the main text, we need to account for the (typical) case of the number of inputs n and the number of slots m differing, i.e. with a cost matrix $\boldsymbol{C} \in \mathbb{R}^{m \times n}$. The problem is that it is impossible to make every row and every column of the transport map sum to 1 when the number of rows and columns is different. Fortunately, there is standard practice for how to deal with this case in optimal transport (Peyré & Cuturi, 2019; Cuturi, 2013; Bonneel et al., 2011). We can define non-negative marginals $\boldsymbol{a} \in \mathbb{R}^m$ and $\boldsymbol{b} \in \mathbb{R}^n$ that specify the row and column sums of the transport map respectively. If $\sum_i \boldsymbol{a}_i = \sum_j \boldsymbol{b}_j$, then convergence is as normal.

In our case, we learn both $\boldsymbol{a} = m \cdot \mathrm{softmax}(h_{\boldsymbol{a}}(\boldsymbol{Z}))$ and $\boldsymbol{b} = m \cdot \mathrm{softmax}(h_{\boldsymbol{b}}(\boldsymbol{X}))$ with neural networks $h_{\boldsymbol{a}} : \mathbb{R}^d \to \mathbb{R}$ and $h_{\boldsymbol{b}} : \mathbb{R}^c \to \mathbb{R}$ that are shared across the m slots or n input elements respectively. These allow the model to put focus on important input elements (e.g. the inputs corresponding to objects) and ignore unimportant input elements (e.g. the inputs corresponding to the background), as well as put focus on the relevant number of slots. Since both softmaxes sum to one, we have $\sum_i \boldsymbol{a}_i = \sum_j \boldsymbol{b}_j = m$ so there is no problem with convergence.

For the Sinkhorn algorithm, it now repeatedly alternates normalizing all the rows to sum to $\boldsymbol{a}$, then all the columns to sum to $\boldsymbol{b}$. For the EMD solver that we use (Bonneel et al., 2011), these marginals are standard parameters in the algorithm. In the main text, we omit these marginals whenever we refer to $\mathrm{sinkhorn}$ or emd for simplicity of notation.

B. Enforcing C' to be similar to C

The following discussion is not critical to understanding the main text, since we find empirically that with the right initialization (e.g. $\boldsymbol{C}' = \boldsymbol{C} + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(0, 10^{-6})$), learning is not a problem. We only find that this variant is

necessary with an initialization such as $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$. As we mention in the main text, the amount of noise is not important as long as it remains above machine precision after applying sinkhorn, which can be easily checked a-priori.

In order to enforce $\boldsymbol{C}'$ to be related to $\boldsymbol{C}$ more explicitly, we define the following objective instead:

$$\begin{aligned}\text{MESH}(\boldsymbol{C}) = \arg\min_{\boldsymbol{C}'} [&H(\text{sinkhorn}(\boldsymbol{C}')) \\ &+\alpha||\text{sinkhorn}(\boldsymbol{C}')\boldsymbol{S} - \text{sinkhorn}(\boldsymbol{C})||^2] \end{aligned} \tag{13}$$

The second term relates $\boldsymbol{C}'$ to $\boldsymbol{C}$ directly with a regularization factor α. $\boldsymbol{S}$ is a similarity matrix, which we will define shortly. The idea behind it is to allow costs to be freely changed among similar slots, but disallow this for dissimilar slots. The aim of $||\text{sinkhorn}(\boldsymbol{C}')\boldsymbol{S} - \text{sinkhorn}(\boldsymbol{C})||^2$ is thus to make sure that $\text{sinkhorn}(\boldsymbol{C}')$ looks the same as $\text{sinkhorn}(\boldsymbol{C})$ after allowing weight in the transport map to be moved around among similar slots.

Example Consider the case where we have three slots: $\boldsymbol{Z} = [\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}]$ and three inputs $\boldsymbol{X} = [\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}]$. Let us assume for this example that the cost matrix prefers associating $\boldsymbol{\gamma}$ with $\boldsymbol{y}$ and both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ with $\boldsymbol{x}$. Computing unregularized optimal transport solutions would therefore give us either of two solutions:

$$\boldsymbol{T}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad \boldsymbol{T}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

However, the Sinkhorn algorithm is unable to break the tie between the two $\boldsymbol{x}$ slots, so even with a temperature approaching 0, we obtain the following result:

$$\text{sinkhorn}(\boldsymbol{C}) = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Suppose we have a similarity matrix $\tilde{\boldsymbol{S}} \in \mathbb{R}^{m \times m}$ (m is the number of slots) that measures pairwise similarities ranging from 0 to 1:

$$\tilde{\boldsymbol{S}} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

The first two slots are similar amongst themselves but dissimilar to the $\boldsymbol{y}$ slot. If we normalize each column of $\tilde{\boldsymbol{S}}$ to sum to 1 to obtain $\boldsymbol{S}$, then we see the following:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{T}_1} \underbrace{\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{S}} = \underbrace{\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{sinkhorn}(\boldsymbol{C})} \tag{17}$$

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{T}_2} \underbrace{\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{S}} = \underbrace{\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{sinkhorn}(\boldsymbol{C})} \tag{18}$$

This means that $\text{sinkhorn}(\boldsymbol{C}') = \boldsymbol{T}_1$ and $\text{sinkhorn}(\boldsymbol{C}') = \boldsymbol{T}_2$ are both valid solutions for the minimization of $||\text{sinkhorn}(\boldsymbol{C}')\boldsymbol{S} - \text{sinkhorn}(\boldsymbol{C})||^2$. Note that any other permutation matrix for $\boldsymbol{T}$ (i.e. one where there is not a 1 in the bottom right corner) would not be a valid solution. This restricts Equation 9 to only consider transport maps that are convex combinations of $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ for this example, with the entropy minimization preferring $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ specifically. The small amount of noise in the $\boldsymbol{C}'$ initialization arbitrarily makes it prefer one of the two.

Definition We define the similarity matrix $\tilde{S}_{ij} = g(\boldsymbol{Z}_i, \boldsymbol{Z}_j)$, where $g : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ is a small neural network that takes pairs of slots as input and produces a similarity score as output. We then normalize each column of $\tilde{\boldsymbol{S}}$ to sum to 1 by applying softmax on each column.

$$\boldsymbol{S} = \text{softmax}(\tilde{\boldsymbol{S}}) \tag{19}$$

If we set up a training task for the example described above, we observe that $\boldsymbol{S}$ is learned to be virtually the same as the $\boldsymbol{S}$ we use in the example.

C. Multiset-equivariance of SA-SH, SA-EMD, and SA-MESH

First, we show that SA-SH is set-equivariant, and therefore not exclusively multiset-equivariant.

Proposition C.1. *SA-SH is set-equivariant.*

Proof. Slot attention is set-equivariant (Locatello et al., 2020). All the additional operations in SA-SH are set-equivariant for a similar reason to Deep Sets (Zaheer et al., 2017): only sum, broadcast, and elementwise operations are used in Sinkhorn. Since composition of set-equivariant operations maintains set-equivariance, SA-SH is set-equivariant. □

Proposition C.2. *SA-EMD and SA-MESH are exclusively multiset-equivariant.*

Proof. To show exclusive multiset-equivariance, we need to show that they are not set-equivariant, but still multiset-equivariant. We begin with the former.

To show that SA-EMD and SA-MESH are not set-equivariant, it is enough to give a counter-example. Suppose we have the following cost matrix:

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{20}$$

Running EMD gives us one of the following transport maps as the solution, depending on the arbitrary tiebreaking in the EMD implementation.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{21}$$

Both have zero entropy, hence they are also possible solutions when the MESH objective is perfectly optimized. Set-equivariance requires that a permutation applied to the input in Equation 20 changes the output by the same permutation. As pointed out by Zhang et al. (2022), this does not happen because the arbitrary tiebreaking remains the same. Therefore, both SA-EMD and SA-MESH are not set-equivariant.

To show that they are multiset-equivariant, first recall the definition of multiset-equivariance.

$$\begin{aligned} &\forall \boldsymbol{X} \in \mathbb{R}^{n \times c}, \forall \boldsymbol{P}_1 \in \Pi, \exists \boldsymbol{P}_2 \in \Pi : \\ &f(\boldsymbol{P}_1 \boldsymbol{X}) = \boldsymbol{P}_2 f(\boldsymbol{X}) \ \wedge \ \boldsymbol{P}_1 \boldsymbol{X} = \boldsymbol{P}_2 \boldsymbol{X}. \end{aligned} \tag{22}$$

EMD produces a solution with the minimum total cost by definition. This means that the transport map must remain the same, up to permutation. This is because if any of the values in the transport map were to change (aside from being permuted), then the original solution was not a minimum, which is a contradiction. Therefore, we know that a $\boldsymbol{P}_2$ must exist for any $\boldsymbol{P}_1$.

In the same way with MESH, with infinitesimally small noise, the only thing that can change is the permutation of the solution: if no ties are broken the noise has virtually no effect because the subsequent operations in MESH are continuous, if a tie is broken then the ordering of the tie is random. In either case, we can again always find a $\boldsymbol{P}_2$ for every $\boldsymbol{P}_1$ on the inputs, since the values in the solution remain the same up to permutation.

We have thus shown that SA-EMD and SA-MESH are not set-equivariant, but are multiset-equivariant (i.e. exclusively multiset-equivariant). □

D. Sinkhorn algorithm implementation

Ideally, we want to compute the Sinkhorn algorithm for as few steps as possible since it is used in every MESH step, which means that we also have to differentiate through the Sinkhorn algorithm in every MESH step. However, we also need to run it for a sufficient number of steps for (good enough) convergence. Fortunately, because we repeatedly run the Sinkhorn algorithm on *similar* inputs over different MESH steps, we can optimize its implementation. The idea is that the gradient descent for minimizing the entropy makes small changes (especially near MESH convergence), which allows us to reuse computation between different MESH steps.

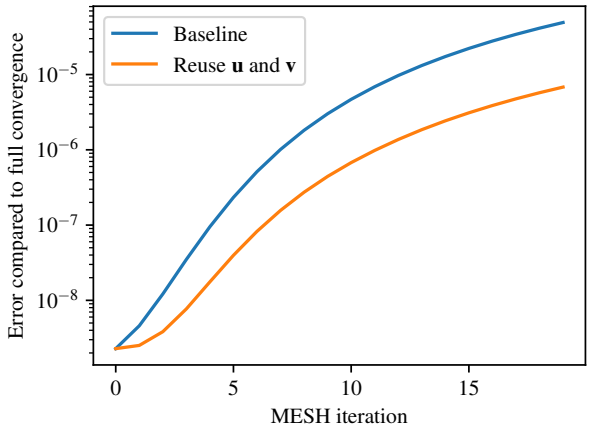


Figure 3. Mean absolute error gap to the fully converged Sinkhorn for the two different Sinkhorn implementations for varying numbers of MESH iterations. Both implementations always use 5 Sinkhorn iterations and thus have comparable computational costs. Reusing $\boldsymbol{u}$ and $\boldsymbol{b}$ is clearly more effective.

If the entropy minimization has converged, then the Sinkhorn output does not change. This means that if we keep track of the operations we performed in the previous SA-MESH iteration, we can simply reapply them. Let us take a look at these operations: the Sinkhorn algorithm (multiplicatively) rescales rows and columns repeatedly. Multiplication is commutative, so we can collect all the row normalizations together into a single row normalizer $\boldsymbol{u}$ and all the column normalizations into a single column normalizer $\boldsymbol{v}$. Applying these two normalizers on the (exponentiated) cost matrix gives us exactly the same result as if we had manipulated the matrix with the normalizations directly. The next time we run the Sinkhorn algorithm with a slightly changed cost matrix, we can bootstrap the algorithm with the previously-found $\boldsymbol{u}$ and $\boldsymbol{v}$. We can then run a few more iterations to account for the changes in the cost matrix, giving us a new $\boldsymbol{u}$ and $\boldsymbol{v}$.

In Figure 3, we ablate whether there are benefits to reusing $\boldsymbol{u}$ and $\boldsymbol{v}$ in SA-MESH. In particular, we compute the ideal solution by running the Sinkhorn algorithm until convergence at every ME iteration. Then we examine whether reusing $\boldsymbol{u}$ and $\boldsymbol{v}$ helps close the gap to the ideal solution

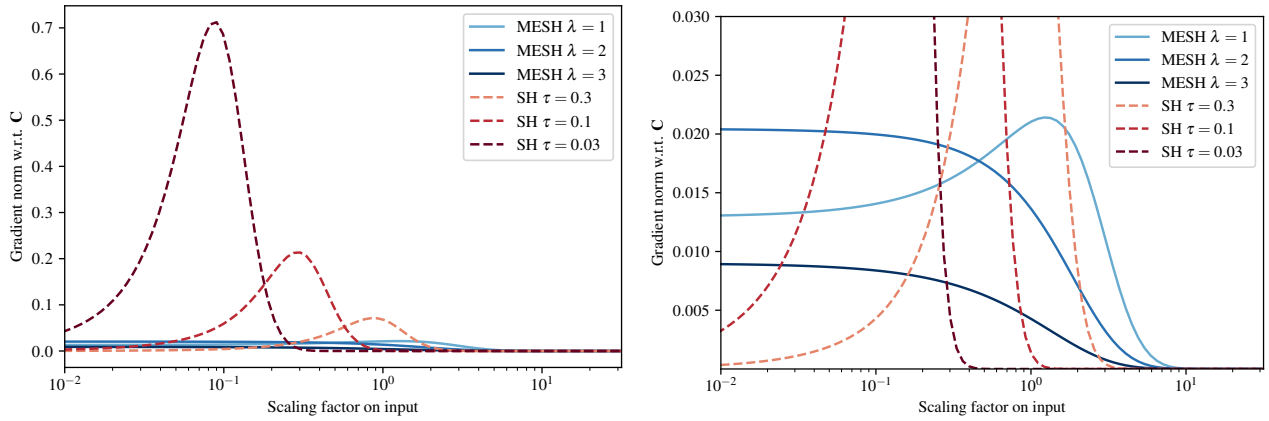


Figure 4. Same gradient norm plot as Figure 1 but without normalizing each model to have a maximum of 1. The only difference between left and right is the scale of the y-axis. This shows that the gradient norm for tempered SH varies drastically with temperature, while the gradient norm of MESH remains more similar across learning rates. This makes tuning the learning rate hyperparameter easier as it can be considered more independently from other hyperparameters of the model.

when we limit the number of SH iterations. At 1 ME iteration there are no $\boldsymbol{u}$ and $\boldsymbol{v}$ for bootstrapping available, so both exhibit the same gap. At more than 1 ME iteration we observe that reusing $\boldsymbol{u}$ and $\boldsymbol{v}$ helps in narrowing the gap to the ideal solution, or equivalently, can achieve the same approximation with fewer iterations.

We do not claim that this technique of collecting Sinkhorn operations into $\boldsymbol{u}$ and $\boldsymbol{v}$ is novel, as there are several implementations that use this trick for performing the Sinkhorn algorithm. Usually, it is a minor implementation detail since the approaches of manipulating the matrix directly and collecting normalizations into $\boldsymbol{u}$ and $\boldsymbol{v}$ are mathematically equivalent. In our case however, this formulation leads to a concrete benefit due to our setup where we run the Sinkhorn algorithm on similar inputs, which allows us to reuse $\boldsymbol{u}$ and $\boldsymbol{v}$ in a beneficial way.

E. Gradients of tempered SH and MESH, without normalizing their scale

In Figure 4, we show the same gradient norms as in Figure 1. However, rather than normalizing each model to have a maximum of 1 (which is more useful for visualizing them all at once), we maintain their native scaling. This shows that changing the SH temperature has a major effect on the scale of the gradients while changing the MESH learning rate only has minor effects on the gradients (but still significant effects on entropy reduction). In other words, the amount of entropy minimization can be changed without major impacts on the hyperparameters of other parts of the network.

F. Experimental details

F.1. Random object detection

We generate a dataset of 64,000 data points to train on, each being a multiset with five 32-dimensional objects sampled from $\mathcal{N}(0, \sigma^2 \boldsymbol{I})$ and 100 zero vectors. We directly apply slot attention on this: the dimensionality of the slot attention weights are all 32. Since we know that there are always five objects, we set the number of slots to five. The loss is computed by computing a mean squared error between all pairs of predicted and ground-truth objects, then using the Hungarian algorithm find the matching with the lowest loss. We find that using implicit differentiation of slots (Chang et al., 2022) is significantly more stable on this dataset, so we use it for all models. We train all models for 20 epochs with a batch size of 64 (1,000 steps each epoch).

F.2. Object detection on CLEVR

We largely follow same training setup as DSPN and iDSPN (Zhang et al., 2019; 2022) and adapt the slot attention implementation to it. Matching Zhang et al. (2022) and Locatello et al. (2020), we resize the input images to 128x128. To compute the loss, we use the Hungarian algorithm to compute the least-cost matching between predicted objects and ground-truth objects.

F.3. Object discovery

We compute the temporal consistency (TC) by first matching the predicted objects in each frame to their corresponding ground-truth objects. Then, we calculate the proportion of objects that have the same slot in both frames out of those that appear in both frames. To find the optimal assign-

ment between the predicted and ground-truth objects, we first compute all pairwise IoUs between the predicted and ground-truth masks. We then invert these IoUs by applying $1 - \text{IoU}$ and use the Hungarian matching algorithm to find the best match—the one with the highest mIoU.

We compute the mIoU in a similar manner as the TC metric, by finding the matching between the ground-truth objects and the predicted objects that results in the highest mIoU.

Multi-dSprites. We closely follow the experimental setup described by Locatello et al. (2020). Specifically, we use the same image encoder, decoder, and hyperparameters where applicable. Locatello et al. (2020) used 500k training steps, while all of our runs were trained for 530 epochs, which results in slightly fewer than 500k training steps.

ClevrTex. We pre-process the images by applying the same center crop as suggested by Karazija et al. (2021) and resize the images to 64×64 resolution instead of 128×128 resolution. This allows us to use the same neural network architecture as we did for the Multi-dSprites dataset.

Since the dataset has more complicated visuals we increase the model size by increasing the channel sizes. In particular, we double the number of channels in the image encoder and decoder to 64, and we double the dimensions of the slots to 128 (with the MLP in slot attention having an intermediate dimension of 256). We again train all models for 530 epochs which correspond to around 330k gradient update steps in this case. The maximum number of objects in an image is 10, so we set the number of slots to 11.

F.4. CLEVRER

We construct the datasets from the 20k videos in the CLEVRER dataset, by picking two frames at specific timesteps from each video. In the CLEVRER-S dataset, we use the first and 16th video frames. In the second dataset, we use the first and 128th (last) frames. The proportion of examples where new objects appear increases with the time gap between the two frames, and similarly the amount of displacement for objects that are in both frames increases too. We show examples from the two dataset variants in Figure 6.

For the distance function d that computes the cost matrix of the optimal transport problem in SA-MESH, we empirically find that the cosine distance works better in this case than the $l2$ distance. We suspect that since the $l2$ distance allows the slots to be pushed arbitrarily far apart that learning might slow down in the later stages of training.

We extend the model which we used in the Multi-dSprites experiment to video data. Our setup is similar to Kipf et al. (2022), but we do not use a predictor model (except for the SA learned noise baseline) to update the slots when transitioning from one video frame to the next. In particular, the model first applies the image encoder to all video frames independently to compute the input feature maps. Next, the SA (or our proposed variants) is applied to the features of one video frame at a time, and every time the slots are initialized from the slots of the previous frame. Finally, each image is decoded independently. The learned noise baseline uses a 2-layer MLP with LayerNorm to predict the mean and variance of a Gaussian, from which the initial slots are sampled for the next frame, following the stochastic SAVi setup by Wu et al. (2022). We use 8 slots.

G. CLEVR object prediction results

In Table 6 we show our results for CLEVR object prediction without implicit differentiation of slots (Chang et al., 2022). All results are slightly lower than the results reported in Table 2, but the overall message remains exactly the same. The only major difference is that SA-SH performs much worse compared to SA-SH†.

H. Extra ablations

Additional slot attention iterations In general the benefit of more iterations is minor (see ablations in Locatello et al. (2020), Appendix C) and using too many can hurt in some cases, which is why many recent works (Wu et al., 2022; Kipf et al., 2022) set the number of iterations to 3 or even fewer. We ran an additional experiment where we trained the plain slot attention baseline on Multi-dSprites with 5 iterations resulting in 81.9±6.1 FG-ARI, which is worse than the 92.2±0.5 achieved with 3 iterations reported in our main results. Also note that the slot attention module is only a part of the full neural network and is not the bottleneck when using larger encoders.

Learned slot initializations In our perspective, the initialization should be thought of as separate to the slot attention method itself. A different initialization does not change the fact that the slots can collapse, especially in cases like the video datasets where the initialization is not a free parameter but dependent on the previous timestep. Thus, having control over the initialization should not be relied upon.

Locatello et al. (2020) report in their Appendix B that learning the initial slots decreases the performance in unsupervised learning. We ran experiments with SA using a learned initialization on Multi-dSprites to evaluate this as well. On the Multi-dSprites dataset, SA with a learned initialization achieves 93.0±1.0 FG-ARI, which is comparable to the standard SA at 92.2±0.5 and remains lower than the 95.6±0.2 of SA-MESH.

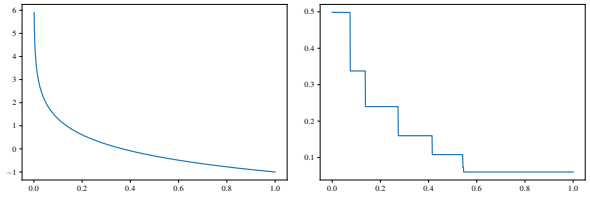
Table 6. Results on CLEVR object property multiset prediction, average precision (AP) in % (mean ± standard deviation) over 5 random seeds, higher is better. All SA results are based on our re-implementation. SA (original) results copied from Locatello et al. (2020), iDSPN results from Zhang et al. (2022).

Model	AP_∞	AP_1	$AP_{0.5}$	$AP_{0.25}$	$AP_{0.125}$	$AP_{0.0625}$	Time
iDSPN (Zhang et al., 2022)	*98.8*$_{\pm 0.5}$	*98.5*$_{\pm 0.6}$	*98.2*$_{\pm 0.6}$	*95.8*$_{\pm 0.7}$	*76.9*$_{\pm 2.5}$	*32.3*$_{\pm 3.9}$	—
SA (original) (Locatello et al., 2020)	94.3$_{\pm 1.1}$	86.7$_{\pm 1.4}$	56.0$_{\pm 3.6}$	10.8$_{\pm 1.7}$	0.9$_{\pm 0.2}$	—	—
SA	89.1$_{\pm 1.2}$	85.7$_{\pm 1.0}$	73.3$_{\pm 1.2}$	35.4$_{\pm 1.5}$	9.0$_{\pm 0.8}$	2.0$_{\pm 0.3}$	2.4 h
SA-SH	95.6$_{\pm 1.0}$	94.0$_{\pm 1.1}$	84.5$_{\pm 1.7}$	41.3$_{\pm 3.0}$	10.4$_{\pm 0.7}$	2.5$_{\pm 0.4}$	2.5 h
SA-EMD	99.2$_{\pm 0.2}$	98.7$_{\pm 0.4}$	97.0$_{\pm 0.8}$	82.4$_{\pm 1.2}$	34.0$_{\pm 2.2}$	8.3$_{\pm 0.9}$	9.7 h
SA-MESH	**99.2**$_{\pm 0.3}$	**99.1**$_{\pm 0.3}$	**98.8**$_{\pm 0.5}$	**88.3**$_{\pm 0.8}$	**40.8**$_{\pm 1.0}$	**10.6**$_{\pm 0.3}$	2.5 h

I. Alternative MESH objective

In Section 4 we choose the entropy as the inner objective function because the goal was to reverse the effect of the entropy-regularized optimal transport version. Alternatively, it is possible to learn a neural network with scalar inputs and outputs in place of the entropy function. For the neural network, we choose a simple 2-layer MLP with ReLU activations and 32 hidden dimensions. We plot the derivative of a learned objective function in Figure 5b. Its shape is similar to the derivative of the entropy Figure 5a, but learning it incurs additional compute compared to simply using the entropy function H.

Justified by this analysis we can directly use the entropy function as the MESH objective for improved efficiency. Empirically we observe that we do not even need to backpropagate through the gradient descent optimization procedure of MESH and it suffices to treat MESH as the identity function during backprop. One perspective that might explain this: since the negative derivative of H is a monotonic increasing function, changes to the input will also affect the output in the same direction.



(a) H (b) MLP

Figure 5. Derivative of the objective function in MESH

J. Object discovery example results

In the following, we show examples of SA and SA-MESH performing object discovery on the various datasets that we use. We always show the original image on the left, followed by either the attention maps for each slot or the final alpha masks for each slot. The attention map or the alpha masks are multiplied with the original image to make it easier to tell how precise their locations are.

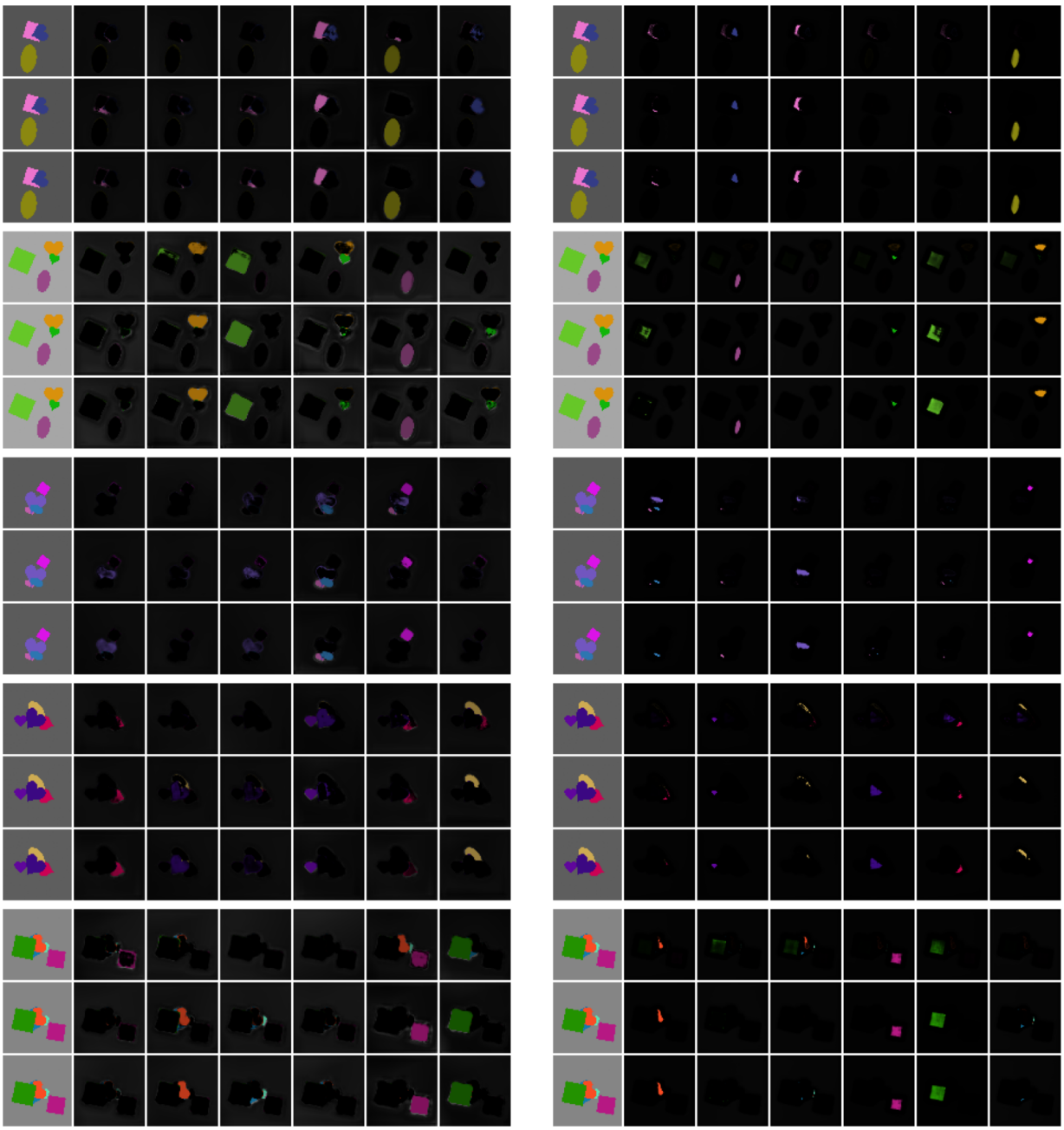
- Figure 7 shows the intermediate attention maps over the three slot attention iterations on the Multi-dSprites dataset.
- Figure 8 shows the final per-slot alpha masks on the Multi-dSprites dataset.
- Figure 9 shows the intermediate attention map of the last slot attention iteration, as well as the final per-slot alpha masks on the ClevrTex dataset.
- Figure 10 shows the final per-slot alpha masks on the CLEVRER-S and CLEVRER-L datasets.

Please refer to the individual figure captions for a more detailed description of observations on these results.

(a) CLEVRER-S

(b) CLEVRER-L

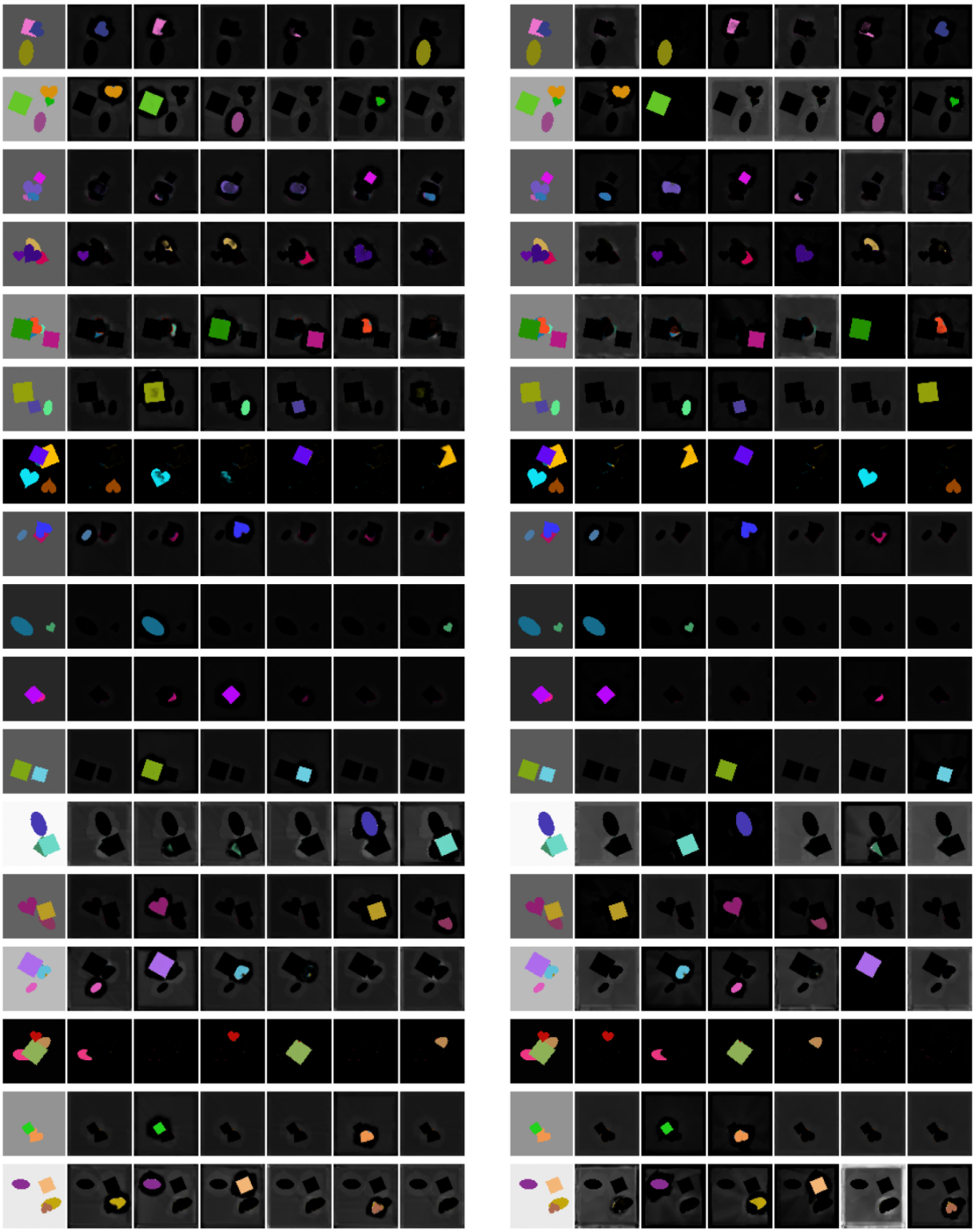
Figure 6. Examples from the two datasets derived from CLEVRER. Significant changes like multiple new objects appearing occur less frequently in CLEVRER-S. Objects can be significantly displaced from one frame to the other in CLEVRER-L.



(a) SA

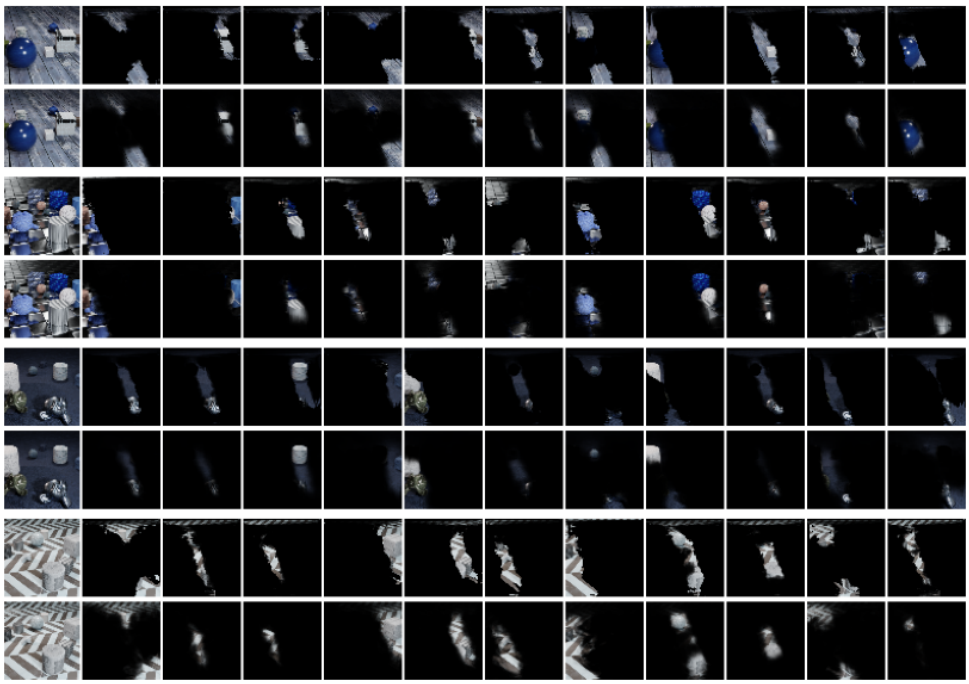
(b) SA-MESH

Figure 7. Attention maps for all three slot attention iterations for five different examples from the validation split of Multi-dSprites. Note how the shade of SA is generally darker indicating higher attention values even in background areas. In the third example, we can see how SA models the pink and blue ellipses using one slot while splitting the purple heart over two slots. In contrast, SA-MESH is able to route the three objects into three different slots in the second slot attention iteration.

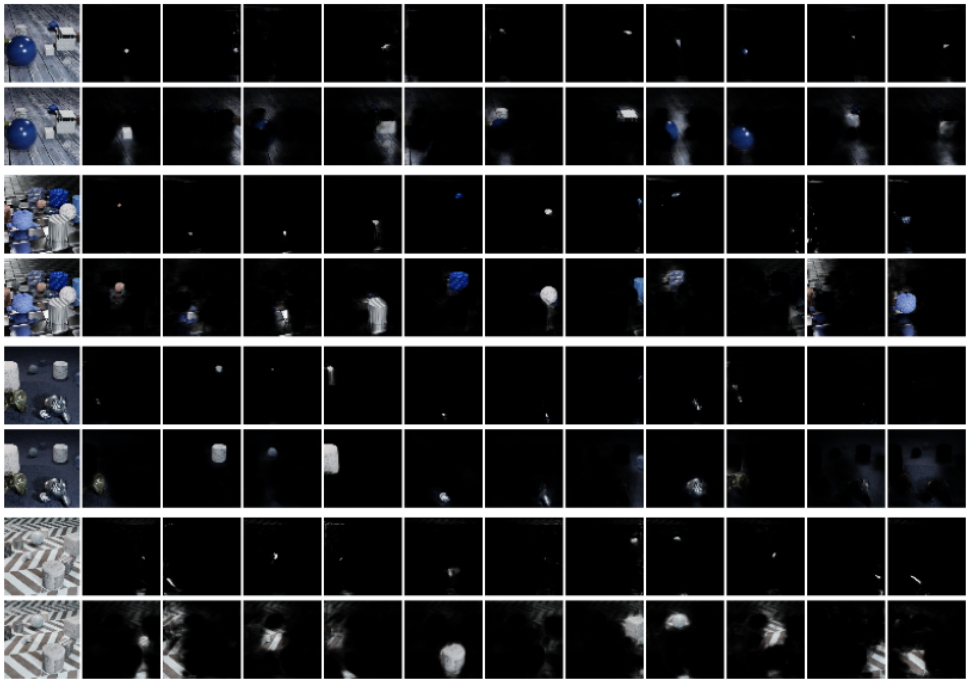


(a) SA (b) SA-MESH

Figure 8. Predicted alpha masks from the validation split of Multi-dSprites. In general, the masks for SA-MESH are sharper than for SA. For example, in the last row, the brown heart is only recognizable as a blown blob for SA, but is a distinct heart shape for SA-MESH. In the third example, SA splits the purple heart into two slots, while SA-MESH models it with one as desired.



(a) SA



(b) SA-MESH

Figure 9. Attention maps (top) and masks (bottom) for four different examples from the validation split of ClevrTex. SA commonly learns that each slot should attend to a spatial region as opposed to a specific object. SA-MESH on the other hand is better able to localize individual objects, though the background is often still handled with a region-specific approach.

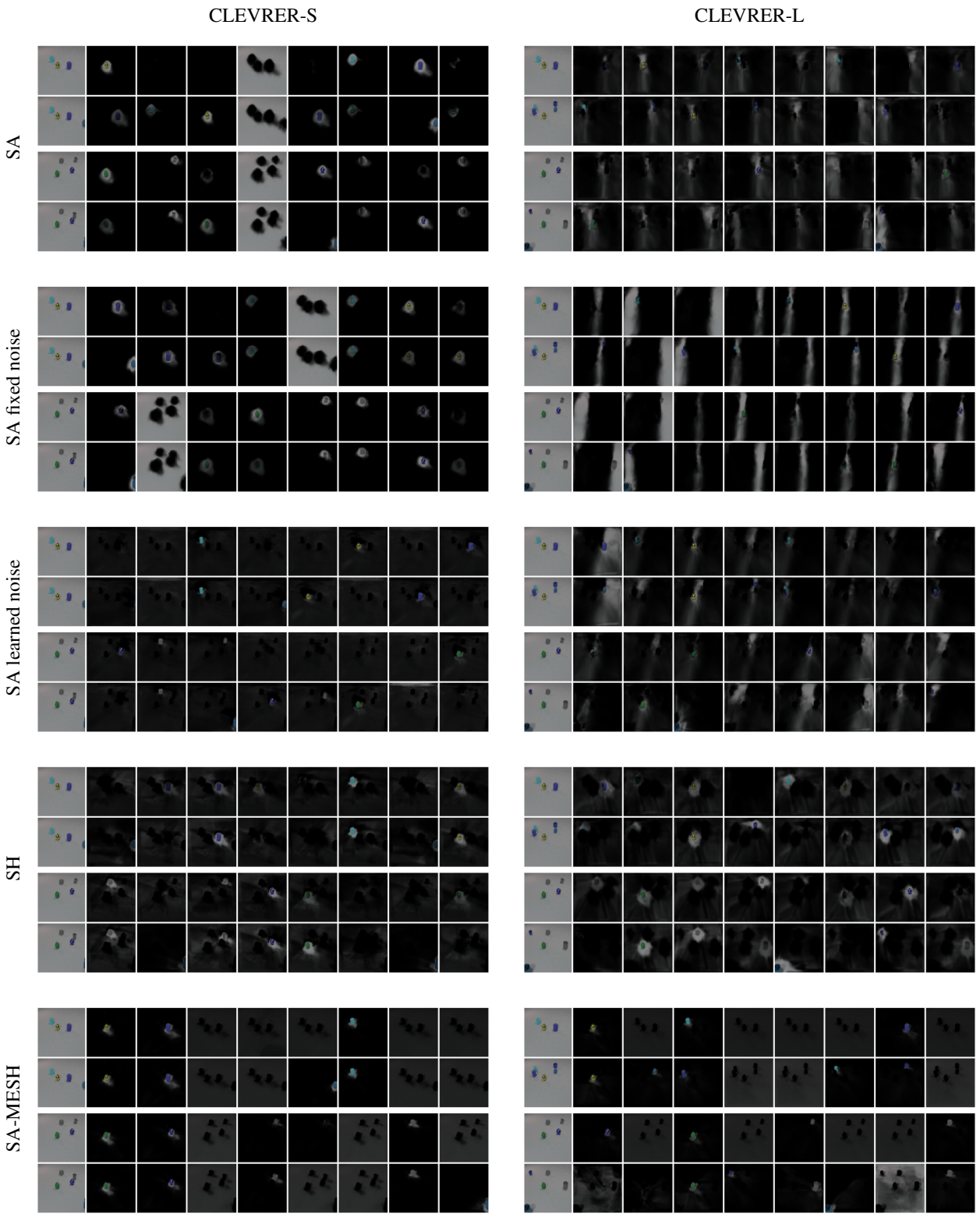


Figure 10. Example alpha masks for CLEVRER-S and CLEVRER-L. Possibly due to the difficulty of handling multiple new objects, SA and its noise variants choose a region-based decomposition on CLEVRER-L instead of an object-based decomposition like on CLEVRER-S. SA-SH and SA-MESH learn an object-based decomposition for both datasets.