Loci-Segmented:
Improving Scene Segmentation Learning

Manuel Traub [1] **Frederic Becker** [1] **Adrian Sauter** [1] **Sebastian Otte** [1] **Martin V. Butz** [1]

Abstract

Current slot-oriented approaches for compositional scene segmentation from images and videos rely on provided background information or slot assignments. We present a segmented location and identity tracking system, Loci-Segmented (Loci-s), which does not require either of this information. It learns to dynamically segment scenes into interpretable background and slot-based object encodings, separating rgb, mask, location, and depth information for each. The results reveal largely superior video decomposition performance in the MOVi datasets and in another established dataset collection targeting scene segmentation. The system's well-interpretable, compositional latent encodings may serve as a foundation model for downstream tasks.

1. Introduction

Visual scene understanding from images or videos presents unique challenges. Classical architectures such as CNNs (Liu et al., 2022) or ViTs (Vision Transformers) (Dosovitskiy et al., 2020) exacerbate existing limitations, being data-hungry, susceptible to adversarial attacks, and low on interpretability. To address these challenges, slot attention mechanisms have emerged as a promising avenue (Locatello et al., 2020). These architectures address critical challenges

[1]Neuro-Cognitive Modeling Group, University of Tübingen, Germany. Correspondence to: Manuel Traub <manuel.traub@uni-tuebingen.de>, Frederic Becker <frederic.becker@uni-tuebingen.de>.
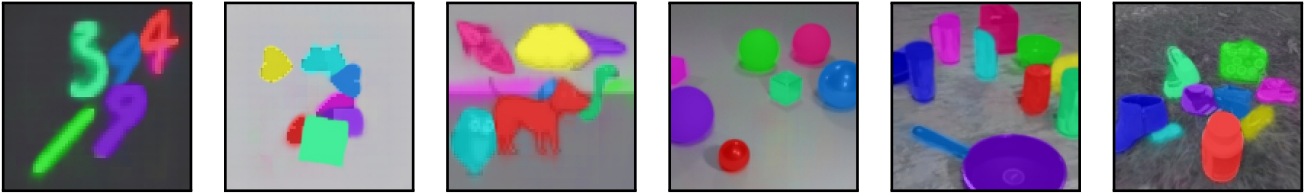
Source Code: github.com/CognitiveModeling/Loci-Segmented

associated to the binding problem (Greff et al., 2020). In particular, they offer a way to bind features into 'slots' that dynamically represent distinct entities in a scene (Locatello et al., 2020), building upon prior work in attention mechanisms (Vaswani et al., 2017) and capsule networks (Sabour et al., 2017). Current state-of-the-art systems include the highly potent Slot Attention for Video model (SAVi++, Elsayed et al.) and the location and identity tracking slot-based recurrent architecture (Loci, Traub et al., 2023b). Still, both systems have a weakness: SAVi++ relies on supervised slot assignments upon trial initialization while Loci relies on the provision of static background information.

Loci has shown superior performance on the CATER challenge, in which objects (balls and cones) are transported hidden within other objects (cones). It is rather closely related to other slot-based object processing architectures including SAVi++ (Elsayed et al.; Locatello et al., 2020; Kipf et al., 2022; Wu et al., 2023), surveyed in (Yuan et al., 2023). It differs in (i) its slot-specific encoding approach that starts from pixels, (ii) its emergent disentanglement of objects from positions, and (iii) its event-oriented internal processing loop. However, its reliance on a provided, static background module prevented processing more complex dynamic backgrounds or moving cameras. Moreover, Loci was not able to profit from or predict depth information.

In this work, we enable Loci to deal with (i) dynamic complex backgrounds, (ii) videos where the camera is moving, and (iii) depth information without initial slot assignments and without the provision of background information during evaluation. Additional improvements enable the segmentation of scenes with more complex and diverse objects. Thereby, we enhance the state-of-the-art of scene segmentation algorithms in both the MOVi-* datasets (Greff et al.,

Figure 1. Exemplar slot-based autoregressive segmentations inferred by Loci-s, generalizing to 7-10 objects while being trained on only 4-6 objects. Shown are moving MNIST digits and dSprites, the Abstract Scene dataset, CLEVR, SHOP VRB, and a combination of GSO and HDRI-Haven, all of which were considered in a recent compositional scene understanding review (Yuan et al., 2023).
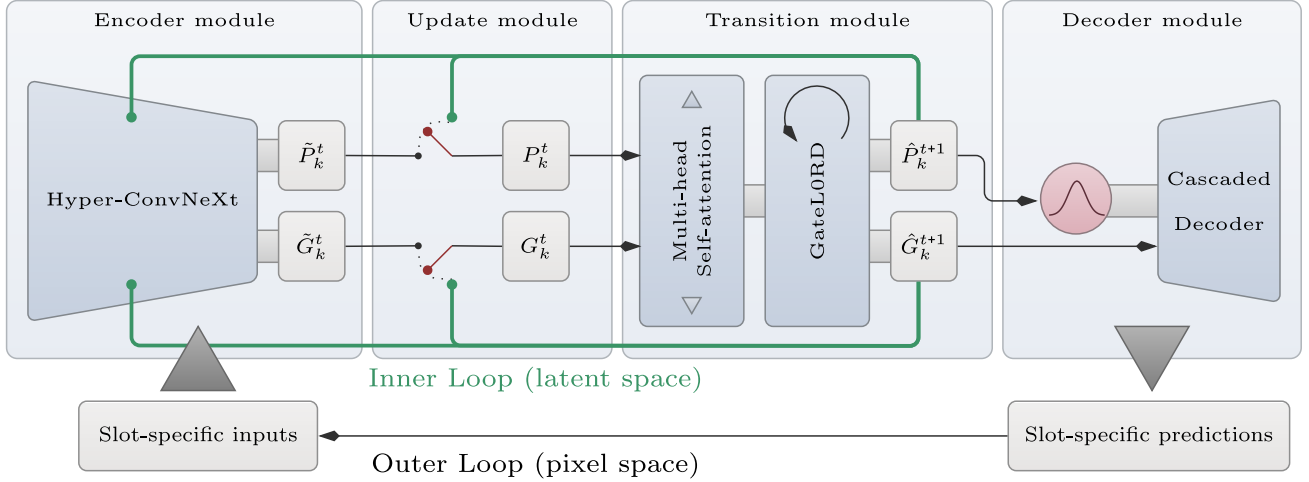
Figure 2. The primary Loci-s architecture features: a Hyper-ConvNeXt encoder, which generates Position and Gestalt codes slot-individually; an Update module, which adaptively fuses current encoder information with prior temporal predictions; a Transition module, which calculates object dynamics via a GateL0RD layer (a strongly gated RNN, cf. Gumbsch et al., 2021) and inter-slot interactions via self-attention; finally, a Decoder module computing sequential slot-wise predictions including depth estimates for the subsequent frame.

2022) as well as in a scene segmentation dataset benchmark suite used in a recent review paper (Yuan et al., 2023). Our key contributions are:

(i) We introduce Loci-Segmented a compositional slot-based video segmentation model that makes several key improvements over related slot-based compositional scene segmentation algorithms (Traub et al., 2023b; Yuan et al., 2023; Greff et al., 2022) by introducing a *background module*, implementing enhanced *encoder* and *decoder* pipelines, and including *Scene-Relative-Depth* as target and, optional, as input.

(ii) Demonstrating superior segmentation performance on the challenging multi-object video benchmark (MOVi) (Greff et al., 2022) through extensive supervised per-taining while not relying on ground truth slot initialization during testing like previous approaches.

(iii) Generalization to and largely outperforming all other methods on a recently introduced compositional scene segmentation benchmark suite (Yuan et al., 2023).

(iv) Providing well interpretable latent codes that by design disentangle mask, depth and texture (rgb) codes.

2. Loci Architecture

Before detailing the novel extensions in Loci-s, we briefly introduce Loci (Traub et al., 2023b), a slot-based object-oriented processing architecture that consists of a slot-wise encoder, a transition, and a decoder module (cf., Figure 2).

Encoder Module: In contrast to other slot-based approaches (Yuan et al., 2023), Loci slots each start from the input image. At time point t, each slot k receives as input the actual video frame I^t, the previous prediction error E^t, and a background mask $\hat{M}_{bg}^t$. Moreover, to focus each slot on its own object encoding, previous slot predictions are fed in as additional input, including its predicted position $\hat{Q}_k^t$ encoded as an isotropic Gaussian in pixel space, its visibility mask $\hat{M}_k^{t,v}$ and object mask $\hat{M}_k^{t,o}$ encoded as grayscale images, its RGB image $\hat{R}_k^t$, and the summed visibility masks of the remaining slots $\hat{M}_k^{t,s}$.

As output, the encoder produces two types of codes for each slot: **Gestalt Codes** $\tilde{G}_k^t$**:** A 1D latent representations of an object's appearance, capturing shape, color, texture, and other visual attributes; **Position Codes** $\tilde{P}_k^t$**:** A spatial code including the object's 2D location (x_k, y_k), its size (σ_k), and its distance in depth encoded as a priority code (ρ_k).

Transition Module: The transition module contains a slot-wise recurrent module and a multi-head attention module. The recurrent module implements GateL0RD units, which encode LSTM-like recurrent cells with an even stronger shielding, to foster event-predictive encodings (Gumbsch et al., 2021).

The multi-head attention module enables the object-interaction-oriented exchange of information between slots. As its result, the transition module outputs next object-respective Gestalt-Codes $\hat{G}_k^{t+1}$ and positions $\hat{P}_k^{t+1}$.

Decoder Module: The decoder reconstructs the predicted scene starting from a 3D tensor that combines the Gestalt code as channels with the positional encoding (x_k, y_k, σ_k).
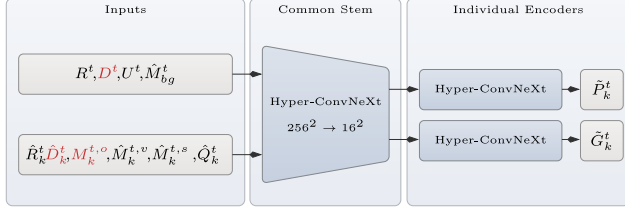
Figure 3. New Encoder design with the added depth information and slot-wise depth and object mask channels (in red). One encoder head processes both common (RGB frame R^t, depth frame D^t, uncertainty mask U^t, background mask $\hat{M}_{bg}^t$) and slot specific inputs (decoder outputs from the previous iteration: slot rgb $\hat{R}_k^t$, slot depth $\hat{D}_k^t$, amodal mask $M_k^{t,o}$, visibility mask $\hat{M}_k^{t,v}$, summed masks from other slots $\hat{M}_k^{t,s}$, the 2d Gaussian position $\hat{Q}_k^t$). In addition to the top-down feedback provided by the slot specific inputs, Hyper-ConvNeXt blocks also provide top-down feedback in the form of dynamic weight residuals computed from predicted Gestalt-Codes $\hat{G}_k^t$.

It then upscales this tensor to the full input resolution via a ResNet. The outputs are an RGB slot image $\hat{R}_k^{t+1}$, visibility mask $\hat{M}_k^{t+1,v}$, and position $\hat{Q}_k^{t+1}$. The scene is finally recomposed by combining the masked RGB outputs with respect to their respective priority codes $\hat{\rho}_k$ and the background mask.

3. Methodology

Loci-s builds on Loci but significantly enhances its abilities: We enable the processing and prediction of depth information; we design an even more dynamic encoder-decoder framework; and we introduce a dedicated background processing module. Detailed Loci-s network wiring and size information can be found in Appendix D.

3.1. Depth as Input

We introduce a novel input channel to the Loci-s model, denoted as Scene-Relative Depth (see Appendix B for more details). Depth normalization is expected to significantly support object segmentation, as object edges will naturally be marked by spatial depth non-linearities.

3.2. Encoder

The encoder and decoder subnetworks adopt a ConvNeXt-like architecture (Liu et al., 2022), replacing the previously used ResNet architecture. Furthermore, we add an inner top-down processing loop to the architecture, which propagates predicted Gestalt code information $\hat{G}_k^t$ directly into the encoder. These codes are utilized within a hypernetwork to compute dynamic residuals for the depth-wise convolutional kernels present in the ConvNeXt blocks of the encoder (see Figure 3). This architectural modification enables the en-
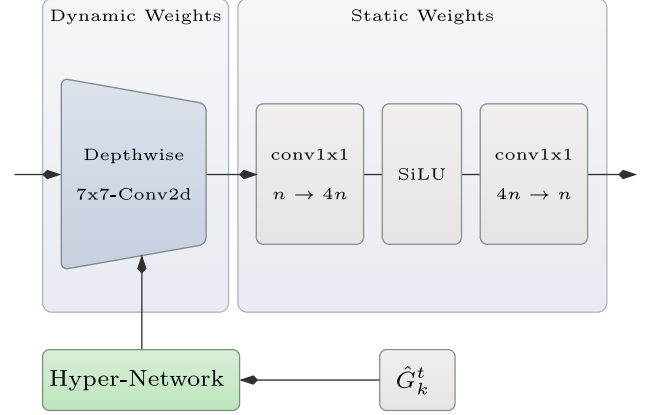

Figure 4. A single Hyper-ConvNeXt block within the encoder where a top-down hyper-network translates the Gestalt-Code predicted in the last iteration $\hat{G}_k^t$ into spatial convolutional kernel weight residuals augmenting the receptive field of the particular slot encoder to be more susceptible to the previously encoded entity.

coder to integrate top-down feedback into its computations, thereby improving the object-specific encoding pipeline.

3.3. Decoder

We introduce a cascaded decoder architecture, shown in Figure 5, and partition the Gestalt Code G_k^t into three segments, each comprising 256 elements. These segments encode mask Gm_k^t, depth Gd_k^t, and RGB channels Gr_k^t. The Mask Decoder module uses element-wise multiplication between the Gestalt Code $\hat{Gm}_k^t$ and a two-dimensional isotropic Gaussian heatmap generated from Position Code $\hat{P}_k^t$. This modulated spatialized Gestalt Code is then subject to a compact convolutional neural network. The Depth Decoder module is implemented by a U-Net architecture. It decodes the depth information via the predicted Depth Gestalt Code $\hat{Gd}_k^t$, which is multiplied layer-wise with the computed object mask, enforcing masked outlines. The RGB Decoder module clones the Depth Decoder architecture but additionally receives the Depth Decoder's output as input. Finally, the RGB image of the encoded object is reconstructed via the RGB Gestalt code, which is layer-wise multiplied with the computed object mask and additionally informed by the generated depth estimations.

The cascading of the decoder architecture does not only encourage a disentangled encoding of an object's mask (i.e., its shape), its distance to the camera (i.e., its depth), and its appearance, but it also facilitates appearance reconstruction because the prediction of the mask is easier and then informs the depth and RGB-pattern reconstruction. Additionally, the cascaded decoder facilitates the reconstruction of the unoccluded raw mask $\hat{M}_k^{t,o}$ and the occlusion-aware mask
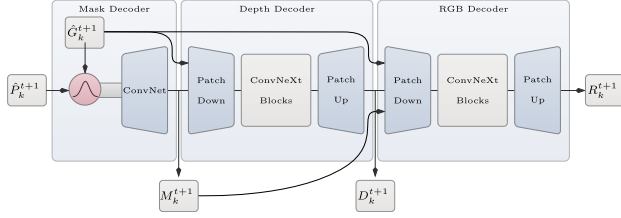
Figure 5. Decoder visualization illustrating the cascaded reconstruction strategy, first decoding the mask, then the depth, and finally the RGB image of a slot-encoded entity. Specifically the Gestalt code is partitioned into three equi-length segments of 256 elements each. The Mask Decoder is implemented by a compact convolutional network; its input comprises the Mask-Gestalt Code $\hat{Gm}_k^t$ modulated by a 2D Gaussian heatmap, which is derived from the Position Code. The Depth Decoder features a U-Net architecture with aggressive down-sampling and up-sampling pathways, altering the spatial resolution by a factor of 16. This Depth Decoder receives as input the Depth-Gestalt Code $\hat{Gd}_k^t$ modulated by the mask output from the Mask Decoder. The RGB Decoder operates on the same principle as the Depth Decoder but incorporates an additional input: the depth map generated by the Depth Decoder.
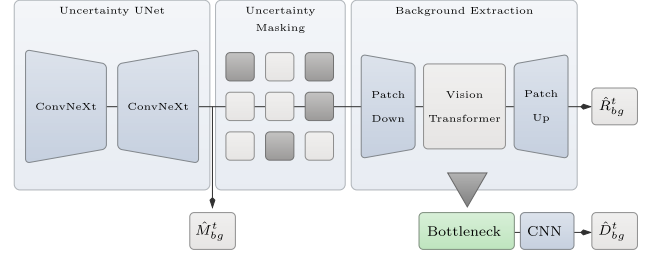

Figure 6. Background module: Input RGB or RGB+Depth is used to compute an Uncertainty or Foreground Mask via a U-Net. The mask is used in a Masked Autoencoder to reconstruct a background representation from the foreground masked input image (RGB or RGB+Depth). Within this process, input patches characterized by high uncertainty are effectively masked out. Moreover, the Depth background reconstruction is subjected to a bottleneck layer, which encourages to encode a regular background without any non-liner foreground depth masks, thereby increasing the complexity for the autoencoder to accurately reconstruct foreground objects.

$\hat{M}_k^t$, because only the Mask-decoder, but not the Depth and RGB decoders, needs to be re-run to generated the occlusion-aware mask.

3.4. Background

Another pivotal enhancement in our work is the development of a Background Module, which is trained prior to the slotted architectural components. This module enables the application of Loci-s to environments with complex backgrounds, featuring both intricate backgrounds and moving cameras. As delineated in Figure 6, this module is bifurcated into two core elements: an Uncertainty Network and a Background Extraction Network.

The Uncertainty Network employs a U-Net architecture with ConvNeXt residual blocks. Skip connections between downsampling and up-sampling layers avoid vanishing gradients. The network is trained in a supervised manner to compositionally segment the foreground in a scene, generating an uncertainty mask that predicts the provided foreground mask from either pure RGB or RGB+Depth depending on the used version (Loci-s or Loci-s_d). It thus learns to deem dynamic foreground objects 'uncertain', in contrast with the generally stable background elements in natural scenes.

The output from the Uncertainty Network serves as a masking function for the Background Extraction Network. This network implements a masked autoencoder using a Vision Transformer. This Vision Transformer is designed to predict both the RGB values and the depth map from either RGB alone or from both RGB and depth maps . By se-

lectively masking-out foreground objects, we introduce a bias favoring the exclusive reconstruction of the background elements. To further accentuate this bias, we constrain the depth reconstruction module with a narrow bottleneck. The network is trained as a background autoencoder by masking the reconstruction loss with the inverse of the foreground mask.

3.5. Pretraining object encodings and decodings

The original Loci architecture was fully trained end-to-end without any information on objects whatsoever. While Loci-s could also be trained in this way, in order to speed-up learning and save computational resources, we implement a sequential supervised pre-training strategy specifically tailored for Loci-s's encoder and decoder components. It is trained on single-object detection and reconstruction tasks.

To initialize it, the encoder is feed with an input frame with all slot-specific inputs nullified except for the slot-specific 2D Gaussian position, which is computed from the ground-truth target mask during this training stage. To avoid reliance on exact position encodings, the encoding is subjected to stochastic perturbations while ensuring its confinement within the mask's boundary. Any slot-specific inputs dependent on other slots are explicitly nullified. Note that during this pretraining stage we thus encourage slots to encode particular masks, but we do not initialize the slots explicitly to ground-truth bounding boxes. During all evaluations, we do not provide any supervised slot information whatsoever. This is in stark contrast to SAVi++, where slots are initialized to the ground-truth bounding boxes in the first frame during training and evaluation (Elsayed et al.).
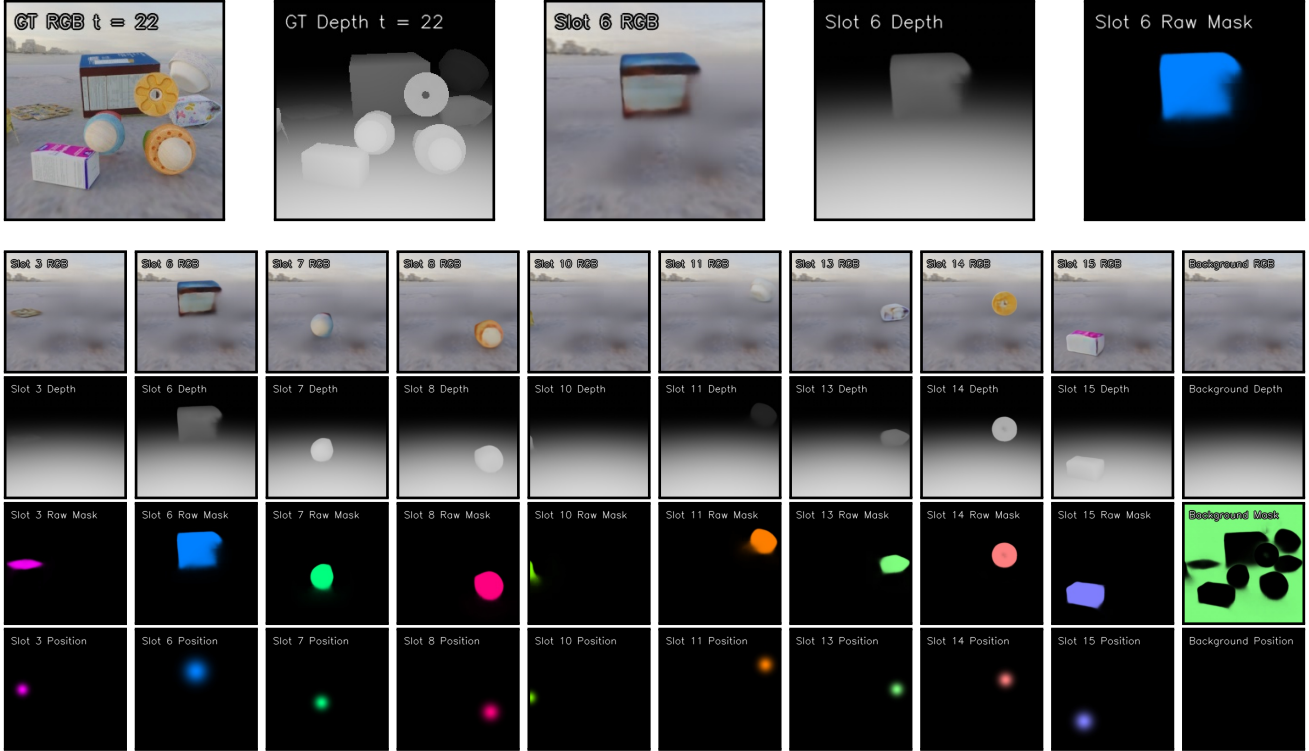
Figure 7. Example of an internal scene segmentation inferred and processed by Loci-s. First row: ground truth RGB image and depth map as well as details of the encoding of the partially occluded large box. Columns in subsequent rows: Interpretable slot-wise decomposition (columns) of the inputs into RGB, depth, and mask reconstructions as well as their position estimates (rows). Only occupied slots (columns) and the background module output (most right column) are shown.

3.6. Segmentation Preprocessing

In the process of pre-training the encoder-decoder architecture on discrete object instances, the Loci-s network acquires a foundational ability to en- and decode objects. Learning is furthermore supported by the pre-trained background module, which distinguishes foreground entities from the background context. The remaining challenge lies in the accurate identification and allocation of objects into distinct slots. SAVi++ provides ground-truth bounding box information to accomplish this step (Elsayed et al.). Our work explores three methodologies for the initial assignment of slots, without relying on ground-truth information.

First, we utilize a stochastic positioning strategy within the foreground mask that is generated by the Uncertainty Network. In particular, we sample a location uniformly randomly within the generated uncertainty map, which essentially predicts object masks. We initialize an empty slot with this location encoded as a 2D Gaussian position Q_k^t, similar to the pretraining of the object-specific encodings and decodings specified above.

This first approach, however, is susceptible to the erroneous partitioning of larger objects, because multiple random positions may be selected within the same object. To miti-

gate this, our second approach—termed "Regularized Initial Slots"—retains the random sampling paradigm during a warm-up phase. Following each network pass, though, we compute a similarity metric for each slot pair, based on both the Euclidean distance between their positional codes and the correlation of their Gestalt codes. Slots exhibiting a similarity below a predefined threshold are nullified in a stochastic manner.

The third approach employs a specialized segmentation network akin to YOLACT (Bolya et al., 2019), which was trained supervised using a Cross-Entropy loss comparing predicted instance masks with the best matching ground truth once. For more details see appendix (cf. Table 7 and Table 3). Initial slot positions are calculated based on the instance masks outputted by this network.

4. Experiments & Results

In our experimentation pipeline, we initially pretrain our models on the Kubric MOVi-(a-f) dataset (Greff et al., 2022). Subsequently, we employ two distinct strategies: (1) full model training on MOVi-(a-e) for benchmarking against SAVi++ (Elsayed et al.), and (2) fine-tuning the encoder, decoder, and background modules on the datasets delineated

Table 1. Loci-s demonstrates largely superior performance in the MOVi Challenge, benchmarked against SAVi++ (Elsayed et al.). The results show that segmentation performance critically depends on the strategy for initial slot assignment. Note that SAVI++ provides ground-truth masks to each slot in the first frame. Loci-s, on the other hand, uses our novel segmentation preprocessing strategy. A segmentation network-informed slot assignment (seg) with depth information as input (Loci-s_d) yields the best score. Random slot assignments given the uncertainty map from the Uncertainty Module (rnd) clearly show the importance to start with good slot assignments, yielding performance worse than SAVi++ but still better than SAVi in MOVi-D and MOVi-E. Employing the regularized initial slot technique (reg) during the initialization phase yields intermediate outcomes. The smaller standard deviation ($\pm$) of our results also hints at a more reproducible training and evaluation of Loci-s than SAVI++ and especially SAVI.

	mIoU↑ (%)			FG-ARI↑ (%)		
Model	MOVi-C	MOVi-D	MOVi-E	MOVi-C	MOVi-D	MOVi-E
CRW	27.8 ± 0.2	45.3 ± 0.0	47.5 ± 0.1	*	*	*
SAVi	43.1 ± 0.7	22.7 ± 7.5	30.7 ± 4.9	77.6 ± 0.7	59.6 ± 6.7	55.3 ± 5.8
SAVi++	45.2 ± 0.1	48.3 ± 0.5	47.1 ± 1.3	$\mathbf{81.9 \pm 0.2}$	$\mathbf{86.0 \pm 0.3}$	84.1 ± 0.9
Loci-s_d (rnd)	40.3 ± 0.1	40.9 ± 0.3	44.4 ± 0.2	58.5 ± 0.6	49.8 ± 0.2	60.8 ± 0.2
Loci-s_d (reg)	45.7 ± 0.2	47.9 ± 0.3	49.2 ± 0.3	74.1 ± 0.3	74.0 ± 0.9	81.3 ± 0.8
Loci-s_d (seg)	$\mathbf{45.5 \pm 0.1}$	$\mathbf{51.8 \pm 0.1}$	$\mathbf{53.5 \pm 0.1}$	79.2 ± 0.3	81.1 ± 0.2	$\mathbf{88.5 \pm 0.2}$
Loci-s (rnd)	33.4 ± 0.3	38.8 ± 0.2	41.2 ± 0.2	60.4 ± 0.5	54.3 ± 0.3	63.9 ± 0.6
Loci-s (reg)	35.7 ± 0.2	41.2 ± 0.2	42.4 ± 0.1	68.1 ± 0.5	71.4 ± 0.7	78.3 ± 0.1
Loci-s (seg)	36.2 ± 0.1	44.9 ± 0.1	47.0 ± 0.1	72.7 ± 0.3	79.5 ± 0.2	85.1 ± 0.0

in the Computational Scene Representation Review (Yuan et al., 2023) prior to training the full Loci-s system on these datasets.

For video dataset training, we employ a warm-up phase comprising three iterations, during which only the encoder and decoder are updated, omitting the predictor. This warm-up occurs on the initial frame. We utilize truncated backpropagation through time (BPTT) with a sequence length of 2 for sequence-based learning. In contrast, for image datasets, we omit the warm-up phase and iteratively forward and backward propagate the same image for three cycles.

In the inference phase, we extend the warm-up iterations to 10 for both video and image data, which showed an empirical improvement in performance. Additionally, in image-centric tasks, we augment the number of full-architecture iterations to 10, totaling 20 processing steps: 10 for encoder/decoder-only warm-up and 10 for full architecture evaluation.

4.1. Video Evaluation

The design of Loci-s primarily centers around temporal predictions, hence a detailed comparative study is performed against SAVi++ on the MOVi-(c-e) dataset. To maintain evaluative consistency, we adhere to the same performance measures as outlined in the SAVi++ paper, namely the Per-Sequence Intersection over Union (IoU) and the Per-Sequence Foreground Adjusted Rand Index (FG-ARI).

As illustrated in Table 1, Loci-s manifests a notable performance uplift, registering a 13.59% relative IoU improvement on the most demanding MOVi-E dataset, elevating the

score from 47.1% (attained by SAVi++) to 53.5% (Loci-s with depth input and segmentation preprocessing). The FG-ARI results are more mixed with Loci-s achieving superior performance in the most challenging MOVi-E dataset while achieving slightly lower scores than SAVi++ on MOVi-C and MOVi-D.

However, an important consideration in interpreting the discrepancy between IoU and FG-ARI scores involves recognizing the inherent nature of these metrics, particularly in the scope of temporal segmentation. IoU, in this context, can be interpreted as a 'worst-case' metric. It fundamentally penalizes any misalignment between predicted and ground truth instances, inherently emphasizing the lower bound of segmentation accuracy. This stringent criterion means that even minor deviations in spatial alignment or instance misidentification across the sequence are heavily penalized, making high IoU scores indicative of exceptional spatial and temporal precision in segmentation.

Conversely, the FG-ARI, by design, is a metric that emphasizes the consistency and correctness of instance tracking over time, offering a more nuanced perspective on temporal segmentation. It quantifies the accuracy of instance groupings while allowing some leeway for minor errors in boundary delineation or slight temporal shifts. This characteristic of FG-ARI implies that, despite a lower score in this metric, Loci-s may still effectively capture the overall dynamics and structure of the scene over time while performing slightly worse in object tracking than SAVi++ on MOVi-C and MOVi-D.

In light of these architectural design choices, Loci has demonstrated stable slot activations over extended tempo-

Table 2. Comparative results of Loci-s across six datasets for compositional scene understanding (moving MNIST digits and dSprites, the Abstract Scene dataset, CLEVR, SHOP VRB, and a combination of GSO and HDRI-Haven) (Yuan et al., 2023). The table shows in-distribution performance (3-6 objects) and out-of-distribution generalization (7-10 objects). Models were pretrained on MOVi-(a-f), fine-tuned on all datasets, and evaluated using the epoch with lowest validation error. Generalization capacity was assessed by increasing the maximum number of slots to 10 without further training. Loci-s largely outperforms all other reported approaches.

	AMI-A	ARI-A	AMI-O	ARI-O	IoU	F1	OCA	OOA
Test 1 Validation (3-6 Objects)								
AIR	0.380	0.397	0.845	0.827	N/A	N/A	0.549	0.709
N-EM	0.208	0.233	0.341	0.282	N/A	N/A	0.013	N/A
IODINE	0.638	0.700	0.772	0.752	N/A	N/A	0.487	N/A
GMIOO	0.738	0.811	**0.916**	0.914	0.708	0.808	**0.772**	**0.846**
MONet	0.657	0.699	0.863	0.857	N/A	N/A	0.663	0.583
GENESIS	0.411	0.412	0.420	0.382	0.105	0.170	0.213	0.603
SPACE	0.640	0.678	0.817	0.765	0.630	0.739	0.436	0.666
Slot Attention	0.393	0.321	0.758	0.711	N/A	N/A	0.028	N/A
EfficientMORL	0.341	0.279	0.673	0.621	N/A	N/A	0.107	N/A
GENESIS-V2	0.304	0.206	0.728	0.693	N/A	N/A	0.153	0.574
Loci-s (rnd)	0.835	0.918	0.735	0.891	0.742	0.822	0.421	-
Loci-s (reg)	0.838	0.921	0.730	0.898	0.731	0.810	0.443	-
Loci-s (seg)	**0.844**	**0.922**	0.748	**0.920**	**0.781**	**0.860**	0.505	-
Test 2 Generalization (7-10 Objects)								
AIR	0.410	0.402	0.802	0.740	N/A	N/A	0.327	0.689
N-EM	0.256	0.268	0.354	0.261	N/A	N/A	0.017	N/A
IODINE	0.633	0.652	0.781	0.731	N/A	N/A	0.387	N/A
GMIOO	0.732	0.781	**0.891**	0.868	0.647	0.746	**0.534**	**0.823**
MONet	0.635	0.665	0.820	0.785	N/A	N/A	0.446	0.619
GENESIS	0.380	0.378	0.415	0.315	0.076	0.132	0.160	0.584
SPACE	0.628	0.639	0.802	0.717	0.543	0.654	0.265	0.650
Slot Attention	0.447	0.330	0.761	0.696	N/A	N/A	0.029	N/A
EfficientMORL	0.366	0.236	0.662	0.562	N/A	N/A	0.085	N/A
GENESIS-V2	0.378	0.235	0.723	0.655	N/A	N/A	0.189	0.617
Loci-s (rnd)	0.820	0.866	0.766	0.875	0.667	0.755	0.228	-
Loci-s (reg)	0.828	**0.888**	0.768	0.865	0.637	0.724	0.244	-
Loci-s (seg)	**0.832**	0.877	0.783	**0.905**	**0.706**	**0.792**	0.315	-

ral windows (Traub et al., 2023b). However, the masks decoded from the predicted Gestalt codes M_k^{t+1} are less accurate than those derived directly from the encoder $\tilde{M}_k^t$. The encoder masks, on the other hand, suffer from low temporal consistency, since the information fusion of $\tilde{G}_k^t$ with G_k^{t-1} happens afterwards via the update gate and also inside the predictor itself (via the GateL0RD recurrences). At the moment the challenge remains to further improve the fusion of accurate mask reconstructions ($\tilde{G}_k^t$) with stable temporal predictions ($\tilde{G}_k^{t-1}$).

4.2. Image Evaluation

In a recent review paper about compositional scene understanding, Yuan et al. (2023) proposed a total of 6 datasets ranging in complexity from compositing MNIST to realistic texture simulations like MOVi-(c-e). These datasets are constructed in a way to perform two test: an in-distribution test with the same number of objects in a scene as seen during training (between 3 and 6); and another out-of-distribution test that probes generalization abilities with object numbers ranging from 7 to 10. In our experiments we used a pretrain (on MOVi-(a-f)) encoder-decoder network and fine-tuned it using all 6 datasets at once. We then further trained Loci-s on these combined 6 datasets and set the maximum number of slots to 6 during training. We then selected the model checkpoint form the epoch with the lowest validation error.

For the generalization test we simply increase the maximum number of slots without further training to 10. We test the following metrics, which were reported by (Yuan et al., 2023): Adjusted Mutual Information (AMI), Adjusted Rand Index (ARI), Intersection over Union (IoU), F1 score and Object Counting Accuracy (OCA). As shown in Table 2 Loci-s shows superior performance in most metrics for both the in-distribution test and the generalization test. Note
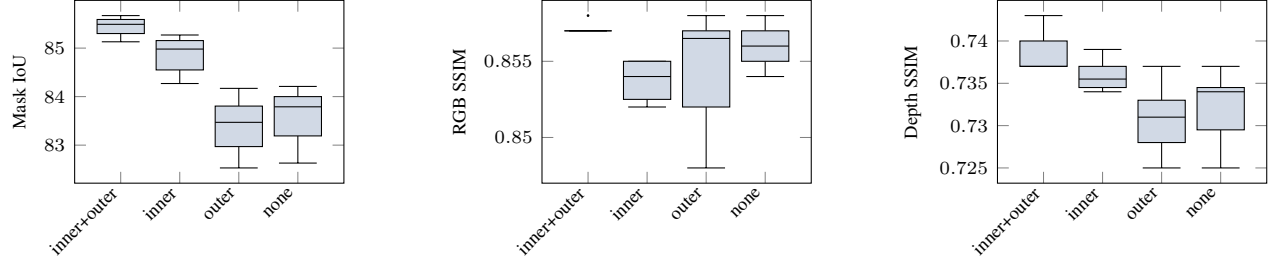
Figure 8. Ablating inner or outer feedback loop confirms their efficacy seeing improvements in per-object mask intersection-over-union (IoU) and depth structural similarity index measure (SSIM).

that we did not explicitly train or fine-tune Loci-s on each dataset individually, as don in Yuan et al. (2023), but rather trained them on all datasets combined, scaling individual resolutions up to 256×256 where necessary. We expect even better performance with a dataset-specific fine-tuning of model and hyper parameters.

4.3. Top Down Feedback Ablations

In Figure 8, we conduct a further ablation study to investigate the impact of top-down feedback in our architecture. We restrict our evaluation to pre-trained encoder-decoder networks, motivated by their substantially lower computational cost. Indeed, these networks are trainable using a single GTX 1080 GPU with a single slot for pre-training. Each experimental configuration is executed five times, utilizing a consistent set of five random seeds for reproducibility. Our ablation compares four scenarios evaluating the inner and outer top-down feedback loops: the inner feedback controls the hyper-network tuning the spatial convolutional encoder kernels top-down using $\hat{G}_k^t$; the outer feedback refers to the slot specific inputs $\hat{R}_k^t$, $\hat{D}_k^t$, $M_k^{t,o}$, $\hat{M}_k^{t,v}$ and $\hat{M}_k^{t,s}$. We compare the performance of (i) the proposed architecture with both inner and outer feedback loops, (ii) a version with only outer feedback, (iii) a version with only inner feedback, and (iv) a baseline with no feedback mechanisms.

The results in Figure 8 demonstrate that the inclusion of top-down feedback is particularly advantageous for mask prediction tasks, resulting in a significant improvement in the Intersection-over-Union (IoU) metric. While the benefits for the Structural Similarity Index Measure (SSIM) in RGB reconstruction are less consistent, the depth reconstruction task also profits from the presence of top-down feedback information.

5. Conclusion

The Loci-s model introduces several key innovations in the domain of scene understanding and object segmentation. A novel background reconstruction and foreground density estimation approach greatly facilitates object-oriented scene

segmentations without relying on ground-truth slot initialization. Moreover, dynamic convolution kernels via a hyper-network-controlled top-down residual network facilitates object-specific visual encoding. Finally, the incorporation of depth information additionally facilitates segmentation performance event further. These advancements collectively contribute to a 13.59% relative improvement in IoU on the challenging MOVi-E dataset compared to state-of-the-art models like SAVi++. Still, Loci-s falls short in some performance metrics, particularly in FG-ARI in the MOVi-C and MOVi-D datasets when compared to SAVi++. This suggests that while Loci-s excels in segmenting complex environments, it still struggles with fully accurate temporal object trackings. We suspect that this can be attributed to the fact that Loci-s fully compresses past video frame information in the internal recurrent state of its Transition Module. Our results demonstrate robustness in both in-distribution and out-of-distribution tests, highlighting the model's generalization abilities. However, it remains an open question whether this robustness extends to more varied or even more dynamic environments, and how it fares against models optimized for such scenarios. Furthermore, Loci-s shows great potential in terms of interpretability of deep learning systems, as shown in Figure 1 (further examples can be found in the appendix and supplementary video material).

Taking inspiration from human cognition, from the binding problem, and from recent computational and conceptual insights into our modularized minds (Greff et al., 2020; Mattar & Lengyel, 2022; Heald et al., 2023; Butz et al., 2021; Schwöbel et al., 2021), the background processing module may yet be enhanced to a universal background extraction module relative to which foreground objects may be extracted. Furthermore, optimally distributing cognitive processing resources onto currently task-relevant objects and interactions remains as an important challenge. We believe that segmentation-oriented algorithms, such as Loci-s, constitute one crucial foundation-model-like module that offers itself to be effectively combined with (i) reinforcement learning, planning, and reasoning approaches and (ii) language processing modules in future work.

6. Acknowledgement

References

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157–9166, 2019.

Butz, M. V., Achimova, A., Bilkey, D., and Knott, A. Event-predictive cognition: A root for conceptual human thought. *Topics in Cognitive Science*, 13:10–24, 2021. doi: 10.1111/tops.12522.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Elsayed, G., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M. C., and Kipf, T. Savi++: Towards end-to-end object-centric learning from real-world videos. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 28940–28954. Curran Associates, Inc.

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

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3749–3761, 2022.

Gumbsch, C., Butz, M. V., and Martius, G. Sparsely changing latent states for prediction and planning in partially observable domains. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17518–17531. Curran Associates, Inc., 2021. URL https://arxiv.org/abs/2110.15949.

Ha, D. and Schmidhuber, J. World Models. March 2018. 10.5281/zenodo.1207631.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2555–2565. PMLR, May 2019. ISSN: 2640-3498.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to Control: Learning Behaviors by Latent Imagination, March 2020. 10.48550/arXiv.1912.01603.

Heald, J. B., Lengyel, M., and Wolpert, D. M. Contextual inference in learning and memory. *Trends in Cognitive Sciences*, 27(1):43–64, 2023. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2022.10.004. URL https://www.sciencedirect.com/science/article/pii/S1364661322002650.

Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. Conditional object-centric learning from video. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=aD7uesX1GF_.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Mattar, M. G. and Lengyel, M. Planning in the brain. *Neuron*, 110(6):914–934, 2022. ISSN 0896-6273. doi: 10.1016/j.neuron.2021.12.018. URL https://www.sciencedirect.com/science/article/pii/S0896627321010357.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *Advances in neural information processing systems*, 30, 2017.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4.

Schwöbel, S., Marković, D., Smolka, M. N., and Kiebel, S. J. Balancing control: A bayesian interpretation of habitual and goal-directed behavior. *Journal of Mathematical Psychology*, 100:102472, 2021. ISSN 0022-2496. doi: 10.1016/j.jmp.2020.102472. URL https://www.sciencedirect.com/science/article/pii/S0022249620301000.

Traub, M., Becker, F., Otte, S., and Butz, M. V. Looping loci: Developing object permanence from videos. *arXiv preprint arXiv:2310.10372*, 2023a.

Traub, M., Otte, S., Menge, T., Karlbauer, M., Thuemmel, J., and Butz, M. V. Learning what and where: Disentangling location and identity tracking without supervision. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=NeDc-Ak-H_.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, Z., Dvornik, N., Greff, K., Kipf, T., and Garg, A. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TFbwV6I0VLg.

Yuan, J., Chen, T., Li, B., and Xue, X. Compositional scene representation learning via reconstruction: A survey, 2023.

A. Appendix

A.1. Closing the Inner Loop

We enhance Loci's object tracking abilities similar to (Traub et al., 2023a), which draws inspiration from Kalman filtering (Kalman, 1960). Originally, Loci predicts the next object states via a pixel space-routed outer loop (see Figure 2; outer loop). We draw inspiration from work in model-based reinforcement learning, which has recently advocated latent world model predictions (Hafner et al., 2019; 2020; Ha & Schmidhuber, 2018; Schrittwieser et al., 2020). These allow the imagination of future scene dynamics via an inner loop, without explicit pixel-based generations. Similarly, we apply an inner processing loop in Loci-s. (see Figure 2; inner loop).

In accordance with Kalman filtering, Loci-s is enabled to linearly interpolate between the current sensor information and its predictions. Formally, the current object states $S_k^t =$

(G_k^t, P_k^t) become a linear blending of the observed object states $\tilde{G}_k^t, \tilde{P}_k^t$ and the predicted object states $\hat{G}_k^t, \hat{P}_k^t$:

$$G_k^t = \alpha_k^{t,G}\tilde{G}_k^t + (1 - \alpha_k^{t,G})\hat{G}_k^t \tag{1}$$

$$P_k^t = \alpha_k^{t,P}\tilde{P}_k^t + (1 - \alpha_k^{t,P})\hat{P}_k^t \tag{2}$$

The weighting α is specific for each Gestalt and position code in each slot k. Importantly, Loci-s learns to regulate this percept gate on its own in a fully self-supervised manner. It learns an update function g_θ, which takes as input the observed state $\tilde{S}_k^t$, the predicted state $\hat{S}_k^t$, and the last positional encoding P_k^{t-1}:

$$(z_k^{t,G}, z_k^{t,P}) = g_\theta(\tilde{S}_k^t, \hat{S}_k^t, P_k^{t-1}) + \varepsilon \qquad \text{with} \quad \varepsilon \sim \mathcal{N}(0, \Sigma), \tag{3}$$

We model g_θ with a feed-forward network. To be able to fully rely on its own predictions, Loci-s needs to be able to fully close the gate by setting α exactly to zero. We therefore use a rectified hyperbolic tangent to compute α:

$$(\alpha_k^{t,G}, \alpha_k^{t,P}) = \max(0, \tanh((z_k^{t,G}, z_k^{t,P}))). \tag{4}$$

An L_0 loss on gate openings encourages the reliance on internal beliefs rather than external updates.

B. Depth Input Normalization

We log-normalized the Scene-Relative Depth, according to Equation 5:

$$d = \frac{1}{1 + \exp\left(\frac{\hat{d} - \mu}{\sigma}\right)}, \tag{5}$$

where $\hat{d}$ represents the natural logarithm of the raw depth values. Parameters μ and σ denote the mean and standard deviation of the log-transformed depth, respectively.

C. Additional tables and figures

D. Detailed Loci-s Size and Wiring Information

Table 3. Performance of our segmentation prepossessing network. While achieving adequate performance on evaluation datasets, the preprocessing network clearly fails on the generalization dataset.

mIoU↑ (%)				
MOVi-C	MOVi-D	MOVi-E	Review datasets Test 1	Review datasets Test 2
88.39 ± 0.03	82.48 ± 0.05	80.89 ± 0.07	90.98 ± 0.12	67.31 ± 0.05

Figure 9. The slotwise decomposition of the input from Figure 1 into unmasked rgb and depth reconstructions per slot and the background rgb and depth reconstruction. For simplicity we only show occupied slots.
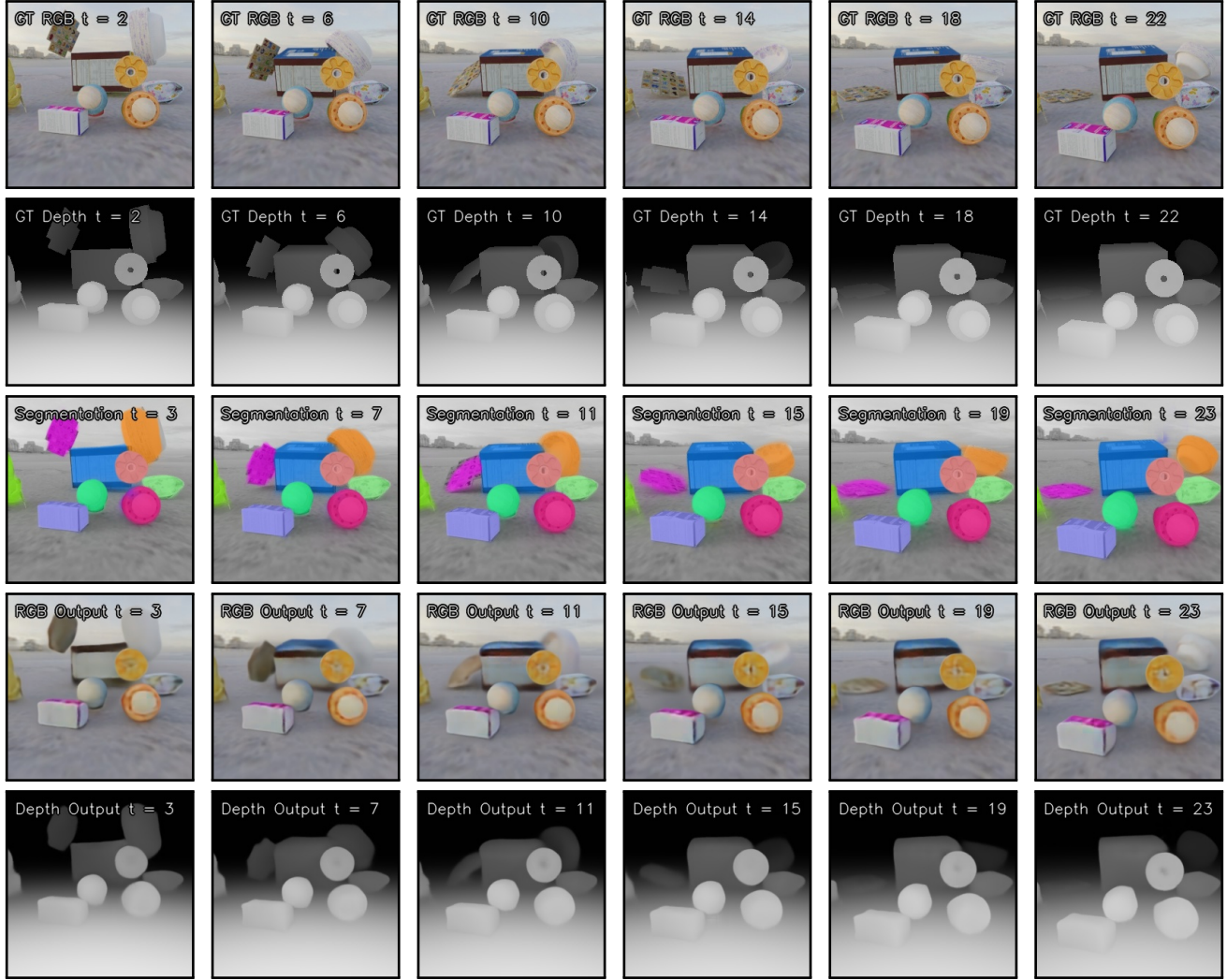
Figure 10. Qualitative Analysis of Results on the MOVi-E Dataset: The top two rows show the input frames while the third row shows the target frame superimposed with the slot masks, and the last two rows show the next frame predictions.

Table 4. Encoder Architecture, for more information see the source file in *nn/hyper_encoder.py*

Component	Layer	Configuration
Encoder Base	ResidualPatchEmbedding	Conv2D(16, 32, 4, 4) + AvgPool + Channel Copy
	HyperConvNext	$32 \rightarrow 32$
	ResidualPatchEmbedding	Conv2D(32, 64, 2, 2) + AvgPool + Channel Copy
	HyperConvNext	$64 \rightarrow 64$
	ResidualPatchEmbedding	Conv2D(64, 128, 2, 2) + AvgPool + Channel Copy
	HyperConvNext	$128 \rightarrow 128$
	HyperConvNext	$128 \rightarrow 128$
	HyperConvNext	$128 \rightarrow 128$
Position Encoder	HyperConvNext	$128 \rightarrow 128$
	HyperConvNext	$128 \rightarrow 128$
	HyperConvNext	$128 \rightarrow 4$
	FeaturesMapToPosition	
Gestalt Base Encoder	HyperConvNext	$128 \rightarrow 128$
	HyperConvNext	$128 \rightarrow 128$
Mask Gestalt Encoder	ResidualPatchEmbedding	Conv2D(128, 256, 2, 2) + AvgPool + Channel Copy
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	PositionPooling	
Depth Gestalt Encoder	ResidualPatchEmbedding	Conv2D(128, 256, 2, 2) + AvgPool + Channel Copy
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	PositionPooling	
RGB Gestalt Encoder	ResidualPatchEmbedding	Conv2D(128, 256, 2, 2) + AvgPool + Channel Copy
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	HyperConvNext	$256 \rightarrow 256$
	PositionPooling	

Table 5. Predictor Architecture, for more information see the source file in *nn/predictor.py*

Component	Layer	Configuration
UpdateController	Linear	$1550 \rightarrow 256$
	SiLU	
	Linear	$256 \rightarrow 256$
	SiLU	
	Linear	$256 \rightarrow 2$
Predictor	InputEmbedding	
	Linear	$774 \rightarrow 1024$
	SiLU	
	Linear	$1024 \rightarrow 1024$
	GateL0rd	$1024 \rightarrow 1024$
	MultiheadSelfAttention	$1024 \rightarrow 1204$
	GateL0rd	$1024 \rightarrow 1024$
	MultiheadSelfAttention	$1024 \rightarrow 1204$
	GateL0rd	$1024 \rightarrow 1024$
	MultiheadSelfAttention	$1024 \rightarrow 1204$
	GateL0rd	$1024 \rightarrow 1024$
	MultiheadSelfAttention	$1024 \rightarrow 1204$
	GateL0rd	$1024 \rightarrow 1024$
	OutputEmbedding	
	Linear	$1024 \rightarrow 1024$
	SiLU	
	Linear	$1024 \rightarrow 774$

Table 6. Decoder Architecture, for more information see the source file in *nn/decoder.py*

Component	Layer	Configuration
MaskDecoder	GestaltPositionFussion	
	Conv2d	256 → 128, kernel=3, pad=1
	SiLU	
	Conv2d	128 → 64, kernel=3, pad=1
	SiLU	
	Conv2d	64 → 32, kernel=3, pad=1
	SiLU	
	Conv2d	32 → 128, kernel=1
	TransposedConv2d	128 → 1, kernel=16, stride=16
DepthDecoder	MaskEncoder	
	Conv2d	1 → 128, kernel=16, stride=16
	SiLU	
	Conv2d	128 → 32, kernel=1
	GestaltMaskFussion	Gestalt * MaxPool(mask, kernel=16)
	Concat	ModulatedGestalt, EncodedMask, PositionalEmbedding
	Conv2d	304 → 64, kernel=1
	ConvNeXt	64 → 64
	ConvNeXt	64 → 64
	ConvNeXt	64 → 64
	Conv2d	64 → 256, kernel=1
	SiLU	
	TransposedConv2d	256 → 1, kernel=16, stride=16
RGBDecoder	MaskEncoder	
	Conv2d	1 → 128, kernel=16, stride=16
	SiLU	
	Conv2d	128 → 32, kernel=1
	DepthEncoder	
	Conv2d	1 → 256, kernel=16, stride=16
	SiLU	
	Conv2d	256 → 64, kernel=1
	GestaltMaskFussion	Gestalt * MaxPool(mask, 16)
	Concat	ModulatedGestalt, EncodedMask, EncodedDepth, PositionalEmbedding
	Conv2d	368 → 128, kernel=1
	ConvNeXt	128 → 128
	ConvNeXt	128 → 128
	ConvNeXt	128 → 128
	ConvNeXt	128 → 128
	ConvNeXt	128 → 128
	Conv2d	128 → 512, kernel=1
	SiLU	
	TransposedConv2d	512 → 3, kernel=16, stride=16

Table 7. Segmentation preprocessing, see *nn/proposal_v2.py* for more details

Component	Layer	Configuration
	Cat	Depth + 2D Grid(-1,1)
	ResidualPatchEmbedding	Conv2D(3, 64, 4, 4) + AvgPool + Channel Copy
	ConvNeXt	$64 \rightarrow 64$
	ResidualPatchEmbedding	Conv2D(64, 128, 2, 2) + AvgPool + Channel Copy
	ConvNeXt	$128 \rightarrow 128$
	ConvNeXt	$128 \rightarrow 128$
	ResidualPatchEmbedding	Conv2D(128, 256, 2, 2) + AvgPool + Channel Copy
	ConvNeXt	$256 \rightarrow 256$
	ConvNeXt	$256 \rightarrow 256$
	ConvNeXt	$256 \rightarrow 256$
	ResidualPatchEmbedding	Conv2D(256, 512, 2, 2) + AvgPool + Channel Copy
	ConvNeXt	$256 \rightarrow 256$
	HyperNetwork	
	GlobalAvgPool	
	Linear	$512 \rightarrow 512$
	SiLU	
	Linear	$512 \rightarrow 512$
	SiLU	
	Linear	$512 \rightarrow 512$
	ConvNeXt	$512 \rightarrow 512$
SegmentationUNet	Conv2d	$512 \rightarrow 2048$, kernel=1
	SiLU	
	Conv2d	$2048 \rightarrow 256$, kernel=2, stride=2
	ConcatFeatures	
	Conv2d	$512 \rightarrow 256$, kernel=1
	ConvNeXt	$256 \rightarrow 256$
	Conv2d	$256 \rightarrow 1024$, kernel=1
	SiLU	
	Conv2d	$1024 \rightarrow 128$, kernel=2, stride=2
	ConcatFeatures	
	Conv2d	$256 \rightarrow 128$, kernel=1
	ConvNeXt	$128 \rightarrow 128$
	Conv2d	$128 \rightarrow 512$, kernel=1
	SiLU	
	Conv2d	$512 \rightarrow 64$, 2, stride=2
	ConcatFeatures	
	Conv2d	$128 \rightarrow 64$, kernel=1
	ConvNeXt	$64 \rightarrow 64$
	Conv2d	$64 \rightarrow 512$, kernel=1
	SiLU	
	Conv2d	$512 \rightarrow 32$, kernel=4, stride=4
	ApplyHyperWeights	Features @ weights, 32, 16

Table 8. Background Architecture, for more information see the source file in *nn/decoder.py*

Component	Layer	Configuration
Uncertainty UNet	MaskEncoder	
	ResidualPatchEmbedding	Conv2D(4, 16, 4, 4) + AvgPool + Channel Copy
	ConvNeXt	$16 \rightarrow 16$
	ResidualPatchEmbedding	Conv2D(16, 32, 2, 2) + AvgPool + Channel Copy
	ConvNeXt	$32 \rightarrow 32$
	ResidualPatchEmbedding	Conv2D(32, 64, 2, 2) + AvgPool + Channel Copy
	ConvNeXt	$64 \rightarrow 64$
	ResidualPatchEmbedding	Conv2D(64, 128, 2, 2) + AvgPool + Channel Copy
	ConvNeXt	$128 \rightarrow 128$
	ConvNeXt	$128 \rightarrow 128$
	ResidualPatchUpscaling	Conv2D(128, 64, 2, 2) + Upscale + Channel Avg
	ConvNeXt	$64 \rightarrow 64$
	ResidualPatchUpscaling	Conv2D(64, 32, 2, 2) + Upscale + Channel Avg
	ConvNeXt	$32 \rightarrow 32$
	ResidualPatchUpscaling	Conv2D(32, 16, 2, 2) + Upscale + Channel Avg
	ConvNeXt	$16 \rightarrow 16$
	ResidualPatchUpscaling	Conv2D(16, 1, 4, 4) + Upscale + Channel Avg
Background Extractor Base-Encoder	PatchEmbedding	
	Conv2d	$4 \rightarrow 256$, kernel=16, stride=16
	SiLU	
	Conv2d	$256 \rightarrow 64$, kernel=1
	MHA-Layer	$64 \rightarrow 64$
	MHA-Layer	$64 \rightarrow 64$
Background Extractor RGB-Encoder	MHA-Layer	$64 \rightarrow 64$
Background Extractor Depth-Encoder	MHA-Layer	$64 \rightarrow 64$
	Bottleneck	token avg + cross attention to single token
	Sigmoid	
	Binarize	$x \leftarrow x + x(1-x)\mathcal{N}(0,1)$
Background Extractor Depth-Decoder	ConvNeXt	$64 \rightarrow 64$
	ConvNeXt	$64 \rightarrow 64$
	PatchUpscaling	
	Conv2d	$64 \rightarrow 256$, kernel=1
	SiLU	
	TransposedConv2d	$256 \rightarrow 1$, kernel=16, stride=16
Background Extractor RGB-Decoder	Depth-Encoder	
	PatchEmbedding	$1 \rightarrow 64$
	ConvNeXt	$64 \rightarrow 64$
	Cross-Attention-Layer	$64 \rightarrow 64$
	ConvNeXt	$64 \rightarrow 64$
	ConvNeXt	$64 \rightarrow 64$
	PatchUpscaling	
	Conv2d	$64 \rightarrow 256$, kernel=1
	SiLU	
	TransposedConv2d	$256 \rightarrow 1$, kernel=16, stride=16