SLOTODE: SLOT TRAJECTORY SEPARATION FOR COLLISION REDUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Object-Centric Learning (OCL) aims to decompose a scene into slot representations corresponding to individual objects, and Slot Attention is a representative framework that iteratively updates these slots. However, how slot representations evolve across updates and whether they maintain separation remain unclear. For clarity, we define *slot-state collision* as the case in which initially different slot representations become identical during iterative updates, so that the collided slots can no longer maintain different representations for different objects. To address this problem, we propose **SlotODE**, which formulates iterative slot updates as an ordinary differential equation (ODE). Instead of directly predicting the next slot representations, SlotODE learns a vector field that defines the time derivatives of the slot representations. Our theoretical analysis shows that SlotODE prevents slot-state collision over any finite update interval, and further derives an explicit lower bound on pairwise slot separation. In experiments, SlotODE significantly improves object discovery performance across OCL benchmarks. In-depth analyses demonstrate the impact of the ODE formulation and its components, and the robustness of the learned vector field across different ODE solvers and numbers of update steps.

1 INTRODUCTION

Object-Centric Learning (OCL) aims to decompose a scene into a set of object-centric latent representations, where different slots can capture different objects or entities in the scene. Slot Attention (Locatello et al., 2020) is a representative OCL framework that represents these object-centric latent representations as slots and iteratively updates them based on attention weights normalized across slots and feature aggregation. Despite advances in slot-based OCL, learning object-centric representations across multiple slots remains challenging. Prior work has reported redundant slot representations (Zhao et al., 2025b; Liu et al., 2026), overlapping attentions (Zhang et al., 2023), and difficulties in inducing different roles across slots (Jia et al., 2022; Kori et al., 2024).

Previous works have addressed these challenges by introducing slot-level constraints or redundancy control (Fan et al., 2024; Liu et al., 2026; Zhao et al., 2025b), or improving slot initialization (Jia et al., 2022; Kori et al., 2024; Zhao et al., 2025a). While these studies mainly consider how to obtain slot representations, how slot representations evolve across iterative updates during object-centric representation learning remains underexplored. How slots evolve across iterative updates is important, as changes to the intermediate update process have been shown to affect the resulting slot representations (Zhao et al., 2025b).

A few studies have analyzed or modified the iterative slot update process itself (Zhang et al., 2023; Chang et al., 2022; Kirilenko et al., 2024). However, whether multiple slot representations remain separate throughout this process has not been well established. Maintaining this separation is important for preserving different slot representations corresponding to individual objects throughout the update process. To make this issue precise, we define *slot-state collision* as a case in which initially different slot representations become identical during iterative updates. We refer to the representation of the i-th slot at an update point t as its slot state $\boldsymbol{s}_i(t)$. Once a collision occurs, conventional permutation-equivariant slot updates preserve this equality in subsequent updates, preventing the two slot representations from separating again.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To address slot-state collision, we propose **SlotODE**, which reformulates the iterative slot updates of Slot Attention as an ordinary differential equation (ODE) over the joint slot state $\boldsymbol{S}(t) \in \mathbb{R}^{K \times D}$. Rather than directly predicting the next slot states, SlotODE learns a vector field that defines the time derivatives of the slot states. Starting from the initial slot state, an ODE solver numerically integrates this vector field over the update interval to obtain the updated slot states.

Our theoretical analysis shows that SlotODE prevents slot-state collision throughout the update process. This guarantee follows from the uniqueness of ODE solutions, which prevents solutions starting from different initial states from merging during the update process. Combined with permutation equivariance, this ensures that two initially different slot states cannot become identical, as formalized in **Theorem 1**. In contrast, conventional discrete slot updates can map different states to identical states and therefore do not generally provide this guarantee. We further derive an explicit lower bound on pairwise slot separation throughout the interval in **Theorem 2**. In experiments, SlotODE significantly improves object discovery performance over conventional Slot Attention. We further evaluate SlotODE using a previously proposed adaptive slot selection method, which adjusts the number of active slots, and observe strong performance relative to existing OCL methods. In-depth analyses demonstrate the impact of the ODE formulation and its components, as well as the robustness of the learned vector field across different ODE solvers and numbers of update steps.

Our contributions are threefold. (1) We define *slot-state collision* as a case in which initially different slot representations become identical during iterative updates, and propose **SlotODE**, which reformulates the iterative updates of Slot Attention as an ODE over slot states. (2) We prove that SlotODE prevents slot-state collision and further provide an explicit lower bound that specifies the guaranteed pairwise slot separation throughout the update process. (3) We empirically show that SlotODE significantly improves object discovery performance across OCL benchmarks, and demonstrate the impact of the ODE formulation and its components, as well as the robustness of the learned vector field across different ODE solvers and numbers of update steps.

2 Related Work

Challenges in learning object-centric slot representations. In slot-based OCL, learning object-centric representations across multiple slots remains a challenge. Some studies address this challenge by reducing slot redundancy or adapting the number of active slots (Behjati & Henderson, 2021; Fan et al., 2024; Liu et al., 2026; Zhao et al., 2025b; Ouyang et al., 2026), improving initialization and optimization (Jia et al., 2022; Kori et al., 2024; Zhao et al., 2025a), or explicitly encouraging distinct slot representations (Didolkar et al., 2023). Other approaches improve attention and feature assignment (Zhang et al., 2023; Kakogeorgiou et al., 2024), modify slot update mechanisms (Kirilenko et al., 2024), or use reconstruction-based objectives (Moon et al., 2026; Seong et al., 2026). In this work, we focus on whether slot representations remain separated across iterative updates.

Iterative updates of Slot Attention. Iterative updates have become a common approach in OCL, including IODINE (Greff et al., 2019) and Slot Attention (Locatello et al., 2020). Within Slot Attention, subsequent studies have analyzed or modified different aspects of this update process. Chang et al. (2022) interpret object representations as fixed points of iterative updates and employ implicit differentiation to improve the stability of training. Zhang et al. (2023) interpret attention-based feature assignment through optimal transport and modify it through sparsity regularization. Kirilenko et al. (2024) interpret Slot Attention as a learnable clustering procedure and reformulate the slot update using a Gaussian mixture model. Other studies improve the initialization, optimization, or stability of repeated updates (Jia et al., 2022; Zhao et al., 2025a), modify the update procedure through self-distillation (Zhao et al., 2025b), or introduce an attention-refining kernel (Kim et al., 2023). While these studies analyze or improve different aspects of iterative slot updates, it remains unclear whether multiple slot representations remain separate as they evolve across successive updates.

3 Preliminaries: Slot Attention

Given a set of input features $\boldsymbol{X} = \{\boldsymbol{x}_m\}_{m=1}^{M}$, Slot Attention maintains K latent slots $\boldsymbol{S}^{(n)} = [\boldsymbol{s}_1^{(n)}, \ldots, \boldsymbol{s}_K^{(n)}]$, where $\boldsymbol{s}_i^{(n)} \in \mathbb{R}^D$ denotes the i-th slot at iterative step n out of a total N steps. At

each step, the current slots act as queries and the input features as keys and values in the attention operation. The resulting attention weights are normalized across slots for each input feature, determining how strongly each feature is assigned to the slots. These weights are then used to aggregate the value features into slot-specific evidence $\boldsymbol{U}^{(n)} = \mathcal{A}(\boldsymbol{S}^{(n)}, \boldsymbol{X})$. The slots are then updated recurrently using a GRU (Cho et al., 2014) followed by a residual Multi-Layer Perceptron (MLP). We represent this update as the learned transition $\boldsymbol{S}^{(n+1)} = G_\theta(\boldsymbol{S}^{(n)}, \boldsymbol{X})$, where the same update function G_θ is shared across all N update steps. Repeated application of G_θ progressively updates the initial slots into the final slot representations, where the initial slots are sampled from a Gaussian distribution with learnable parameters $\mu_\theta, \sigma_\theta \in \mathbb{R}^D$. The final slots are decoded to reconstruct the input image, and the model is trained by minimizing the reconstruction loss.

4 Slot-State Collision and SlotODE

In this section, we define *slot-state collision* and propose **SlotODE**, which reformulates iterative slot updates as an ordinary differential equation over the joint slot state. We then prove that SlotODE prevents slot-state collision and further derive an explicit lower bound on pairwise slot separation. Finally, we describe the architecture used to implement SlotODE.

4.1 Slot-State Collision

We focus on whether multiple slot representations remain separate as they evolve across iterative updates. We define *slot-state collision* as follows. Given any pair of slots $i \neq j$ with distinct initial states $\boldsymbol{s}_i(0) \neq \boldsymbol{s}_j(0)$, we say that a slot-state collision occurs if there exists some τ^* at which their states become identical:

$$\boldsymbol{s}_i(0) \neq \boldsymbol{s}_j(0), \qquad \boldsymbol{s}_i(\tau^*) = \boldsymbol{s}_j(\tau^*) \quad \text{for some } i \neq j. \tag{1}$$

Here, we use τ as a unified notation for the discrete update step n in conventional Slot Attention and continuous time t in SlotODE. Since Slot Attention repeatedly applies a shared update process, the intermediate slot states from one step serve as the inputs to the next. Therefore, changes in these states affect subsequent updates and the final slot states. However, the conventional learned transition G_θ does not generally satisfy the following separation preserving property: $\boldsymbol{s}_i^{(n)} \neq \boldsymbol{s}_j^{(n)} \implies \boldsymbol{s}_i^{(n+1)} \neq \boldsymbol{s}_j^{(n+1)}$.

To make this limitation precise, let $\mathcal{C}_{ij} = \{\boldsymbol{S} : \boldsymbol{s}_i = \boldsymbol{s}_j\}$ denote the pairwise collision set and $\boldsymbol{P}_{ij}$ denote the permutation matrix that swaps slots i and j. For a permutation equivariant update process G_θ, any $\boldsymbol{S} \in \mathcal{C}_{ij}$ satisfies $\boldsymbol{P}_{ij}\boldsymbol{S} = \boldsymbol{S}$. By permutation equivariance, $G_\theta(\boldsymbol{S}, \boldsymbol{X}) = G_\theta(\boldsymbol{P}_{ij}\boldsymbol{S}, \boldsymbol{X}) = \boldsymbol{P}_{ij}G_\theta(\boldsymbol{S}, \boldsymbol{X})$, which implies $G_\theta(\boldsymbol{S}, \boldsymbol{X}) \in \mathcal{C}_{ij}$. Thus, the collision set is forward invariant: once two slot states become identical, subsequent permutation-equivariant updates preserve their equality. However, the complement of $\mathcal{C}_{ij}$ need not be forward invariant. In general, $\boldsymbol{S} \notin \mathcal{C}_{ij}$ does not guarantee $G_\theta(\boldsymbol{S}, \boldsymbol{X}) \notin \mathcal{C}_{ij}$, so permutation equivariance alone does not prevent distinct slot states from entering the collision set.

4.2 ODE Formulation of Iterative Slot Updates

We reformulate iterative slot updates as an ordinary differential equation over the joint slot state, following the standard neural ODE formulation (Chen et al., 2018). Let $\boldsymbol{s}_i(t) \in \mathbb{R}^D$ denote the state of the i-th slot at time t, and let $\boldsymbol{S}(t) = [\boldsymbol{s}_1(t), \ldots, \boldsymbol{s}_K(t)] \in \mathbb{R}^{K \times D}$ denote their joint slot state. In contrast to the conventional transition G_θ, which directly maps the current slot state to the next state, SlotODE defines the derivative of the joint slot state through an ordinary differential equation:

$$\frac{d\boldsymbol{S}(t)}{dt} = F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t), \qquad \boldsymbol{S}(0) = S_0, \qquad \boldsymbol{S}(T) = S_0 + \int_0^T F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t)\, dt. \tag{2}$$

Here, F_θ denotes the SlotODE vector field, which takes the current joint slot state $\boldsymbol{S}(t)$, the input features $\boldsymbol{X}$, and the time t as inputs and determines the time derivatives of the slot states, while $\boldsymbol{X}$ remains fixed throughout the update interval. Integrating the vector field over $t \in [0, T]$ defines the slot states $\boldsymbol{S}(t)$ throughout the interval, including the terminal state $\boldsymbol{S}(T)$.

SlotODE retains the attention-based mechanism of Slot Attention. At each time t, the current slot configuration produces slot-specific feature evidence through attention-based feature aggregation.

The shared network f_θ then takes the current joint slot state, the slot-specific aggregated evidence $\mathcal{A}(\boldsymbol{S}(t), \boldsymbol{X})$, and the time embedding $\phi(t)$:

$$F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t) = f_\theta(\boldsymbol{S}(t), \mathcal{A}(\boldsymbol{S}(t), \boldsymbol{X}), \phi(t)) \,. \tag{3}$$

Since the attention weights are normalized across slots, the aggregated evidence for each slot depends on the current states of all slots. SlotODE therefore defines a single ODE over the joint slot state, rather than independent ODEs for individual slots.

4.3 Guarantees on Pairwise Slot Separation in SlotODE

We now prove theoretical guarantees on pairwise slot separation in SlotODE. Our theoretical analysis first shows that, under standard regularity conditions and permutation equivariance, ODE solution uniqueness prevents initially distinct slot states from becoming identical at any finite time, thereby preventing slot-state collision. We then derive an explicit lower bound on pairwise slot separation under an additional finite-horizon Lipschitz condition. Formal assumptions and complete proofs are provided in Appendix A.

Theorem 1 (Finite-Time Slot-State Collision Exclusion)**.** *Let $\boldsymbol{S}(t)$ be a solution of the SlotODE dynamics on a finite update interval $[0, T]$. Suppose that F_θ is locally Lipschitz with respect to the joint slot state, continuous in time, and permutation equivariant over the slot dimension. Then, for every pair $i \neq j$,*

$$\boldsymbol{s}_i(0) \neq \boldsymbol{s}_j(0) \quad \Longrightarrow \quad \boldsymbol{s}_i(t) \neq \boldsymbol{s}_j(t), \qquad \forall t \in [0, T]. \tag{4}$$

If all slot states are pairwise distinct at initialization, Theorem 1 applies to every pair, ensuring that no two slot states become identical over $[0, T]$.

Proof sketch. For a pair $i \neq j$, let $\boldsymbol{P}_{ij}$ denote the permutation that swaps slots i and j, and define $\widetilde{\boldsymbol{S}}(t) = \boldsymbol{P}_{ij}\boldsymbol{S}(t)$. By permutation equivariance, $\widetilde{\boldsymbol{S}}(t)$ is also a solution of the same ODE, with the permuted initial state $\widetilde{\boldsymbol{S}}(0) = \boldsymbol{P}_{ij}\boldsymbol{S}(0)$. Suppose that a slot-state collision occurs at some finite time t^*, so that $\boldsymbol{s}_i(t^*) = \boldsymbol{s}_j(t^*)$. Then, $\boldsymbol{P}_{ij}\boldsymbol{S}(t^*) = \boldsymbol{S}(t^*)$, and hence $\widetilde{\boldsymbol{S}}(t^*) = \boldsymbol{S}(t^*)$. Since the ODE solution through a given state at time t^* is unique, the two solutions must coincide on their common interval of existence. In particular, $\widetilde{\boldsymbol{S}}(0) = \boldsymbol{S}(0)$, which implies $\boldsymbol{s}_i(0) = \boldsymbol{s}_j(0)$ and contradicts the assumed initial separation. Therefore, no slot-state collision can occur over $[0, T]$. The complete proof is given in Appendix A.2. □

Theorem 1 guarantees that initially distinct slot states do not collide over a finite time interval, but does not characterize how their pairwise separation evolves over time. Under an additional finite-horizon uniform Lipschitz bound, we derive the following lower bound.

Theorem 2 (Exponential Lower Bound on Pairwise Slot Separation)**.** *Let $\boldsymbol{S}(t)$ be a solution of the ODE on $[0, T]$, and suppose that F_θ is permutation equivariant. Further assume that there exists a permutation-invariant region Ω_T containing $\boldsymbol{S}(t)$ for all $t \in [0, T]$ and a finite constant L_T such that $\left\| F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) - F_\theta(\widetilde{\boldsymbol{S}}, \boldsymbol{X}, t) \right\|_F \leq L_T \left\| \boldsymbol{S} - \widetilde{\boldsymbol{S}} \right\|_F$ for all $\boldsymbol{S}, \widetilde{\boldsymbol{S}} \in \Omega_T$ and $t \in [0, T]$. Then, for every pair $i \neq j$ and all $t \in [0, T]$,*

$$\left\| \boldsymbol{s}_i(t) - \boldsymbol{s}_j(t) \right\|_2 \geq e^{-L_T t} \left\| \boldsymbol{s}_i(0) - \boldsymbol{s}_j(0) \right\|_2 . \tag{5}$$

Proof sketch. For a pair $i \neq j$, consider the slot-permuted trajectory $\widetilde{\boldsymbol{S}}(t) = \boldsymbol{P}_{ij}\boldsymbol{S}(t)$ and define $\boldsymbol{D}(t) = \boldsymbol{S}(t) - \widetilde{\boldsymbol{S}}(t)$. By permutation equivariance, $\boldsymbol{S}(t)$ and $\widetilde{\boldsymbol{S}}(t)$ are also a solution of the same ODE with initial states $\boldsymbol{S}(0)$ and $\widetilde{\boldsymbol{S}}(0) = P_{ij}\boldsymbol{S}(0)$, respectively. Since Ω_T is permutation invariant and $\boldsymbol{S}(t) \in \Omega_T$, we also have $\widetilde{\boldsymbol{S}}(t) \in \Omega_T$ for all $t \in [0, T]$. Hence, the uniform Lipschitz condition gives $\|\dot{\boldsymbol{D}}(t)\|_F \leq L_T \|\boldsymbol{D}(t)\|_F$.

For $y(t) = \|\boldsymbol{D}(t)\|_F^2$, the Cauchy–Schwarz inequality and the uniform Lipschitz bound give $\dot{y}(t) = 2\langle \boldsymbol{D}(t), \dot{\boldsymbol{D}}(t) \rangle_F \geq -2L_T y(t)$. Applying the integrating-factor argument gives $y(t) \geq e^{-2L_T t} y(0)$, and hence

$$\|\boldsymbol{D}(t)\|_F \geq e^{-L_T t} \|\boldsymbol{D}(0)\|_F. \tag{6}$$

Finally, since $\boldsymbol{P}_{ij}$ swaps only slots i and j, $\|\boldsymbol{S}(t) - \boldsymbol{P}_{ij}\boldsymbol{S}(t)\|_F = \sqrt{2}\, \|\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)\|_2$, and the desired pairwise separation bound follows. The complete proof is provided in Appendix A.3. □

Implication of the theoretical results. Together, Theorems 1 and 2 establish guarantees on pairwise slot separation in SlotODE. Theorem 1 shows that slot-state collision does not occur over any finite update interval, while Theorem 2 provides a lower bound on pairwise slot separation. Together, these results show that, under standard regularity conditions and permutation equivariance, SlotODE prevents initially distinct slot states from becoming identical and provides a quantitative guarantee on their pairwise separation throughout the update process.

4.4 Architecture of SlotODE

We next describe the practical architecture of SlotODE. Specifically, we introduce the parameterization of the vector field F_θ and the initialization scheme used to construct the initial slot state $\boldsymbol{S}_0$. We also describe the adaptive slot-selection mechanism (Fan et al., 2024) used to determine the active slots after the update process.

Vector field parameterization. We parameterize the SlotODE vector field F_θ using a shared three-layer network f_θ with spectrally controlled linear transformations and DyT nonlinearities, followed by an element-wise $\tanh$ output. For each slot i, f_θ takes the current slot state $\boldsymbol{s}_i(t)$, aggregated feature evidence $\boldsymbol{u}_i(t)$, and time embedding $\phi(t)$ as input and predicts its time derivative. We concatenate these inputs as $\boldsymbol{z}_i(t) = [\boldsymbol{s}_i(t); \boldsymbol{u}_i(t); \phi(t)]$.

We apply spectral normalization (SN) (Miyato et al., 2018) to the linear layers to control their spectral scale and, consequently, their Lipschitz constants. SN normalizes each linear weight by its spectral norm and has been shown to improve training stability through controlled spectral scaling (Lin et al., 2021; Lee et al., 2021). However, overly restrictive spectral constraints can degrade empirical performance (Lin et al., 2021). We therefore relax the unit spectral constraint using a scaling factor ρ, with $\widetilde{W}_\ell = \rho W_\ell / \sigma(W_\ell)$.

For the nonlinear transformations within the network, we use Dynamic Tanh (DyT) (Zhu et al., 2025) instead of LayerNorm (Ba et al., 2016). Although LayerNorm with a positive stabilization constant is Lipschitz, its input-dependent normalization can yield a loose worst-case Lipschitz bound (Kim et al., 2021). DyT is an element-wise transformation defined as $\mathrm{DyT}(x) = \gamma \odot \tanh(\alpha x) + \beta$ and provides learnable scaling and shifting without relying on input normalization statistics. Finally, we apply an element-wise $\tanh$ at the output of the network to bound the predicted slot-state derivative.

Slot initialization. Slot initialization determines the initial condition $\boldsymbol{S}_0$ from which the SlotODE dynamics evolve. Recent studies have highlighted the importance of initialization in Slot Attention (Jia et al., 2022; Kori et al., 2024; Zhao et al., 2025a). In SlotODE, initialization is also relevant to our theoretical analysis, which assumes pairwise distinct initial slot states. Motivated by the input-dependent initialization (Zhao et al., 2025a), we construct $\boldsymbol{S}_0$ directly from the input features. Specifically, we use fixed orthogonal directions to compute softmax assignments over the input features $\boldsymbol{X}$, followed by weighted aggregation to encourage separated initial slot states.

Adaptive slot selection. SlotODE addresses how slot states evolve throughout the update process, but does not determine how many slots should remain active for a given scene. Since the number of objects varies across scenes, a fixed number of slots can introduce unnecessary or redundant slot representations. We combine SlotODE with adaptive slot selection from AdaSlot (Fan et al., 2024), which predicts whether each candidate slot should remain active and masks inactive slots according to the input, while SlotODE governs their update dynamics. Details of the vector field parameterization, slot initialization, and adaptive slot-selection are provided in Appendix C.

5 Experiments

5.1 Experimental Setup

Numerical integration. SlotODE defines the slot update dynamics continuously over the fixed interval $t \in [0, 1]$. In practice, we obtain the slot states by numerically integrating the learned vector field over this interval. We use the forward Euler method with $N = 3$ integration steps: $\boldsymbol{S}^{(k+1)} = \boldsymbol{S}^{(k)} + \frac{1}{N} F_\theta(\boldsymbol{S}^{(k)}, \boldsymbol{X}, t^{(k)})$. We use the same Euler integration for training and the main evaluation.

Table 1: Comparison of SlotODE and conventional Slot Attention (SA*) using the same adaptive slot-selection mechanism. MOVi-C/E and COCO use frozen pretrained ViT, while CLEVR10 is trained end-to-end from scratch. The best results are highlighted in bold, as in the following tables.

	MOVi-C		MOVi-E		COCO		CLEVR10	
Slot Update	FG-ARI	mBO_i	FG-ARI	mBO_i	FG-ARI	mBO_i	FG-ARI	mBO_i
SA*	$75.22_{\pm 0.3}$	$35.60_{\pm 0.4}$	$76.04_{\pm 0.5}$	$29.76_{\pm 0.5}$	$41.12_{\pm 0.4}$	$27.64_{\pm 0.4}$	$94.72_{\pm 0.7}$	$21.04_{\pm 0.1}$
SlotODE	$\mathbf{79.15}_{\pm 0.2}$	$\mathbf{38.62}_{\pm 0.1}$	$\mathbf{78.43}_{\pm 0.2}$	$\mathbf{31.74}_{\pm 0.2}$	$\mathbf{42.63}_{\pm 0.2}$	$\mathbf{29.62}_{\pm 0.1}$	$\mathbf{96.64}_{\pm 0.4}$	$\mathbf{22.45}_{\pm 0.3}$

Object discovery datasets and metrics. Following the evaluation settings of Fan et al. (2024), we evaluate SlotODE on the synthetic MOVi-C/E dataset (Greff et al., 2022), and the real-world MS COCO (2017) dataset (Lin et al., 2014). We additionally evaluate on synthetic CLEVR10 (Johnson et al., 2017). We report Foreground Adjusted Rand Index (FG-ARI ↑) to evaluate foreground grouping quality and object instance-wise mean Best Overlap (mBO_i ↑) to measure the overlap between predicted and ground-truth object masks.

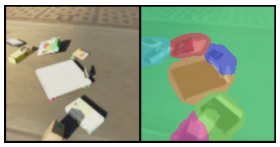
Training configuration. We adopt the adaptive slot-selection mechanism of AdaSlot (Fan et al., 2024) and follow its evaluation settings. For the complex-scene benchmarks, including MOVi-C/E, and COCO, we follow the DINOSAUR (Seitzer et al., 2022) framework. Specifically, we use a frozen pretrained Vision Transformer (ViT) (Seitzer et al., 2022) as the visual encoder (Base with patch size 16) and an MLP decoder with an MSE reconstruction objective. For CLEVR10, we train the visual encoder and slot-based model end-to-end from scratch using a convolutional neural network (CNN)-based architecture. We use 11 candidate slots for MOVi-C, COCO and CLEVR10, 24 slots for MOVi-E. We set the spectral scale to $\rho = 3.0$ for all spectrally normalized linear layers. We optimize the models using Adam with a learning rate of 4×10^{-4} and an exponential decay schedule after a 10K-step warm-up. Detailed configurations are provided in Appendix D.1.

Baselines. We conduct two comparisons. (1) We compare SlotODE with conventional Slot Attention while retaining the remaining model components to evaluate the effect of replacing conventional iterative updates with SlotODE. Both methods use the same adaptive slot-selection mechanism (Fan et al., 2024). (2) We further compare SlotODE with representative slot-based OCL methods on complex-scene benchmarks, including DINOSAUR (Seitzer et al., 2022), MetaSlot (Liu et al., 2026), and SPOT (Kakogeorgiou et al., 2024). For comparisons, SlotODE is evaluated with a pretrained DINO (Caron et al., 2021) ViT encoder and adaptive slot selection from AdaSlot. MetaSlot also addresses the fixed slot-number constraint, while SPOT leverages decoder attention masks as self-training signals for slot formation. For comparability, we use MLP-based decoders where applicable. SPOT employs self-training (ST) which distills the decoder attention masks to the Slot Attention module with path-order sequence permutation (SP), and ensembling (ENS) results of different permutation strategies. We additionally report SPOT with its autoregressive Transformer decoder, as its permutation formulation is based on autoregressive reconstruction. We report the main results over 5 runs and conduct the in-depth analyses over 3 runs. Standard deviations are reported as $\pm\,\mathrm{std}$, while a blank entry indicates that the standard deviation was not provided in the original paper.

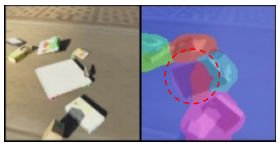
5.2 Experimental Results

Comparison of SlotODE with conventional Slot Attention. Table 1 provides our primary comparison between SlotODE and conventional Slot Attention under the same adaptive slot selection settings of AdaSlot. Replacing the conventional iterative updates with SlotODE significantly improves both FG-ARI and mBO. Importantly, these improvements are observed in both the DINOSAUR-based setting with pretrained visual features (MOVi-C/E, COCO) and the end-to-end setting trained from scratch (CLEVR10). We further verify in Appendix B.1 that the performance gains of SlotODE persist without adaptive slot selection (Table 8). Additional comparisons in Appendix B.2 show consistent gains across a broader set of evaluation metrics. Together, these results show that replacing conventional slot updates with SlotODE consistently improves object discovery performance, demonstrating the effectiveness of the proposed ODE-based slot update formulation.

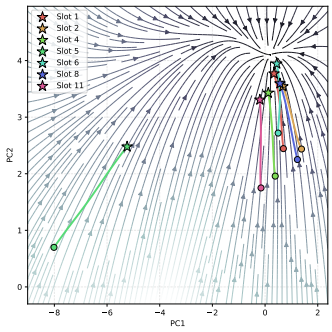
Qualitative comparison of slot-state trajectories. We provide a qualitative comparison between SlotODE and conventional Slot Attention. Both methods use the same adaptive slot-selection mechanism, and we visualize only the retained slots. Figures 1a and 1b show the slot-wise segmentation



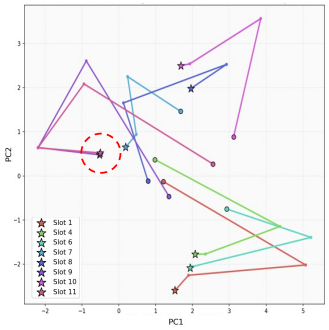
(a) SlotODE masks



(b) Slot Attention masks



(c) SlotODE trajectory



(d) Slot Attention trajectory

Figure 1: Qualitative comparison of object segmentation mask and slot update trajectories between SlotODE and Slot Attention. "Circle" and "star" markers denote the initial and final slot states. Lines show the update trajectories, and colors match slot indices to segmentation masks. Only slots retained by AdaSlot's selection are shown for both methods. We highlight overlapping slots in the projected Slot Attention state space, which correspond to fragmented segmentation masks.

Table 2: Comparison with prior OCL methods. Parentheses denote official results and "–" unavailable results. The default setting is ViT-B/16 with an MLP decoder (†: ViT-B/8; ‡: ViT-S/14). Best/second-best results are bold/underlined, and the average rank is computed across all datasets and metrics (↓). Ours* denotes SlotODE with the pretrained ViT and adaptive slot selection.

Method	COCO		MOVi-C		MOVi-E		Average
	FG-ARI	mBO$_i$	FG-ARI	mBO$_i$	FG-ARI	mBO$_i$	Rank
Dinosaur	40.18$_{\pm 0.5}$ (40.50)	27.52$_{\pm 0.4}$ (27.77)	65.73$_{\pm 0.4}$ (66.00)	34.63$_{\pm 0.2}$ (35.00)	62.21$_{\pm 0.3}$ (65.10)†	31.12$_{\pm 0.3}$ (35.50)†	4.57
AdaSlot	41.12$_{\pm 0.4}$ (39.00)	27.64$_{\pm 0.4}$ (27.36)	<u>75.22</u>$_{\pm 0.3}$ (75.59)	35.60$_{\pm 0.4}$ (35.64)	<u>76.04</u>$_{\pm 0.5}$ (76.73)	29.76$_{\pm 0.5}$ (29.83)	3.50
MetaSlot‡	39.22$_{\pm 0.9}$ (40.68$_{\pm 0.5}$)	28.91$_{\pm 0.3}$ (29.50$_{\pm 0.2}$)	72.20$_{\pm 1.8}$ (-)	38.51$_{\pm 0.3}$ (-)	72.92$_{\pm 1.6}$ (-)	29.55$_{\pm 0.2}$ (-)	4.33
SPOT$^{\text{ST}}_{\text{MLP}}$	<u>41.95</u>$_{\pm 0.8}$ (42.50)	27.01$_{\pm 0.7}$ (28.40)	66.77$_{\pm 0.6}$ (-)	35.63$_{\pm 0.3}$ (-)	62.98$_{\pm 0.6}$ (-)	30.96$_{\pm 0.2}$ (-)	3.50
SPOT$^{\text{ST,SP,ENS}}_{\text{Auto}}$	36.71$_{\pm 1.0}$ (37.80$_{\pm 0.2}$)	**34.79**$_{\pm 0.6}$ (35.00$_{\pm 0.1}$)	49.22$_{\pm 3.6}$ (52.30$_{\pm 3.3}$)	**45.85**$_{\pm 2.8}$ (47.30$_{\pm 1.2}$)	50.64$_{\pm 6.68}$ (56.80$_{\pm 4.3}$)	**40.04**$_{\pm 1.31}$ (40.10$_{\pm 1.2}$)	3.50
Ours*	**42.63**$_{\pm 0.2}$	<u>29.62</u>$_{\pm 0.1}$	**79.15**$_{\pm 0.2}$	<u>38.62</u>$_{\pm 0.1}$	**78.43**$_{\pm 0.2}$	<u>31.74</u>$_{\pm 0.2}$	**1.50**

masks, while Figures 1c and 1d visualize the evolution of slot states from initialization to the final update after projection onto a two-dimensional PCA space. For SlotODE, the arrows indicate the local direction of the learned vector field along the projected trajectories. In conventional Slot Attention, the trajectories of "Slots 9" and "Slot 11" overlap across intermediate updates in the projected space, while the corresponding segmentation masks split a single object across the two slots as highlighted in red. In contrast, SlotODE shows clearer separation between the projected slot states across updates, while the corresponding segmentation masks assign the object to a single slot rather than splitting it across multiple slots. Similar qualitative trends are observed in additional examples (Figure 2 in Appendix B.3). Together, these qualitative results provide empirical evidence that maintaining separation between slot states throughout the update process is important for preserving different slot representations corresponding to individual objects.

Comparison with existing OCL methods. Table 2 compares SlotODE with representative slot-based OCL methods. We evaluate SlotODE using a frozen pretrained ViT encoder following the DINOSAUR framework, together with adaptive slot selection. SlotODE achieves the best FG-ARI performance across all datasets. Compared with SPOT with autoregressive decoder, SlotODE achieves substantially higher FG-ARI while attaining the second-best mBO$_i$. Overall, SlotODE achieves the best average rank across datasets and metrics, demonstrating consistently strong object discovery performance across the evaluated benchmarks. Qualitative segmentation results further show that SlotODE successfully segments individual objects in the scene (Appendix B.4).

Table 3: Ablation of the SlotODE formulation on MOVi-C. "Vector field" denotes predicting the slot state derivative rather than the next state, "Time" denotes conditioning on time t, and Δt denotes step-size scaling with $\Delta t = 1/N$ under Euler integration.

Update function	Vector field	Time	Δt	FG-ARI	mBO$_i$
$\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + \frac{1}{N} F_\theta(\boldsymbol{S}^k, \boldsymbol{X}, t^k)$	O	O	O	**79.15**$_{\pm 0.2}$	**38.62**$_{\pm 0.1}$
$\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + F_\theta(\boldsymbol{S}^k, \boldsymbol{X}, t^k)$	O	O	X	78.48$_{\pm 0.2}$	38.50$_{\pm 0.1}$
$\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + \frac{1}{N} F_\theta(\boldsymbol{S}^k, \boldsymbol{X})$	O	X	O	77.30$_{\pm 0.3}$	36.85$_{\pm 0.2}$
$\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + F_\theta(\boldsymbol{S}^k, \boldsymbol{X})$	O	X	X	77.45$_{\pm 0.4}$	37.16$_{\pm 0.1}$
$\boldsymbol{S}^{k+1} = F_\theta^{\text{disc}}(\boldsymbol{S}^k, \boldsymbol{X})$	X	X	X	76.66$_{\pm 0.2}$	36.33$_{\pm 0.3}$

Table 4: Robustness to different numbers of update steps on MOVi-C.

(a) Evaluation with varying N.

Slot update	N_{train}	N_{eval}	FG-ARI	mBO$_i$
SlotODE	3	3	**79.15**$_{\pm 0.2}$	**38.62**$_{\pm 0.1}$
	3	2	76.37$_{\pm 0.6}$	37.76$_{\pm 0.1}$
	3	6	78.90$_{\pm 0.4}$	38.28$_{\pm 0.1}$
	3	12	78.91$_{\pm 0.3}$	38.49$_{\pm 0.2}$

(b) Evaluation with increasing N.

	Slot Attention		SlotODE	
N	FG-ARI	mBO$_i$	FG-ARI	mBO$_i$
3	75.22$_{\pm 0.3}$	35.60$_{\pm 0.4}$	**79.15**$_{\pm 0.2}$	**38.62**$_{\pm 0.1}$
6	73.18$_{\pm 1.0}$	33.51$_{\pm 1.4}$	77.53$_{\pm 0.3}$	38.29$_{\pm 0.4}$
12	24.80$_{\pm 5.5}$	9.87$_{\pm 1.9}$	77.71$_{\pm 0.3}$	38.06$_{\pm 0.1}$

6 In-Depth Analysis

6.1 Effect of the ODE Formulation

In this section, we analyze SlotODE in terms of (1) the impact of the ODE formulation and its components and (2) the robustness of the learned vector field across different numbers of update steps and ODE solvers.

Ablation of the ODE formulation. We first examine whether the benefit of SlotODE comes from the ODE formulation, rather than replacing the conventional recurrent slot update. Table 5 compares the ODE-based update with a direct next-state update and the conventional recurrent update. Here, F_θ^{disc} uses the same network architecture as that used to parameterize the SlotODE vector field F_θ, but predicts the next slot state. In contrast, SlotODE predicts the time derivative of the slot state and obtains the updated state through numerical integration. The ODE-based update achieves higher FG-ARI and mBO$_i$, indicating that the gain extends beyond replacing the update network and supports the benefit of the ODE formulation.

Table 5: Effect of the ODE formulation beyond replacing the conventional slot update on MOVi-C.

Formulation	FG-ARI	mBO$_i$
ODE-based update $\boldsymbol{s}^{k+1} = \boldsymbol{s}^k + \frac{1}{N} F_\theta(\boldsymbol{s}^k, \boldsymbol{X}, t^k)$	**79.15**$_{\pm 0.2}$	**38.62**$_{\pm 0.1}$
Direct next-state update $\boldsymbol{s}^{k+1} = F_\theta^{\text{disc}}(\boldsymbol{s}^k, \boldsymbol{X})$	76.66$_{\pm 0.2}$	36.33$_{\pm 0.3}$
Conventional recurrent update $\boldsymbol{s}^{k+1} = G_\theta(\boldsymbol{s}^k, \boldsymbol{X})$	75.22$_{\pm 0.4}$	35.60$_{\pm 0.4}$

Components of the ODE formulation. We ablate three components of the ODE formulation: derivative prediction, time conditioning, and step-size scaling with $\Delta t = 1/N$. Derivative prediction parameterizes the time derivative of the slot state rather than predicting the next state, while time conditioning allows the vector field to vary with time. The step-size scaling $\Delta t = 1/N$ interprets the number of update steps N as the discretization resolution over the fixed interval $t \in [0, 1]$ under Euler integration. As shown in Table 3, direct next-state prediction yields the lowest performance. Introducing derivative-based updates improves both FG-ARI and mBO$_i$, and time conditioning provides further gains. Including $\Delta t = 1/N$ further improves FG-ARI while maintaining competitive mBO$_i$. Overall, these results support parameterizing slot updates through a time-dependent vector field and obtaining the updated slot states through numerical integration.

Robustness across different numbers of update steps. We examine whether SlotODE learns an ODE vector field that remains effective under discretizations different from that used during training. As shown in Table 4a, a model trained with $N = 3$ retains strong performance under finer discretizations with $N = 6$ and $N = 12$, while performance decreases with the coarser discretization $N = 2$. This indicates that the learned vector field remains effective when integrated at finer resolutions than

Table 6: Ablation study of (a) initialization method and (b) spectral normalization.

(a) Slot initialization (MOVi-C).

Initialization	Slot update	FG-ARI	mBO_i
Our initialization	SlotODE	**79.15**$_{\pm 0.2}$	**38.62**$_{\pm 0.1}$
Random	SlotODE	76.55$_{\pm 0.4}$	36.67$_{\pm 0.1}$
Fixed orthogonal	SlotODE	77.96$_{\pm 0.1}$	37.75$_{\pm 0.2}$
Embedding	SlotODE	77.91$_{\pm 0.3}$	36.37$_{\pm 0.2}$
Our initialization	Slot Attn.	77.31$_{\pm 0.1}$	37.15$_{\pm 0.1}$

(b) Spectral normalization (SN).

Dataset	SN	FG-ARI	mBO_i
MOVi-C	O	**79.15**$_{\pm 0.2}$	**38.62**$_{\pm 0.1}$
	X	78.82$_{\pm 0.5}$	37.48$_{\pm 1.4}$
COCO	O	**42.63**$_{\pm 0.2}$	**29.62**$_{\pm 0.1}$
	X	36.43$_{\pm 6.3}$	24.49$_{\pm 5.5}$

that used during training. We then examine robustness as the number of update steps increases. With the update interval fixed at $T = 1$, increasing N corresponds to a finer discretization of the SlotODE dynamics. This comparison examines how treating N as the discretization resolution of a fixed ODE interval, rather than as recurrent depth, affects performance as N increases. As shown in Table 4b, SlotODE shows a moderate decrease in FG-ARI while maintaining similar mBO_i as N increases. In contrast, conventional Slot Attention degrades substantially at $N = 12$, showing that SlotODE is more robust to increasing numbers of update steps than conventional Slot Attention.

Robustness across different ODE solvers. We further evaluate the learned vector field using alternative ODE solvers. Euler used for training is a first-order fixed-step solver, whereas RK4 (Hairer et al., 1993) is a fourth-order fixed-step solver and Dopri5 (Dormand & Prince, 1980) is a fifth-order solver that adaptively adjusts the step size during integration. Table 7 shows that replacing Euler with RK4 or Dopri5 at inference yields comparable performance. These results show that the learned vector field remains robust across different ODE solvers, supporting its interpretation as an underlying continuous-time vector field rather than an update rule specific to Euler integration.

Table 7: Robustness of SlotODE to different ODE solvers on MOVi-C.

Train	Eval	FG-ARI↑	mBO_i↑
Euler	Euler	79.15$_{\pm 0.2}$	38.62$_{\pm 0.1}$
Euler	RK4	78.89$_{\pm 0.3}$	**39.41**$_{\pm 0.5}$
Euler	Dopri5	**79.39**$_{\pm 0.2}$	39.09$_{\pm 0.5}$

6.2 Ablation of Architectural Choices

We further evaluate the architectural choices used in our model. (1) Slot initialization: Table 6a compares our initialization with random, fixed orthogonal, and trainable embedding (Jia et al., 2022) alternatives. Our initialization, which combines fixed orthogonal directions with input-dependent feature aggregation, achieves the best performance. Notably, applying our initialization to conventional Slot Attention still yields lower performance than SlotODE, showing that the gain of SlotODE is not explained by initialization alone. (2) Spectral normalization: Table 6b evaluates spectral normalization in the SlotODE vector field parameterization. On MOVi-C, removing spectral normalization decreases average performance and increases variation across seeds. This effect is stronger on COCO, indicating that spectral normalization improves both performance and training stability in the SlotODE vector field parameterization.

7 Conclusion and Future Work

Our work studies how slot representations evolve and remain separated throughout the iterative updates of Slot Attention. We define *slot-state collision*, in which initially different slot states become identical during the update process. To address this problem, we introduce **SlotODE**, which reformulates iterative slot updates as an ordinary differential equation over slot states. Our theoretical analysis shows that SlotODE prevents slot-state collision and further derives an explicit lower bound on pairwise slot separation. In experiments, SlotODE significantly improves object discovery performance across OCL benchmarks. In-depth analyses demonstrate the impact of the ODE formulation and its components, as well as the robustness of the learned vector field across different ODE solvers and numbers of update steps.

Our analysis focuses on pairwise separation between slot states during iterative updates. An important direction for future work is to connect this state-level property to object-level behavior, such as object assignment and semantic specialization of slot representations. More broadly, formulating slot updates as an ODE opens the possibility of studying object-centric representations as evolving states within generative dynamics, rather than treating them only as conditioning signals.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks with required disclosure:

- **Used tasks with required disclosure:** Help develop theoretical models or conceptual frameworks, formulate mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods, assist with translation.
- **Unused tasks with required disclosure:** Generate synthetic data sets, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, clean and reformat dataset, support qualitative and thematic data analysis, interpret results.

Additionally, we used generative AI tools for the following tasks with recommended disclosure:

- **Used tasks with recommended disclosure:** Create or edit software code, draft parts of a research paper, summarize or analyse existing literature, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature, format references, suggest a structure for a research paper, propose a title or keywords for a research paper.
- **Unused tasks with recommended disclosure:** Formulate questions for surveys or interviews, create or modify scientific figures or images, suggest experimental parameters, creation of artifacts, transcribe recordings of research material, discover research topics or identify gaps.

We have reviewed all AI-assisted work. For the AI-assisted work described above, the authors first developed the underlying research content and used generative AI as an assisting tool for the disclosed tasks. All AI-assisted outputs were subsequently reviewed and verified by the authors before being incorporated into this work.

Mathematical claims, theoretical arguments, and proofs were independently checked by the authors for correctness and consistency with the proposed formulation. AI-assisted brainstorming, literature review, information retrieval, and manuscript preparation were reviewed by the authors to ensure that they accurately reflected the intended research direction. AI-assisted descriptions of the research methodology and experiments were manually checked by the authors against the actual research and experimental settings. AI-assisted method implementation and code were also manually reviewed, executed, and tested by the authors before being used in this work.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We ensure the reproducibility of our work as follows:

- **Method and implementation details.** We provide a detailed description of SlotODE in Section 4.2, with architectural and implementation details in Section 4.4 and Appendix C.
- **Theoretical analysis.** We present the theoretical guarantees of SlotODE in Section 4.3, together with the complete proofs and additional theoretical details in Appendix A.
- **Training configurations.** We provide the training configurations of SlotODE and details of our reproduction of the comparison baselines in Section 5.1 and Appendix D.
- **Code.** We include the code for training SlotODE in the supplementary material, together with a README file providing detailed instructions for reproduction and usage.

ETHICS STATEMENT

This work aims to advance object-centric representation learning through a methodological contribution to iterative slot refinement. Our experiments use publicly available benchmark datasets and

do not involve human-subject experiments or the collection of new personal data. We do not anticipate specific negative societal consequences beyond those generally associated with deep learning research, such as potential biases inherited from training datasets and the environmental cost of model training. We hope that improving and better understanding the fundamental representation learning mechanisms underlying object-centric representations can contribute to the development of more reliable and interpretable representation learning systems.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Melika Behjati and James Henderson. Inducing meaningful units from character sequences with dynamic capacity slot attention. *arXiv preprint arXiv:2102.01223*, 2021.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International conference on machine learning*, pp. 573–582. PMLR, 2019.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Michael Chang, Tom Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. *Advances in Neural Information Processing Systems*, 35:32694–32708, 2022.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, eighth workshop on syntax, semantics and structure in statistical translation*, pp. 103–111, 2014.

Aniket Didolkar, Anirudh Goyal, and Yoshua Bengio. Cycle consistency driven object discovery. *arXiv preprint arXiv:2306.02204*, 2023.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

Ke Fan, Zechen Bai, Tianjun Xiao, Tong He, Max Horn, Yanwei Fu, Francesco Locatello, and Zheng Zhang. Adaptive slot attention: Object discovery with dynamic slot number. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23062–23071. IEEE, 2024.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pp. 3154–3164. PMLR, 2020.

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

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34(1):014004, 2018.

Ernst Hairer, Gerhard Wanner, and Syvert P Nørsett. *Solving ordinary differential equations I: Nonstiff problems*. Springer, 1993.

Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. *arXiv preprint arXiv:2210.08990*, 2022.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 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.

Ioannis Kakogeorgiou, Spyros Gidaris, Konstantinos Karantzalos, and Nikos Komodakis. Spot: Self-training with patch-order permutation for object-centric learning with autoregressive transformers. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22776–22786. IEEE, 2024.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571. PMLR, 2021.

Jinwoo Kim, Janghyuk Choi, Ho-Jin Choi, and Seon Joo Kim. Shepherding slots to objects: Towards stable and robust object-centric learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19198–19207. IEEE, 2023.

Daniil Kirilenko, Vitaliy Vorobyov, Aleksey Kovalev, and Aleksandr Panov. Object-centric learning with slot mixture module. In *International Conference on Learning Representations*, volume 2024, pp. 51784–51802, 2024.

Avinash Kori, Francesco Locatello, Fabio De Sousa Ribeiro, Francesca Toni, and Ben Glocker. Grounded object-centric learning. In *International Conference on Learning Representations*, volume 2024, pp. 42559–42603, 2024.

Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu. Vitgan: Training gans with vision transformers. *arXiv preprint arXiv:2107.04589*, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Zinan Lin, Vyas Sekar, and Giulia Fanti. Why spectral normalization stabilizes gans: Analysis and improvements. *Advances in neural information processing systems*, 34:9625–9638, 2021.

Hongjia Liu, Rongzhen Zhao, Haohan Chen, and Joni Pajarinen. Metaslot: Break through the fixed number of slots in object-centric learning. *Advances in Neural Information Processing Systems*, 38:67319–67344, 2026.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International conference on machine learning*, pp. 3276–3285. PMLR, 2018.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *Advances in neural information processing systems*, 33:3952–3963, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

WonJun Moon, Hyun Seok Seong, and Jae-Pil Heo. Reconstruction-guided slot curriculum: Addressing object over-fragmentation in video object-centric learning. *arXiv preprint arXiv:2603.22758*, 2026.

Tianran Ouyang, Xingping Dong, Jing Zhang, Mang Ye, Jun Chen, and Bo Du. Qasa: Quality-guided k-adaptive slot attention for unsupervised object-centric learning. *arXiv preprint arXiv:2601.12936*, 2026.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. *arXiv preprint arXiv:2209.14860*, 2022.

Hyun Seok Seong, WonJun Moon, and Jae-Pil Heo. From vicious to virtuous cycles: Synergistic representation learning for unsupervised video object-centric learning. *arXiv preprint arXiv:2602.03390*, 2026.

Hanshu Yan, Jiawei Du, Vincent YF Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. *arXiv preprint arXiv:1910.05513*, 2019.

Han Zhang, Xi Gao, Jacob Unterman, and Tom Arodz. Approximation capabilities of neural odes and invertible residual networks. In *International Conference on Machine Learning*, pp. 11086–11095. PMLR, 2020.

Yan Zhang, David W Zhang, Simon Lacoste-Julien, Gertjan J Burghouts, and Cees GM Snoek. Unlocking slot attention by changing optimal transport costs. In *International Conference on Machine Learning*, pp. 41931–41951. PMLR, 2023.

Rongzhen Zhao, Wenyan Yang, Juho Kannala, and Joni Pajarinen. Smoothing slot attention iterations and recurrences. *arXiv preprint arXiv:2508.05417*, 2025a.

Rongzhen Zhao, Yi Zhao, Juho Kannala, and Joni Pajarinen. Slot attention with re-initialization and self-distillation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 4185–4192, 2025b.

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14901–14911. IEEE, 2025.

A THEORETICAL ANALYSIS OF SLOTODE

In this section, we provide the theoretical analysis underlying the pairwise separation guarantees of SlotODE. We first define the SlotODE dynamics and the assumptions used throughout the analysis, and then establish finite-time exclusion of slot-state collision together with the lower bound on pairwise slot separation presented in the main text.

A.1 PROBLEM SETUP AND ASSUMPTIONS

Let

$$\boldsymbol{s}_i(t) \in \mathbb{R}^D, \qquad i = 1, \dots, K, \tag{7}$$

denote the state of the i-th slot at time t. We collect the K slot states into the joint slot state

$$\boldsymbol{S}(t) = \begin{bmatrix} \boldsymbol{s}_1(t)^\top \\ \vdots \\ \boldsymbol{s}_K(t)^\top \end{bmatrix} \in \mathbb{R}^{K \times D}. \tag{8}$$

Given fixed input features $\boldsymbol{X}$, SlotODE defines the joint slot state through the following ordinary differential equation:

$$\frac{d\boldsymbol{S}(t)}{dt} = F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t), \qquad \boldsymbol{S}(0) = S_0. \tag{9}$$

Here, $\boldsymbol{X}$ is treated as a fixed parameter throughout the analysis, and F_θ denotes the SlotODE vector field induced by the composition of state-dependent attention-based feature aggregation and the shared vector field parameterization.

We consider the following conditions on the SlotODE vector field.

Assumption 1 (Local Lipschitz continuity in the slot state)**.** *For every $(\boldsymbol{S}^*, t^*)$, there exist a neighborhood $\mathcal{U}$ of $\boldsymbol{S}^*$, an interval $\mathcal{I}$ containing t^*, and a finite constant $L_{\mathcal{U},\mathcal{I}}$ such that*

$$\left\| F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) - F_\theta(\widetilde{\boldsymbol{S}}, \boldsymbol{X}, t) \right\|_F \le L_{\mathcal{U},\mathcal{I}} \left\| \boldsymbol{S} - \widetilde{\boldsymbol{S}} \right\|_F \tag{10}$$

for all $\boldsymbol{S}, \widetilde{\boldsymbol{S}} \in \mathcal{U}$ and $t \in \mathcal{I}$.

Assumption 2 (Continuity in time)**.** *For every fixed $\boldsymbol{S}$, the map*

$$t \mapsto F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) \tag{11}$$

is continuous.

Assumption 3 (Slot permutation equivariance)**.** *For every permutation matrix $\boldsymbol{P} \in \mathbb{R}^{K \times K}$,*

$$F_\theta(\boldsymbol{P}\boldsymbol{S}, \boldsymbol{X}, t) = \boldsymbol{P} F_\theta(\boldsymbol{S}, \boldsymbol{X}, t). \tag{12}$$

Assumption 4 (Pairwise distinct initialization)**.** *The initial slot states satisfy*

$$\boldsymbol{s}_i(0) \neq \boldsymbol{s}_j(0), \qquad \forall i \neq j. \tag{13}$$

Under Assumptions 1 and 2, the initial value problem in Eq. 9 admits a unique local solution. Throughout the following analysis, we consider a finite update interval $[0, T]$ contained in its interval of existence.

A.2 FINITE-TIME SLOT-STATE COLLISION EXCLUSION

We first show that initially distinct slot states do not become identical over any finite interval contained in the solution's interval of existence.

Restatement (Theorem 1)**.** *Let $\boldsymbol{S}(t)$ be a solution of Eq. equation 9 on a finite interval $[0, T]$, and suppose that Assumptions 1–3 hold. Then, for every pair $i \neq j$,*

$$\boldsymbol{s}_i(0) \neq \boldsymbol{s}_j(0) \quad \Longrightarrow \quad \boldsymbol{s}_i(t) \neq \boldsymbol{s}_j(t), \qquad \forall t \in [0, T]. \tag{14}$$

Under Assumption 4, the premise of Theorem A.2 holds for every pair of slots. Hence, no slot-state collision occurs over $[0, T]$.

Proof. Fix a pair $i \neq j$, and let $\boldsymbol{P}_{ij}$ denote the permutation matrix that swaps slots i and j. Define the permuted trajectory $\widetilde{\boldsymbol{S}}(t) = \boldsymbol{P}_{ij}\boldsymbol{S}(t)$. Since $\boldsymbol{P}_{ij}$ is constant in time,

$$\frac{d\widetilde{\boldsymbol{S}}(t)}{dt} = \boldsymbol{P}_{ij}\frac{d\boldsymbol{S}(t)}{dt} = \boldsymbol{P}_{ij}F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t). \tag{15}$$

By Assumption 3,

$$\boldsymbol{P}_{ij}F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t) = F_\theta(\boldsymbol{P}_{ij}\boldsymbol{S}(t), \boldsymbol{X}, t) = F_\theta(\widetilde{\boldsymbol{S}}(t), \boldsymbol{X}, t). \tag{16}$$

Hence, $\widetilde{\boldsymbol{S}}(t)$ is also a solution of the same ODE on $[0, T]$, with initial state $\widetilde{\boldsymbol{S}}(0) = \boldsymbol{P}_{ij}\boldsymbol{S}(0)$.

Suppose, for contradiction, that there exists $t^* \in (0, T]$ such that $\boldsymbol{s}_i(t^*) = \boldsymbol{s}_j(t^*)$. Swapping slots i and j therefore leaves the joint state unchanged at t^*, so $\boldsymbol{P}_{ij}\boldsymbol{S}(t^*) = \boldsymbol{S}(t^*)$, and hence $\widetilde{\boldsymbol{S}}(t^*) = \boldsymbol{S}(t^*)$.

By Assumptions 1 and 2, the initial value problem is locally unique by the Picard–Lindelöf theorem. Since $\boldsymbol{S}$ and $\widetilde{\boldsymbol{S}}$ pass through the same state at t^*, local uniqueness implies that they coincide in a neighborhood of t^*. By uniqueness and continuity, this equality extends throughout their common interval of existence. Hence,

$$\widetilde{\boldsymbol{S}}(t) = \boldsymbol{S}(t), \qquad \forall t \in [0, T]. \tag{17}$$

In particular, $\boldsymbol{P}_{ij}\boldsymbol{S}(0) = \boldsymbol{S}(0)$, which implies $\boldsymbol{s}_i(0) = \boldsymbol{s}_j(0)$. This contradicts the premise $\boldsymbol{s}_i(0) \neq \boldsymbol{s}_j(0)$. Therefore, no such t^* exists, and

$$\boldsymbol{s}_i(t) \neq \boldsymbol{s}_j(t), \qquad \forall t \in [0, T]. \tag{18}$$

□

A.3 Exponential Lower Bound on Pairwise Slot Separation

Theorem A.2 shows that initially distinct slot states do not become identical over any finite interval. Under an additional finite-horizon uniform Lipschitz condition, we further derive an exponential lower bound on their pairwise separation over $[0, T]$.

Assumption 5 (Finite-horizon uniform Lipschitz bound)**.** *For the fixed input $\boldsymbol{X}$, there exist a permutation-invariant region $\Omega_T \subset \mathbb{R}^{K\times D}$ containing $\boldsymbol{S}(t)$ for all $t \in [0, T]$, meaning that $\boldsymbol{P}\Omega_T = \Omega_T$ for every slot permutation matrix $\boldsymbol{P}$, and a finite constant $0 \leq L_T < \infty$ such that*

$$\left\| F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) - F_\theta(\widetilde{\boldsymbol{S}}, \boldsymbol{X}, t) \right\|_F \leq L_T \left\| \boldsymbol{S} - \widetilde{\boldsymbol{S}} \right\|_F \tag{19}$$

for all $\boldsymbol{S}, \widetilde{\boldsymbol{S}} \in \Omega_T$ and all $t \in [0, T]$.

Restatement (Theorem 2)**.** *Let $\boldsymbol{S}(t)$ be a solution of Eq. 9 on $[0, T]$, and suppose that Assumptions 3 and 5 hold. Then, for every pair $i \neq j$,*

$$\|\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)\|_2 \geq e^{-L_T t} \|\boldsymbol{s}_i(0) - \boldsymbol{s}_j(0)\|_2, \qquad \forall t \in [0, T]. \tag{20}$$

Under Assumption 4, the right-hand side is strictly positive for every $i \neq j$. Thus, Theorem A.3 extends the collision-exclusion result of Theorem A.2 by providing a quantitative finite-time lower bound on pairwise slot separation.

Proof. Fix a pair $i \neq j$, and let $\boldsymbol{P}_{ij}$ denote the permutation matrix that swaps slots i and j. As in the proof of Theorem A.2, define the permuted trajectory $\widetilde{\boldsymbol{S}}(t) = \boldsymbol{P}_{ij}\boldsymbol{S}(t)$. By Assumption 3, $\widetilde{\boldsymbol{S}}(t)$ is also a solution of the same ODE:

$$\frac{d\widetilde{\boldsymbol{S}}(t)}{dt} = F_\theta(\widetilde{\boldsymbol{S}}(t), \boldsymbol{X}, t). \tag{21}$$

Since Ω_T is permutation-invariant and $\boldsymbol{S}(t) \in \Omega_T$, we also have $\widetilde{\boldsymbol{S}}(t) \in \Omega_T$ for all $t \in [0, T]$.

Define $\boldsymbol{D}(t) = \boldsymbol{S}(t) - \widetilde{\boldsymbol{S}}(t)$. Then

$$\frac{d\boldsymbol{D}(t)}{dt} = F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t) - F_\theta(\widetilde{\boldsymbol{S}}(t), \boldsymbol{X}, t). \tag{22}$$

By Assumption 5,

$$\left\| \frac{d\boldsymbol{D}(t)}{dt} \right\|_F \leq L_T \|\boldsymbol{D}(t)\|_F. \tag{23}$$

We now study the squared Frobenius norm of $\boldsymbol{D}(t)$. Since $\|\boldsymbol{D}(t)\|_F^2 = \langle \boldsymbol{D}(t), \boldsymbol{D}(t) \rangle_F$, differentiating with respect to t gives

$$\frac{d}{dt}\|\boldsymbol{D}(t)\|_F^2 = 2 \left\langle \boldsymbol{D}(t), \frac{d\boldsymbol{D}(t)}{dt} \right\rangle_F. \tag{24}$$

By the Cauchy–Schwarz inequality for the Frobenius inner product,

$$\left| \left\langle \boldsymbol{D}(t), \frac{d\boldsymbol{D}(t)}{dt} \right\rangle_F \right| \leq \|\boldsymbol{D}(t)\|_F \left\| \frac{d\boldsymbol{D}(t)}{dt} \right\|_F. \tag{25}$$

Therefore,

$$\frac{d}{dt}\|\boldsymbol{D}(t)\|_F^2 \geq -2\|\boldsymbol{D}(t)\|_F \left\| \frac{d\boldsymbol{D}(t)}{dt} \right\|_F \tag{26}$$

$$\geq -2L_T \|\boldsymbol{D}(t)\|_F^2. \tag{27}$$

Let $y(t) = \|\boldsymbol{D}(t)\|_F^2$. Then $y'(t) + 2L_T y(t) \geq 0$. Following the standard integrating-factor argument underlying Grönwall's inequality, we multiply both sides by the positive factor $e^{2L_T t}$:

$$e^{2L_T t} y'(t) + 2L_T e^{2L_T t} y(t) \geq 0. \tag{28}$$

By the product rule,

$$\frac{d}{dt}\left(e^{2L_T t} y(t)\right) = e^{2L_T t} y'(t) + 2L_T e^{2L_T t} y(t), \tag{29}$$

and hence

$$\frac{d}{dt}\left(e^{2L_T t} y(t)\right) \geq 0. \tag{30}$$

Integrating from 0 to t and applying the fundamental theorem of calculus gives $e^{2L_T t} y(t) - y(0) \geq 0$. Therefore, $y(t) \geq e^{-2L_T t} y(0)$, or equivalently,

$$\|\boldsymbol{D}(t)\|_F \geq e^{-L_T t} \|\boldsymbol{D}(0)\|_F. \tag{31}$$

It remains to relate the joint-state difference $\boldsymbol{D}(t)$ to the pairwise slot difference. Since $\boldsymbol{P}_{ij}$ only swaps rows i and j, the matrix $\boldsymbol{S}(t) - \boldsymbol{P}_{ij}\boldsymbol{S}(t)$ has nonzero rows only at positions i and j, given by $\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)$ and $\boldsymbol{s}_j(t) - \boldsymbol{s}_i(t)$, respectively. Since the squared Frobenius norm of a matrix is the sum of the squared Euclidean norms of its rows,

$$\|\boldsymbol{S}(t) - \boldsymbol{P}_{ij}\boldsymbol{S}(t)\|_F^2 = \|\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)\|_2^2 + \|\boldsymbol{s}_j(t) - \boldsymbol{s}_i(t)\|_2^2 \tag{32}$$

$$= 2\|\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)\|_2^2. \tag{33}$$

Hence, $\|\boldsymbol{D}(t)\|_F = \sqrt{2}\|\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)\|_2$, and similarly, $\|\boldsymbol{D}(0)\|_F = \sqrt{2}\|\boldsymbol{s}_i(0) - \boldsymbol{s}_j(0)\|_2$.

Substituting these identities into Eq. equation 31 and cancelling $\sqrt{2}$ yields

$$\|\boldsymbol{s}_i(t) - \boldsymbol{s}_j(t)\|_2 \geq e^{-L_T t} \|\boldsymbol{s}_i(0) - \boldsymbol{s}_j(0)\|_2, \qquad \forall t \in [0, T]. \tag{34}$$

□

A.4 Regularity of the SlotODE Vector Field

We next relate the assumptions used in the preceding analysis to the actual parameterization of SlotODE. For fixed input features $\boldsymbol{X}$, the SlotODE vector field can be written as

$$F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) = f_\theta\big(\boldsymbol{S}, \mathcal{A}(\boldsymbol{S}, \boldsymbol{X}), \phi(t)\big), \tag{35}$$

where $\mathcal{A}$ denotes the normalized attention followed by feature aggregation, $\phi(t)$ is the time embedding, and f_θ is the shared SlotODE vector field parameterization. Importantly, the regularity conditions in our analysis apply to the full vector field F_θ, rather than to the vector field parameterization f_θ alone.

Local Lipschitz continuity. We verify the local Lipschitz regularity of the SlotODE vector field with respect to the joint slot state.

Consider an arbitrary point $(\boldsymbol{S}^*, t^*)$. Let $\mathcal{U}$ be a sufficiently small convex neighborhood of $\boldsymbol{S}^*$ with compact closure, and let $\mathcal{I}$ be a sufficiently small interval containing t^* with compact closure. The state-dependent computation of F_θ is composed of differentiable maps, as described below.

For a linear projection $h(\boldsymbol{S}) = \boldsymbol{S}\boldsymbol{W}$, we have

$$\left\| h(\boldsymbol{S}) - h(\widetilde{\boldsymbol{S}}) \right\|_F \leq \|\boldsymbol{W}\|_2 \left\| \boldsymbol{S} - \widetilde{\boldsymbol{S}} \right\|_F, \tag{36}$$

where $\|\boldsymbol{W}\|_2 < \infty$ for finite network parameters.

Layer normalization with a positive numerical constant ϵ has the form

$$\mathrm{LN}(\boldsymbol{x}) = \gamma \odot \frac{\boldsymbol{x} - \mu(\boldsymbol{x})}{\sqrt{\sigma^2(\boldsymbol{x}) + \epsilon}} + \beta. \tag{37}$$

Since

$$\sigma^2(\boldsymbol{x}) + \epsilon \geq \epsilon > 0, \tag{38}$$

the denominator never vanishes. Hence, the LayerNorm map is continuously differentiable, and its Jacobian has finite operator norm on every compact neighborhood $\mathcal{V}$:

$$\sup_{x \in \mathcal{V}} \|\boldsymbol{J}_{\mathrm{LN}}(\boldsymbol{x})\|_2 < \infty. \tag{39}$$

For fixed $\boldsymbol{X}$, the key and value representations are also fixed with respect to the slot state $\boldsymbol{S}$. The attention logits depend on $\boldsymbol{S}$ through the slot queries, and all linear projections and dot-product operations are therefore locally Lipschitz in $\boldsymbol{S}$.

The softmax map is continuously differentiable, with Jacobian

$$\boldsymbol{J}_{\mathrm{sm}}(\boldsymbol{z}) = \mathrm{diag}(\boldsymbol{p}) - \boldsymbol{p}\boldsymbol{p}^\top, \qquad \boldsymbol{p} = \mathrm{softmax}(\boldsymbol{z}), \tag{40}$$

and hence has finite operator norm.

The subsequent attention re-normalization can be written, for an attention vector $\boldsymbol{a} \in \mathbb{R}^M$, as

$$R_\epsilon(\boldsymbol{a}) = \frac{\boldsymbol{a} + \epsilon \mathbf{1}}{\mathbf{1}^\top(\boldsymbol{a} + \epsilon \mathbf{1})}. \tag{41}$$

Since $\boldsymbol{a}$ is the output of a softmax map, its entries are nonnegative. Therefore, its denominator satisfies

$$\mathbf{1}^\top(\boldsymbol{a} + \epsilon \mathbf{1}) \geq M\epsilon > 0. \tag{42}$$

Thus, the re-normalization map has no singularity, and its Jacobian is continuous and bounded on compact local neighborhoods.

Let $\widehat{\boldsymbol{A}}(\boldsymbol{S}, \boldsymbol{X})$ denote the attention weights after this re-normalization, and let $\boldsymbol{V}(\boldsymbol{X})$ denote the fixed value representations. The aggregated feature evidence is

$$\boldsymbol{U}(\boldsymbol{S}, \boldsymbol{X}) = \widehat{\boldsymbol{A}}(\boldsymbol{S}, \boldsymbol{X})\boldsymbol{V}(\boldsymbol{X}). \tag{43}$$

Since $\boldsymbol{V}(\boldsymbol{X})$ is fixed with respect to $\boldsymbol{S}$,

$$\left\| \boldsymbol{U}(\boldsymbol{S}, \boldsymbol{X}) - \boldsymbol{U}(\widetilde{\boldsymbol{S}}, \boldsymbol{X}) \right\|_F \leq \|\boldsymbol{V}(\boldsymbol{X})\|_2 \left\| \widehat{\boldsymbol{A}}(\boldsymbol{S}, \boldsymbol{X}) - \widehat{\boldsymbol{A}}(\widetilde{\boldsymbol{S}}, \boldsymbol{X}) \right\|_F. \tag{44}$$

Thus, local Lipschitz continuity of the attention map is preserved under feature aggregation.

Finally, the shared vector field parameterization is composed of three linear transformations with scaled spectral normalization, two constrained DyT activations, and a final element-wise $\tanh$. For local Lipschitz continuity, it is sufficient that each linear transformation has finite operator norm. Scaled spectral normalization additionally controls the operator scale of these transformations, although an exact numerical bound on the SlotODE vector field is not required for the present argument. Under exact scaled spectral normalization,

$$\left\| \widetilde{\boldsymbol{W}}_\ell \right\|_2 \leq \rho, \tag{45}$$

where $\rho > 0$ is the prescribed spectral scale.

For the DyT activation, our implementation explicitly bounds its element-wise derivative. For each feature dimension d, we use

$$\mathrm{DyT}_d(x_d) = \widehat{\gamma}_d \tanh(\alpha_d x_d) + \beta_d, \tag{46}$$

where

$$\widehat{\gamma}_d = \frac{\gamma_d}{\max\left(|\alpha_d \gamma_d|, 1\right)}. \tag{47}$$

For fixed network parameters, its derivative with respect to its input is

$$\frac{\partial \,\mathrm{DyT}_d(x_d)}{\partial x_d} = \widehat{\gamma}_d \alpha_d \,\mathrm{sech}^2(\alpha_d x_d). \tag{48}$$

Since $\mathrm{sech}^2(\cdot) \le 1$,

$$\left|\frac{\partial \,\mathrm{DyT}_d(x_d)}{\partial x_d}\right| \le |\widehat{\gamma}_d \alpha_d| \tag{49}$$

$$= \frac{|\alpha_d \gamma_d|}{\max\left(|\alpha_d \gamma_d|, 1\right)} \le 1. \tag{50}$$

Hence, the constrained DyT activation is 1-Lipschitz with respect to its input. The final element-wise $\tanh$ is also 1-Lipschitz since $|\tanh'(x)| \le 1$.

Therefore, the Jacobian of the vector field parameterization with respect to its direct input has finite operator norm. Together with the continuously differentiable state-dependent attention and aggregation maps, the chain rule implies that $F_\theta(\boldsymbol{S}, \boldsymbol{X}, t)$ is continuously differentiable with respect to $\boldsymbol{S}$ in a neighborhood of $(\boldsymbol{S}^*, t^*)$.

Moreover, since the time embedding $\phi(t)$ is continuous in t and the derivatives of the state-dependent components depend continuously on their inputs, the state Jacobian

$$(\boldsymbol{S}, t) \mapsto \boldsymbol{J}_S F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) \tag{51}$$

is continuous on $\overline{\mathcal{U}} \times \overline{\mathcal{I}}$.

Since $\overline{\mathcal{U}} \times \overline{\mathcal{I}}$ is compact, continuity of the state Jacobian implies

$$L_{\mathcal{U},\mathcal{I}} := \sup_{\substack{\boldsymbol{S} \in \overline{\mathcal{U}} \\ t \in \overline{\mathcal{I}}}} \left\| \boldsymbol{J}_S F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) \right\|_{\mathrm{op}} < \infty, \tag{52}$$

where $\|\cdot\|_{\mathrm{op}}$ denotes the operator norm induced by the Frobenius norm on the joint slot-state space. Therefore, by the mean-value inequality, for all $\boldsymbol{S}, \widetilde{\boldsymbol{S}} \in \mathcal{U}$ and $t \in \mathcal{I}$,

$$\left\| F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) - F_\theta(\widetilde{\boldsymbol{S}}, \boldsymbol{X}, t) \right\|_F \le L_{\mathcal{U},\mathcal{I}} \left\| \boldsymbol{S} - \widetilde{\boldsymbol{S}} \right\|_F. \tag{53}$$

Hence, F_θ is locally Lipschitz in the joint slot state, locally uniformly over time, as required by Assumption 1.

Finite-horizon uniform Lipschitz bound. For fixed input features $\boldsymbol{X}$, continuity of the state Jacobian allows the local Lipschitz behavior established above to be extended to a uniform bound over the finite region Ω_T considered in Theorem A.3.

Since $\boldsymbol{S}(t)$ is continuous on the compact interval $[0, T]$, its trajectory is compact. For each fixed permutation matrix $\boldsymbol{P}$, the map $t \mapsto \boldsymbol{P}S(t)$ is continuous, so its image is also compact. Since the set $\mathcal{P}_K$ of slot permutations is finite, the union

$$\mathcal{K}_T = \{\boldsymbol{P}S(t) : t \in [0, T],\ \boldsymbol{P} \in \mathcal{P}_K\} \tag{54}$$

is also compact, where $\mathcal{K}$ denotes the set of $K \times K$ permutation matrices.

Let

$$\Omega_T = \mathrm{conv}(\mathcal{K}_T) \tag{55}$$

be its convex hull. Since $\mathcal{K}_T$ is compact in the finite-dimensional joint slot-state space, Ω_T is compact. Moreover, because $\mathcal{K}_T$ is invariant under slot permutations, Ω_T is also permutation-invariant.

Since the preceding regularity argument was established around an arbitrary state and all constituent maps remain well-defined and continuously differentiable for $\boldsymbol{S} \in \Omega_T$ and $t \in [0, T]$, the state Jacobian

$$(\boldsymbol{S}, t) \mapsto \boldsymbol{J}_{\boldsymbol{S}} F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) \tag{56}$$

is continuous on $\Omega_T \times [0, T]$. Since this set is compact,

$$L_T := \sup_{\substack{\boldsymbol{S} \in \Omega_T \\ t \in [0,T]}} \|\boldsymbol{J}_{\boldsymbol{S}} F_\theta(\boldsymbol{S}, \boldsymbol{X}, t)\|_{\mathrm{op}} < \infty. \tag{57}$$

Because Ω_T is convex, the mean-value inequality therefore gives

$$\left\| F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) - F_\theta(\widetilde{\boldsymbol{S}}, \boldsymbol{X}, t) \right\|_F \leq L_T \left\| \boldsymbol{S} - \widetilde{\boldsymbol{S}} \right\|_F \tag{58}$$

for all $\boldsymbol{S}, \widetilde{\boldsymbol{S}} \in \Omega_T$ and $t \in [0, T]$. Thus, Ω_T and L_T satisfy the finite-horizon uniform Lipschitz condition in Assumption 5.

Continuity in time. The dependence of the SlotODE vector field on time is introduced through the time embedding $\phi(t)$. For fixed $\boldsymbol{S}$ and $\boldsymbol{X}$, the attention-based feature aggregation $\mathcal{A}(\boldsymbol{S}, \boldsymbol{X})$ is independent of t. Since $\phi(t)$ is continuous in t and the SlotODE vector field parameterization is continuous with respect to its inputs, the map

$$t \mapsto F_\theta(\boldsymbol{S}, \boldsymbol{X}, t) \tag{59}$$

is continuous. Thus, the SlotODE vector field satisfies Assumption 2.

Permutation equivariance. The SlotODE vector field is equivariant to permutations of the slot dimension. Let $\boldsymbol{P}$ be any slot permutation matrix. Since the same projection functions are shared across slots, permuting the slot states correspondingly permutes the slot-query and attention-logit dimensions. The softmax over slots commutes with this permutation, and the subsequent feature-wise re-normalization is applied identically to each slot. Hence, the normalized attention weights satisfy

$$\widehat{\boldsymbol{A}}(\boldsymbol{P}\boldsymbol{S}, \boldsymbol{X}) = \boldsymbol{P}\widehat{\boldsymbol{A}}(\boldsymbol{S}, \boldsymbol{X}). \tag{60}$$

Since the value representations $\boldsymbol{V}(\boldsymbol{X})$ are independent of the slot ordering, feature aggregation gives

$$\mathcal{A}(\boldsymbol{P}S, \boldsymbol{X}) = \boldsymbol{P}\mathcal{A}(\boldsymbol{S}, \boldsymbol{X}). \tag{61}$$

The parameterization of SlotODE vector field f_θ is applied independently to each slot with shared parameters, and the same time embedding $\phi(t)$ is provided to every slot. Therefore,

$$F_\theta(\boldsymbol{P}\boldsymbol{S}, \boldsymbol{X}, t) = f_\theta\big(\boldsymbol{P}\boldsymbol{S}, \boldsymbol{P}\mathcal{A}(\boldsymbol{S}, \boldsymbol{X}), \phi(t)\big) \tag{62}$$

$$= \boldsymbol{P} f_\theta\big(\boldsymbol{S}, \mathcal{A}(\boldsymbol{S}, \boldsymbol{X}), \phi(t)\big) \tag{63}$$

$$= \boldsymbol{P} F_\theta(\boldsymbol{S}, \boldsymbol{X}, t). \tag{64}$$

Thus, the SlotODE vector field satisfies Assumption 3.

Finite-horizon existence. In addition to local existence guaranteed by the regularity conditions above, the SlotODE parameterization prevents finite-time divergence of the slot state. In particular, the final element-wise $\tanh$ in the parameterization vector field bounds every coordinate of the vector field as

$$|[F_\theta(\boldsymbol{S}, \boldsymbol{X}, t)]_{k,d}| \leq 1. \tag{65}$$

Therefore,

$$\|F_\theta(\boldsymbol{S}, \boldsymbol{X}, t)\|_F \leq \sqrt{KD}. \tag{66}$$

Along any solution trajectory,

$$\|\boldsymbol{S}(t)\|_F \leq \|\boldsymbol{S}(0)\|_F + \int_0^t \|F_\theta(\boldsymbol{S}(\tau), \boldsymbol{X}, \tau)\|_F \, d\tau \tag{67}$$

$$\leq \|\boldsymbol{S}(0)\|_F + t\sqrt{KD}. \tag{68}$$

Hence, the slot state cannot diverge to infinity at any finite time. Together with continuity in t, local Lipschitz continuity in $\boldsymbol{S}$, and the fact that the SlotODE vector field is well-defined over the entire joint slot-state space, this boundedness excludes finite-time blow-up. Standard continuation results for ordinary differential equations therefore allow the unique local solution to be extended over any finite interval $[0, T]$ considered in our analysis.

A.5 Connections to Neural ODE Theory

Continuous-time interpretations of deep neural networks have been developed by relating residual architectures to numerical discretizations of differential equations (Haber & Ruthotto, 2018; Lu et al., 2018). Neural ordinary differential equations (Neural ODEs) (Chen et al., 2018) make this connection explicit by parameterizing the derivative of the hidden state with a neural network and defining representation evolution through continuous-time integration.

Subsequent work has investigated the structure and architectural design of continuous neural dynamics (Massaroli et al., 2020), as well as their expressivity and approximation capabilities (Dupont et al., 2019; Zhang et al., 2020). Under standard conditions ensuring uniqueness of ODE solutions, distinct trajectories cannot intersect at a common finite time, yielding injective solution maps over their interval of existence. This non-intersection property has been studied in Neural ODEs from the perspective of robustness (Yan et al., 2019), while the resulting structural restrictions have also been examined as an expressivity limitation (Dupont et al., 2019) and in connection with invertibility and approximation (Zhang et al., 2020).

Injectivity is not exclusive to continuous-time models. Related work on invertible residual networks shows that discrete residual mappings can also be made invertible under suitable Lipschitz constraints (Behrmann et al., 2019). Regularization of the vector field and its Jacobian has also been studied for controlling the stability and complexity of Neural ODE dynamics (Finlay et al., 2020).

Our analysis uses the uniqueness property of ODE solutions from a different perspective. Rather than studying the expressivity or invertibility of Neural ODEs themselves, we combine solution uniqueness with permutation-equivariant slot dynamics to show that initially distinct slot states do not become identical over any finite interval.

A.6 Scope of the Theoretical Guarantees

Slot-state guarantees versus object-level behavior. The theoretical results above concern slot-state collision, where two initially distinct slot states become identical during iterative updates. Under the stated assumptions, Theorem A.2 excludes such finite-time collisions, and Theorem A.3 further provides an explicit lower bound on pairwise slot separation. These guarantees concern the slot states themselves and do not imply that different slots necessarily represent different semantic objects, produce distinct attention maps, or avoid other forms of representational redundancy.

ODE dynamics versus numerical integration. Theorems A.2 and A.3 are properties of the SlotODE dynamics defined by

$$\frac{d\boldsymbol{S}(t)}{dt} = F_\theta(\boldsymbol{S}(t), \boldsymbol{X}, t). \tag{69}$$

In practice, the ODE solution is approximated using a numerical solver. The theoretical guarantees therefore apply to the exact ODE dynamics and are not automatically inherited by a finite-step numerical discretization. We separately evaluate the numerical behavior of SlotODE under different integration resolutions and ODE solvers in our experiments.

B Additional Experimental Results

B.1 Effectiveness of SlotODE without Adaptive Slot Selection

We further examine whether SlotODE remains effective without adaptive slot selection by evaluating it with fixed number of slots. This comparison isolates the effect of the slot update formulation by removing the adaptive slot-selection mechanism. We compare SlotODE with conventional Slot Attention within the DINOSAUR (Seitzer et al., 2022) framework on COCO and MOVi-C, while keeping the remaining model and training configurations identical. We use 11 slots for MOVi-C and 7 slots for COCO.

Table 8: Comparison of SlotODE and conventional Slot Attention without adaptive slot selection. Both methods use fixed number of slots with pretrained ViT (DINOSAUR), where K denotes number of slots used for each dataset. Numbers in parentheses indicate improvements over Slot Attention.

Dataset	Slot update	K	FG-ARI	mBO_i
COCO	Slot Attention	7	$40.18_{\pm 0.46}$	$27.52_{\pm 0.35}$
	SlotODE	7	$\mathbf{40.81}_{\pm 0.07}$ (+0.63)	$\mathbf{28.55}_{\pm 0.16}$ (+1.03)
MOVi-C	Slot Attention	11	$65.73_{\pm 0.42}$	$34.63_{\pm 0.22}$
	SlotODE	11	$\mathbf{67.27}_{\pm 0.25}$ (+1.54)	$\mathbf{36.51}_{\pm 0.42}$ (+1.88)

As shown in Table 8, SlotODE consistently improves both FG-ARI and mBO_i over conventional Slot Attention on both datasets. These results show that the performance gains of SlotODE persist in the fixed-slot setting and are not limited to its use with adaptive slot selection.

B.2 Additional measures

To further evaluate object discovery beyond the FG-ARI and mBO_i reported in the main experiments, we follow the evaluation setting of AdaSlot (Fan et al., 2024) and report additional measures grouped into pair-counting, matching-based, and information-theoretic metrics.

Pair-counting metrics. Pair-counting metrics compare the predicted and ground-truth partitions through pairs of foreground elements. FG-ARI computes the adjusted Rand index over ground-truth foreground elements, measuring agreement between the two partitions while correcting for agreement expected by chance. Pairwise precision is the fraction of pairs grouped together by the prediction that belong to the same ground-truth object, whereas pairwise recall is the fraction of ground-truth same-object pairs that are grouped together by the prediction. Pairwise F1 is the harmonic mean of pairwise precision and recall.

Overlap- and matching-based metrics. These metrics evaluate the overlap or correspondence between predicted regions and ground-truth objects. mBO computes, for each ground-truth object, the intersection-over-union with its best-overlapping predicted mask and averages the resulting values. Hungarian-matched mIoU averages the intersection-over-union of predicted and ground-truth masks after one-to-one matching. CorLoc measures the fraction of positive images in which a predicted bounding box correctly localizes an object of the target class, typically using an intersection-over-union threshold of 0.5. Purity assigns each predicted cluster to its most frequent ground-truth label and measures the fraction of elements consistent with these many-to-one assignments.

Information-theoretic metrics. NMI and AMI measure the mutual information between the predicted and ground-truth partitions. NMI normalizes mutual information using the entropies of the two partitions, whereas AMI additionally adjusts for the mutual information expected by chance.

Tables 9, 10, and 11 report the results on COCO and MOVi-C. SlotODE consistently improves over conventional Slot Attention across all reported metrics on both datasets. The improvements are observed across pair-counting, matching-based, and information-theoretic measures, showing that the gains of SlotODE are not limited to the FG-ARI and mBO_i metrics used in the main experiments.

Table 9: Comparison by pair-counting metrics on COCO and MOVi-C.

Dataset	Slot Update	FG-ARI	Precision	Recall	F1
COCO	Slot Attention	$41.12_{\pm 0.41}$	$81.44_{\pm 0.08}$	$67.79_{\pm 0.00}$	$69.31_{\pm 0.20}$
	SlotODE	$\mathbf{42.63}_{\pm 0.17}$	$\mathbf{81.97}_{\pm 0.06}$	$\mathbf{69.74}_{\pm 0.22}$	$\mathbf{70.63}_{\pm 0.37}$
MOVi-C	Slot Attention	$74.78_{\pm 0.69}$	$87.61_{\pm 0.44}$	$83.13_{\pm 1.04}$	$83.86_{\pm 0.71}$
	SlotODE	$\mathbf{79.15}_{\pm 0.08}$	$\mathbf{88.53}_{\pm 0.34}$	$\mathbf{88.35}_{\pm 0.26}$	$\mathbf{87.26}_{\pm 0.06}$

Table 10: Comparison by matching-based metrics on COCO and MOVi-C.

Dataset	Slot Update	mBO	mIoU	CorLoc	Purity
COCO	Slot Attention	$27.64_{\pm 0.35}$	$26.67_{\pm 0.02}$	$51.45_{\pm 0.56}$	$68.48_{\pm 0.21}$
	SlotODE	$\mathbf{29.62}_{\pm 0.07}$	$\mathbf{28.02}_{\pm 0.09}$	$\mathbf{57.03}_{\pm 0.57}$	$\mathbf{69.58}_{\pm 0.47}$
MOVi-C	Slot Attention	$36.00_{\pm 0.30}$	$34.70_{\pm 0.13}$	$76.27_{\pm 0.92}$	$84.07_{\pm 0.16}$
	SlotODE	$\mathbf{38.62}_{\pm 0.13}$	$\mathbf{36.60}_{\pm 0.15}$	$\mathbf{81.19}_{\pm 0.13}$	$\mathbf{87.40}_{\pm 0.16}$

Table 11: Comparison by information-theoretic metrics on COCO and MOVi-C.

Dataset	Slot Update	AMI	NMI
COCO	Slot Attention	$46.44_{\pm 0.25}$	$46.60_{\pm 0.24}$
	SlotODE	$\mathbf{47.06}_{\pm 0.18}$	$\mathbf{47.27}_{\pm 0.08}$
MOVi-C	Slot Attention	$78.00_{\pm 0.24}$	$78.21_{\pm 0.33}$
	SlotODE	$\mathbf{80.95}_{\pm 0.35}$	$\mathbf{80.88}_{\pm 0.25}$

B.3 Additional Qualitative Comparisons of Slot-State Trajectories

Figure 2 presents additional qualitative comparisons between SlotODE and conventional Slot Attention on MOVi-C. As in the main text, both methods use the same adaptive slot-selection mechanism, and we visualize only the retained slots. For each example, the left column shows the corresponding slot-wise segmentation masks, while the middle and right columns show the evolution of the slot states from initialization to the final update after projection onto a two-dimensional PCA space. For SlotODE, the arrows indicate the local direction of the learned vector field along the projected trajectory. These additional examples show that the qualitative trends observed in Figure 1 remain consistent across additional samples.

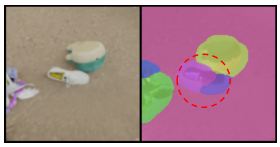
B.4 Qualitative Comparison of Object Segmentation

Figures 3, 4, and 5 present qualitative object segmentation results on COCO, MOVi-C, and MOVi-E, respectively. We compare SlotODE with AdaSlot, MetaSlot, and SPOT with MLP and autoregressive decoders under the object discovery settings considered in the main experiments. For SlotODE, we use the DINOSAUR framework with a frozen pretrained ViT encoder and the adaptive slot-selection mechanism of AdaSlot. The number of slots follows the dataset-specific configuration of each method. AdaSlot uses 11, 11, and 24 slots on COCO, MOVi-C, and MOVi-E, respectively. MetaSlot uses 7, 11, and 24 slots, while SPOT uses 7, 11, and 24 slots on the corresponding datasets. SlotODE uses 11, 11, and 24 candidate slots, respectively.

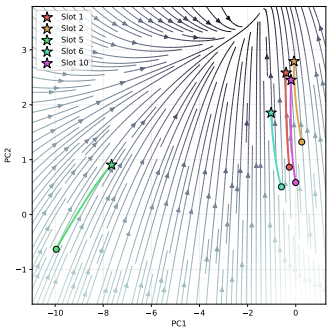
Across the three datasets, SlotODE successfully segments individual objects in the scene. Compared with AdaSlot, which uses the same adaptive slot-selection mechanism with conventional Slot Attention updates, we observe cases where AdaSlot splits parts of a single object across multiple slots. We also observe that SlotODE can assign separate masks to visually distinct regions that are not annotated in the ground-truth masks. For example, in the third row of the COCO results, SlotODE separately segments the wheels of the truck, while in the third row of the MOVi-C results, it also captures background buildings that are not individually annotated in the ground truth. These qualitative observations are consistent with the object discovery results reported in the main experiments.



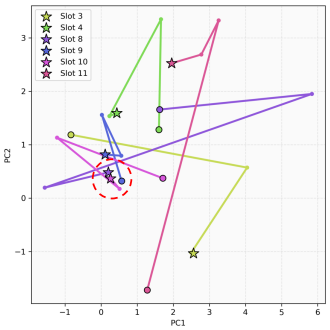
(a) SlotODE masks



(b) Slot Attention masks



(c) SlotODE trajectory



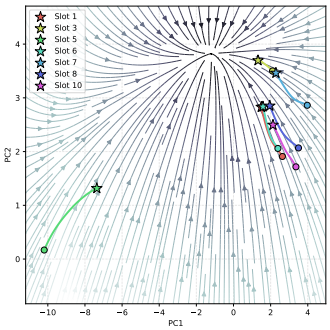
(d) Slot Attention trajectory



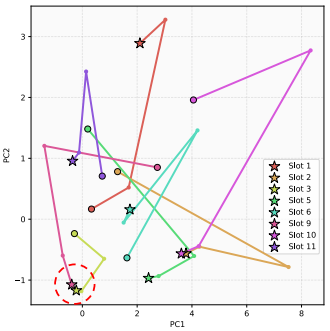
(e) SlotODE masks



(f) Slot Attention masks



(g) SlotODE trajectory



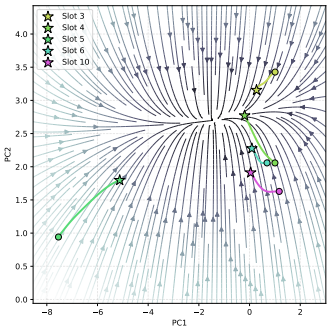
(h) Slot Attention trajectory



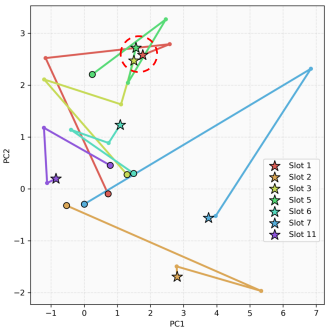
(i) SlotODE masks



(j) Slot Attention masks

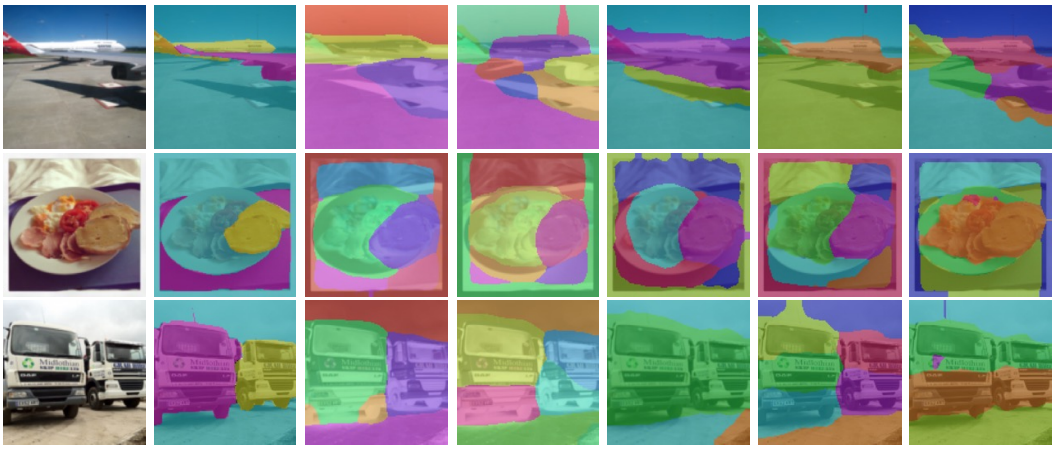


(k) SlotODE trajectory



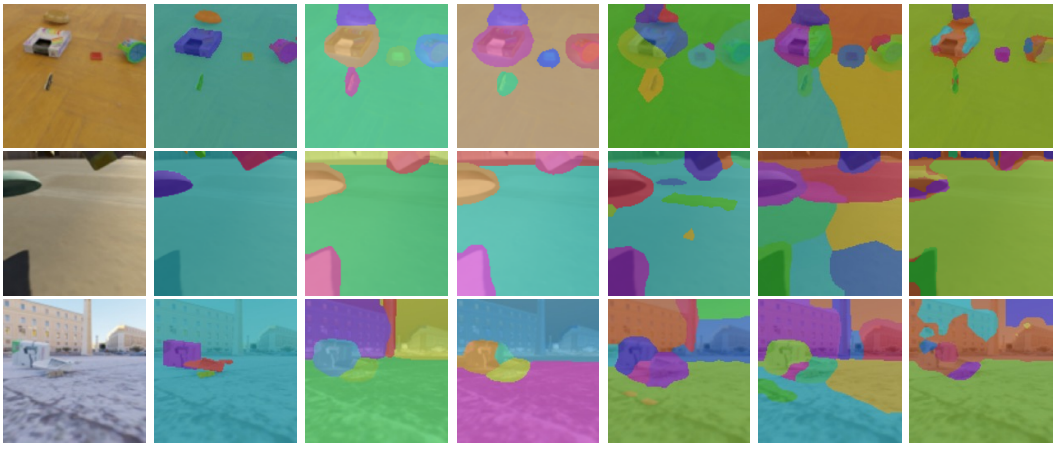
(l) Slot Attention trajectory

Figure 2: Qualitative comparison of object segmentation mask and slot update trajectories between SlotODE and Slot Attention. Circles and stars mark the initial and final slot states, with lines showing the update trajectories and colors match slot indices to segmentation masks. Only slots retained by AdaSlot's selection are shown for both methods. We highlight overlapping slots in the projected Slot Attention state space, which correspond to fragmented segmentation masks.



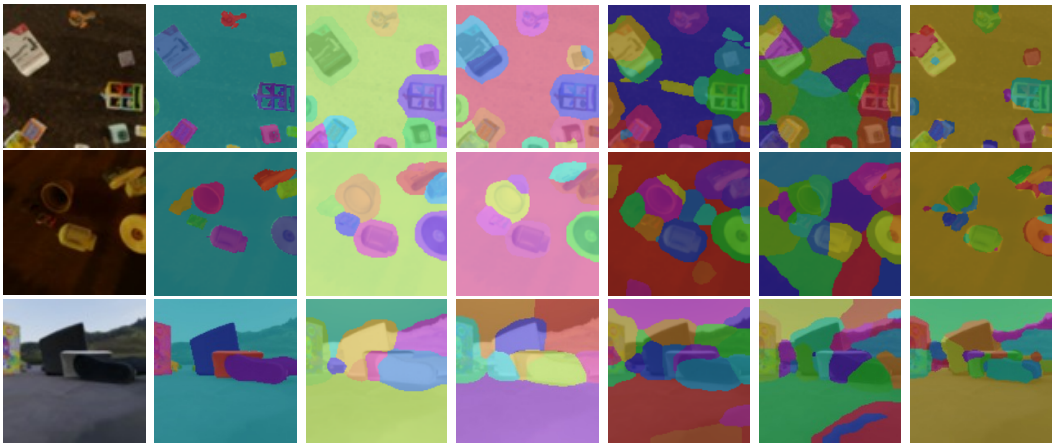
(a) Image (b) GT mask (c) Ours (d) AdaSlot (e) MetaSlot (f) SPOT$_{\text{MLP}}$ (g) SPOT$_{\text{Auto}}$

Figure 3: Object segmentation results on COCO. Each example shows the predicted object segmentation obtained from the refined slot representations.



(a) Image (b) GT mask (c) Ours (d) AdaSlot (e) MetaSlot (f) SPOT$_{\text{MLP}}$ (g) SPOT$_{\text{Auto}}$

Figure 4: Qualitative object segmentation results on MOVi-C. Each example shows the predicted object segmentation obtained from the refined slot representations.



(a) Image (b) GT mask (c) Ours (d) AdaSlot (e) MetaSlot (f) SPOT$_{\text{MLP}}$ (g) SPOT$_{\text{Auto}}$

Figure 5: Qualitative object segmentation results on MOVi-E. Each example shows the predicted object segmentation obtained from the refined slot representations.

C Implementation Details

C.1 SlotODE vector field parameterization with Scaled Spectral Normalization

Vector field parameterization. The SlotODE vector field uses the current slot state, the slot-specific aggregated feature evidence, and the time embedding to predict the time derivative of each slot state. For each slot i, the input to the network f_θ used to parameterize the vector field is

$$\boldsymbol{z}_i(t) = [\, \boldsymbol{s}_i(t);\, \boldsymbol{u}_i(t);\, \phi(t)\,], \tag{70}$$

where $\boldsymbol{s}_i(t) \in \mathbb{R}^D$ is the current slot representation, $\boldsymbol{u}_i(t) \in \mathbb{R}^{D_u}$ is the aggregated feature evidence, and $\phi(t) \in \mathbb{R}^D$ is the time embedding.

The vector field parameterization is implemented as a three-layer MLP with scaled spectral normalization applied to each linear layer. Following spectral normalization (SN) (Miyato et al., 2018), we control the spectral norm of each linear transformation using a relaxed spectral scale, as described below. The first two linear layers are followed by constrained Dynamic Tanh (DyT) activations (Zhu et al., 2025), while the final layer is followed by an element-wise $\tanh$. The final $\tanh$ bounds each coordinate of the predicted slot-state derivative, while DyT provides smooth nonlinear transformations without relying on input-dependent normalization statistics.

Time embedding. We explicitly condition the shared network on time t. The scalar time input is first encoded using a sinusoidal positional embedding and then passed through a two-layer MLP with a GELU activation. The resulting time embedding $\phi(t)$ is concatenated with the current slot state and the aggregated feature evidence as input to the network. This allows the shared network to represent different dynamics at different times during the update process.

Scaled spectral normalization. We apply scaled spectral normalization to all linear transformations in the network used to parameterize the SlotODE vector field. For each weight matrix W_ℓ, we parameterize the effective weight as

$$\widetilde{W}_\ell = \rho \frac{W_\ell}{\sigma(W_\ell)}, \tag{71}$$

where $\sigma(W_\ell)$ denotes its spectral norm and ρ controls the spectral scale. Standard spectral normalization corresponds to $\rho = 1$ (Miyato et al., 2018). Motivated by prior observations that overly strict spectral constraints can be restrictive (Lin et al., 2021), we allow $\rho > 1$ to relax the unit spectral constraint while retaining explicit control over the spectral norm of each linear transformation. Table 12 reports the effect of different spectral scales on MOVi-C. Based on this comparison, we use $\rho = 3$ throughout our experiments.

Table 12: Effect of the spectral normalization scale ρ on MOVi-C. All variants use the same SlotODE architecture with 11 slots and an MLP decoder.

Scale ρ	FG-ARI↑	mBO$_i$↑
1.0	$77.93_{\pm 0.13}$	$37.88_{\pm 0.22}$
3.0	$79.15_{\pm 0.08}$	$38.62_{\pm 0.13}$
5.0	$79.18_{\pm 0.15}$	$38.56_{\pm 0.11}$

C.2 Slot Initialization

Slot initialization defines the initial condition $\boldsymbol{S}_0$ from which the SlotODE dynamics evolve. Conventional Slot Attention initializes each slot by sampling from a learned Gaussian distribution, $\boldsymbol{s}_i^{(0)} \sim \mathcal{N}(\mu_\psi, \sigma_\psi)$, and subsequently updates these randomly initialized slots through iterative updates. Motivated by the pairwise distinct initialization assumed in our theoretical analysis, we use an input-dependent initialization designed to encourage separation among the initial slot states while remaining deterministic for a fixed input.

Specifically, we first construct a set of fixed orthogonal directions, one for each candidate slot. These directions are used to compute slot-specific scores over the input features. We then apply a softmax

normalization across slots so that each input feature has a relative preference among the candidate slots. Finally, the initial slot states are obtained by weighted aggregation of the input features using these normalized assignment weights.

Let $\boldsymbol{q}_i$ denote the fixed orthogonal direction associated with slot i and $\boldsymbol{x}_m$ the m-th input feature. We compute

$$a_{im} = \frac{\exp(\boldsymbol{q}_i^\top \boldsymbol{x}_m)}{\sum_{j=1}^{K} \exp(\boldsymbol{q}_j^\top \boldsymbol{x}_m)}, \tag{72}$$

and construct the initial slot as

$$\boldsymbol{s}_i(0) = \frac{\sum_{m=1}^{M} a_{im} \boldsymbol{x}_m}{\sum_{m=1}^{M} a_{im}}. \tag{73}$$

The fixed orthogonal directions provide slot-specific anchors, while the feature aggregation makes the resulting initialization input-dependent. In contrast to using the orthogonal directions directly as slot embeddings, this construction initializes each slot from differently weighted evidence in the input features.

We compare this initialization with random initialization, fixed orthogonal slot embeddings, and trainable slot embeddings in Table 6a. The proposed initialization is used throughout the main experiments. Notably, applying the same initialization to conventional Slot Attention (AdaSlot) still yields lower performance than SlotODE, indicating that the initialization alone does not account for the observed gains.

C.3 Adapting to Dynamic Number of Slots

Adaptive slot selection. AdaSlot (Fan et al., 2024) addresses varying numbers of objects in an input scene by dynamically selecting a subset of slots from a fixed set of candidate slots. Specifically, it first generates $K_{\max}$ candidate slots and applies a lightweight slot-selection network to each slot. For each candidate slot $\boldsymbol{s}_i$, the two outputs represent the probabilities of dropping or retaining the slot. A hard binary mask $Z_i \in \{0, 1\}$ is then sampled using Gumbel-Softmax with straight-through estimation, enabling discrete slot selection while retaining end-to-end differentiability.

The resulting binary mask is applied during decoding such that unselected slots do not contribute to reconstruction. In particular, the decoder masks associated with slots for which $Z_i = 0$ are suppressed, while the retained slots contribute normally to the reconstructed output.

Since reconstruction alone tends to favor retaining more slots, AdaSlot additionally introduces a sparsity regularization term based on the expected number of retained slots:

$$\mathcal{L}_{\text{reg}} = \mathbb{E}\left[\sum_{i=1}^{K_{\max}} Z_i\right] = \sum_{i=1}^{K_{\max}} \mathbb{E}[Z_i]. \tag{74}$$

The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda \mathcal{L}_{\text{reg}}, \tag{75}$$

where λ controls the trade-off between reconstruction quality and the number of retained slots.

Modification of sparsity regularization. In our reproduction, we observed that the linear sparsity penalty was sensitive to its weighting coefficient under our training configuration. Because the linear objective monotonically favors fewer retained slots, a sufficiently large coefficient can impose excessive pressure toward sparse slot selection, whereas a small coefficient may provide insufficient control over the number of retained slots. Table 13 shows this sensitivity on MOVi-C. Performance remains comparable within a relatively narrow range of coefficients, but decreases sharply as the linear weight is increased.

To reduce this sensitivity, we use a target-centered quadratic regularization. Let

$$r = \frac{1}{K_{\max}} \sum_{i=1}^{K_{\max}} \mathbb{E}[Z_i] \tag{76}$$

Table 13: Sensitivity to the linear sparsity regularization coefficient in our AdaSlot reproduction on MOVi-C. All experiments use 11 candidate slots and an MLP decoder.

Linear weight λ	FG-ARI↑	mBO$_i$↑
0.04	74.84	35.43
0.05	74.45	35.86
0.06	73.50	33.97
0.07	31.11	6.21
0.10	2.91	3.62

denote the expected proportion of retained slots. We define

$$\mathcal{L}_{\text{reg}}^{\text{quad}} = \lambda_q \left(r - r_0\right)^2, \tag{77}$$

where $r_0 = 0.5$ denotes the target retention ratio and λ_q controls the regularization strength.

Unlike the linear penalty, which continually favors fewer retained slots, the quadratic objective penalizes deviations from a target retention level in both directions. It therefore reduces the tendency toward excessively sparse slot selection when the regularization term becomes dominant.

Using the reported linear coefficient directly in our reproduction resulted in an excessively sparse solution under our implementation and training environment. We therefore use the quadratic regularization above for the reproduced AdaSlot baseline. As shown in Table 14, this modification recovers performance close to the reported AdaSlot result.

Table 14: Effect of the sparsity regularization used in our AdaSlot reproduction on MOVi-C.

Setting	FG-ARI↑	mBO$_i$↑
Official (Fan et al., 2024)	75.59	35.64
Reproduced (linear, $\lambda = 0.1$)	2.91	3.62
Reproduced (quadratic)	75.22	35.60

Slot-selection network configuration. The slot-selection network is implemented as a two-layer MLP applied independently to each candidate slot. Its configuration is summarized in Table 15.

Table 15: Configuration of the slot-selection network.

Configuration	Value
Input dimension	D
Initial normalization	LayerNorm
Hidden dimension	$4 \times D$
Activation	ReLU
Output dimension	2

D Training Configurations

D.1 SlotODE Training Configurations

Table 16 summarizes the training and architecture configurations used across datasets.

For training SlotODE in complex datasets (MOVi-C/E, COCO), we follow the DINOSAUR framework (Seitzer et al., 2022) for the overall training and evaluation setup. For all datasets, we use a ViT-B/16 (Dosovitskiy et al., 2020) encoder initialized with DINO v1 pretrained weights (Caron et al., 2021) and keep the visual encoder frozen during training. Following the MLP-based decoding setup, we use an MLP decoder to reconstruct the encoder feature representations from the updated slot states.

For CLEVR10, we train the entire model end-to-end from scratch following the training configuration of AdaSlot (Fan et al., 2024). We use 11 slots, and the encoder and decoder architectures used for CLEVR10 are summarized in Table 17.

SlotODE is trained using three integration steps with Adam and an exponential learning-rate schedule with a 10K-step warm-up. The number and dimensionality of slots are adjusted for each dataset, while the remaining optimization settings are kept consistent across datasets.

For the vector field parameterization, we use scaled spectral normalization with $\rho = 3.0$ for all spectrally normalized linear layers. Further discussion of scaled spectral normalization is provided in Appendix C.1. For adaptive slot selection, we use the regularization proposed by Fan et al. (2024) with a weighting coefficient of $\lambda = 1.0$ across all datasets. Further details on adaptive slot selection are provided in Appendix C.3.

Table 16: Training configurations for SlotODE across datasets.

Category	Configuration	COCO	MOVi-C	MOVi-E	CLEVR10
Training	Training steps	500K	500K	500K	500K
	Learning rate	4×10^{-4}	4×10^{-4}	4×10^{-4}	4×10^{-4}
	LR scheduler	Exponential	Exponential	Exponential	Exponential
	LR warm-up	10K	10K	10K	10K
	Optimizer	Adam	Adam	Adam	Adam
	Gradient clipping	1.0	1.0	1.0	1.0
	Batch size	64	64	64	64
	Sparsity loss λ	1.0	1.0	1.0	0.3
SlotODE	Number of slots	11	11	24	11
	Refinement steps	3	3	3	3
	Slot dimension	256	128	128	64
	Hidden dimension	512	512	512	128
	Spectral Norm Scale ρ	3.0	3.0	3.0	3.0
Encoder	Architecture	ViT-B/16	ViT-B/16	ViT-B/16	CNN
	Pretrained weights	DINO v1	DINO v1	DINO v1	-
	Image size	224	224	224	224
	Number of tokens	196	196	196	-
	Input dimension	768	768	768	-
Decoder	Type	MLP	MLP	MLP	CNN
	Number of layers	4	4	4	-
	Hidden dimension	2048	1024	1024	-

Table 17: Encoder and decoder architectures used for CLEVR10.

Module	Architecture	Kernel / Stride	Activation
Encoder	Conv$(3, 64)$ + 3×Conv$(64, 64)$	5×5 / 1	ReLU
Decoder	ConvT$(D, 64)$ + 3×ConvT$(64, 64)$	5×5 / 2	ReLU
	ConvT$(64, 64)$	5×5 / 1	ReLU
	ConvT$(64, D_{\text{out}})$	3×3 / 1	–

D.2 BASELINES

In this section, we describe the training configurations used to reproduce the baselines. Our reproduced results are reported over five independent runs. For entries where the original work does not report standard deviations, we present only the reported mean performance, and use "–" where the original paper does not report results.

Dinosaur. We reproduce DINOSAUR following the training configuration reported in the original work (Seitzer et al., 2022). The official results reported in the original paper and our reproduced results are summarized in Table 18. The number of slots was set to 7 for COCO, 11 for MOVi-C,

Table 18: Comparison between the official and reproduced DINOSAUR results.

Dataset	Setting	FG-ARI	mBO_i
COCO	Official (Seitzer et al., 2022)	40.40	27.70
	Reproduced	$40.18_{\pm 0.46}$	$27.52_{\pm 0.35}$
MOVi-C	Official (Seitzer et al., 2022)	66.00	35.00
	Reproduced	$65.73_{\pm 0.42}$	$34.63_{\pm 0.22}$
MOVi-E	Official (Seitzer et al., 2022)	65.10	35.50
	Reproduced	$62.21_{\pm 0.33}$	$31.12_{\pm 0.26}$

and 24 for MOVi-E. For MOVi-E, we report results obtained with a ViT-B/16 encoder, whereas the original DINOSAUR paper reports results using ViT-B/8.

AdaSlot. We reproduce the AdaSlot baseline following the training configuration reported in the original work (Fan et al., 2024). The training configurations used for reproduction are summarized in Table 19. The number of slots was set to 11 for COCO, 11 for MOVi-C, and 24 for MOVi-E. Note that the original paper used 33 slots for COCO. We also modify the additional regularization loss used for adaptive slot selection, as discussed in Appendix C.3. Specifically, we set the balancing coefficient λ to 0.5 for COCO, MOVi-C, and MOVi-E. For CLEVR10, we use 11 slots, and the encoder and decoder architectures used for CLEVR10 are summarized in Table 17. The official results reported in the original paper and our reproduced results are summarized in Table 20.

Table 19: Training configurations used for reproducing AdaSlot and MetaSlot.

Category	Configuration	AdaSlot	MetaSlot	
		CLEVR10	MOVi-C	MOVi-E
Training	Training steps	500K	100K	100K
	Learning rate	4×10^{-4}	4×10^{-4}	4×10^{-4}
	LR scheduler	Exponential	Cosine	Cosine
	LR warm-up	10K	10K	10K
	Optimizer	Adam	Adam	Adam
	Gradient clipping	1.0	1.0	1.0
	Batch size	64	64	64
	Sparsity loss λ	0.3	–	–
Slot Attention	Number of slots	11	11	24
	Refinement steps	3	3	3
	Slot dimension	64	128	128
	Hidden dimension	128	1024	1024
	Codebook size	–	512	512
Encoder	Architecture	CNN	ViT-S/14	ViT-S/14
	Pretrained weights	-	DINO v2	DINO v2
	Image size	224	224	224
	Number of tokens	-	256	256
	Input dimension	-	384	384
Decoder	Type	CNN	MLP	MLP
	Number of layers	-	4	4
	Hidden dimension	-	2048	2048

MetaSlot. We reproduce the MetaSlot baseline following the training configuration reported in the original work (Liu et al., 2026). Since the original paper does not report results on MOVi-C and MOVi-E, we extend the baseline to these datasets using the corresponding training configurations from DINOSAUR (Seitzer et al., 2022), while retaining the original MetaSlot architecture. The training configurations used for reproduction are summarized in Table 19. The official results reported in the original paper and our reproduced results are summarized in Table 21.

Table 20: Comparison between the official and reproduced AdaSlot results.

Dataset	Setting	FG-ARI↑	mBO$_i$↑
COCO	Official (Fan et al., 2024)	39.00	27.36
	Reproduced	$41.12_{\pm 0.41}$	$27.64_{\pm 0.35}$
MOVi-C	Official (Fan et al., 2024)	75.59	35.64
	Reproduced	$75.22_{\pm 0.34}$	$35.60_{\pm 0.41}$
MOVi-E	Official (Fan et al., 2024)	76.73	29.83
	Reproduced	$76.04_{\pm 0.46}$	$29.76_{\pm 0.51}$
CLEVR10	Official (Fan et al., 2024)	97.65	22.51
	Reproduced	$94.72_{\pm 0.7}$	$21.04_{\pm 0.1}$

Table 21: Comparison between the official and reproduced MetaSlot results.

Dataset	Setting	FG-ARI↑	mBO$_i$↑
COCO	Official (Liu et al., 2026)	$40.30_{\pm 0.50}$	$29.50_{\pm 0.20}$
	Reproduced	$39.22_{\pm 0.91}$	$28.91_{\pm 0.28}$
MOVi-C	Official (Liu et al., 2026)	–	–
	Reproduced	$72.20_{\pm 1.78}$	$38.51_{\pm 0.26}$
MOVi-E	Official (Liu et al., 2026)	–	–
	Reproduced	$72.92_{\pm 1.06}$	$29.55_{\pm 0.16}$

SPOT. We reproduce the SPOT baseline following the training configuration reported in the original work (Kakogeorgiou et al., 2024). SPOT employs self-training which distills the decoder attention masks to the student Slot Attention module with path-order sequence permutation, and ensembling results of different permutation strategies for training Slot Attention. For the MLP-based reproduction, we only use ST during training, since the patch-order permutation and ensembling are not applicable, as these techniques are specific to the autoregressive decoding setting. We therefore provide results using autoregressive Transformer decoders for SPOT.

Since the original paper does not report results for the MLP decoder on MOVi-C and MOVi-E, we follow the same dataset-specific training configuration as the autoregressive Transformer of SPOT for these datasets. The official results reported in the original paper and our reproduced results are summarized in Table 22.

Table 22: Comparison between the official and reproduced SPOT results.

Dataset	Setting	Decoder	FG-ARI↑	mBO$_i$↑
COCO	Official (Kakogeorgiou et al., 2024)	MLP	42.50	28.40
	Reproduced	MLP	$40.35_{\pm 2.73}$	$26.91_{\pm 0.58}$
	Official (Kakogeorgiou et al., 2024)	Auto	$37.00_{\pm 0.20}$	$35.00_{\pm 0.10}$
	Reproduced	Auto	$36.71_{\pm 0.99}$	$34.79_{\pm 0.64}$
MOVi-C	Official (Kakogeorgiou et al., 2024)	MLP	–	–
	Reproduced	MLP	$66.77_{\pm 0.61}$	$35.63_{\pm 0.32}$
	Official (Kakogeorgiou et al., 2024)	Auto	$52.30_{\pm 3.30}$	$47.30_{\pm 1.20}$
	Reproduced	Auto	$49.22_{\pm 3.56}$	$45.85_{\pm 2.76}$
MOVi-E	Official (Kakogeorgiou et al., 2024)	MLP	–	–
	Reproduced	MLP	$63.26_{\pm 0.58}$	$31.16_{\pm 0.36}$
	Official (Kakogeorgiou et al., 2024)	Auto	$56.80_{\pm 4.30}$	$40.10_{\pm 1.20}$
	Reproduced	Auto	$50.64_{\pm 6.68}$	$40.04_{\pm 1.31}$