Efficient Object Detection and Discovery for Real-World Robotics Applications



Martin Engelcke

Somerville College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2020

To my parents

Acknowledgements

Supervisor First and foremost, I would like to thank my supervisor Prof. Ingmar Posner for his support, guidance, advice, mentorship, and innumerable contributions to this thesis. This thesis would not exist without Ingmar and I am grateful for the many things I learned while working together.

Collaborators Next, I would like to thank my collaborators whom I had the pleasure of working with and who contributed to the materials presented in this thesis:

- Vote3Deep: Dushyant Rao, Dominic Zeng Wang, and Chi Hay Tong;
- GENESIS: Adam R. Kosiorek and Oiwi Parker Jones;
- GENESIS++: Oiwi Parker Jones.

Oxford Robotics Institute (ORI) I would like to thank the other DPhil students, postdocs, and PIs at the ORI who provided informal input during casual conversations, reading groups, and otherwise. Particular mention deserve Yizhe Wu for his help with re-creating the Multi-dSprites dataset and re-implementing MONET as well as Oliver Groth for his help with the ShapeStacks and GQN datasets. Similarly, I would like to thank the engineers who helped with technical issues and the support staff who helped with a range of tasks.

DPhil Funding I am grateful for the funding of my DPhil via an EPSRC DTA Studentship and a Google Studentship.

Computing Resources I would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work (http://dx.doi.org/10.5281/zenodo.22558) and the use of Hartree Centre Resources via JADE HPC UK. Moreover, I would like to thank the technical support teams at both ARC and Hartree for their help with various issues.

Reviewers I would like to thank the reviewers who provided constructive feedback on the publications that resulted from the work conducted for this thesis.

College Adviser I would like to thank my college adviser Prof. Richard Stone at Somerville. Richard has been incredibly helpful and supportive in a variety of matters and provided valuable guidance and advice.

Examiners I would like to express my gratitude to Prof. Andrea Vedaldi from the University of Oxford and Prof. Raquel Urtasun from the University of Toronto for taking the time to examine this thesis and for providing insightful feedback.

Family and Friends Finally, I would like to thank my family and friends for their support, in particular my parents whom this thesis is dedicated to.

Abstract

Robots need to interact with the world to complete practical tasks. This typically requires the interaction with objects, rendering object perception a fundamental capability in robotics. Real-world robotics applications provide particular challenges and opportunities for object perception. It is possible to utilise data from a variety of sensing modalities, such as cameras and lidar devices, but physical hardware constraints demand careful management of computational resources. Similarly, the need to respond to changing environmental conditions and task definitions requires robots to learn efficiently from limited human supervision. Efficiency in terms of computation and supervision are thus essential. To these ends, this thesis makes several contributions. Firstly, we develop Vote3Deep which is—to the best of our knowledge—the first method that performs efficient, native 3D object detection in real-world lidar point clouds with convolutional neural networks (CNNs). This is achieved by maintaining the natural sparsity inherent in the data throughout the CNN hierarchy and by leveraging an efficient, sparse convolution operation. Secondly, we present GENESIS which is—to the best of our knowledge—the first object-centric generative model that learns to decompose rendered images of simulated 3D environments into object-like components without supervision, while also being able to generate entire coherent scenes in an object-centric fashion. This is facilitated by an autoregressive prior that captures correlations between objects in the generative model, offering the potential of utilising GENESIS as an object-centric "world model" for efficient skill acquisition. Given that we want to apply object-centric generative models in real-world applications, we finally develop GENESIS++ which utilises a novel algorithm for fully-differentiable clustering of instance embeddings to facilitate the symmetric inference of object representations. We show that GENESIS++ outperforms competitive baselines both on well-established simulated datasets and on challenging real-world datasets collected in the context of robotics applications.

Contents

Acronyms **viii**

1 Introduction **1**
1.1 Contributions 5
1.2 Publications 7

2 Native 3D Object Detection in Real-World Lidar Point Clouds with Efficient Spatially Sparse CNNs **10**
2.1 Related Work 14
2.2 Vote3Deep 16
2.2.1 Sparse Convolutions via Voting 18
2.2.2 Maintaining Sparsity with Rectifiers 20
2.2.3 Hinge-Loss for Output Sparsity 21
2.2.4 Sparsity-Inducing Regularisation 22
2.3 Experiments 23
2.3.1 Training Details 24
2.3.2 Model Comparison 26
2.3.3 Representation Sparsity 30
2.3.4 KITTI Benchmark Results 33
2.3.5 Detection Speed 35
2.4 Conclusions 37
2.4.1 Limitations 37
2.4.2 Future Work 38

3 Unsupervised Object Discovery and Object-Centric World Modelling with Autoregressive Priors **40**
3.1 Related Work 44
3.2 GENESIS 47
3.2.1 Generative Model 47
3.2.2 Inference Model 50
3.2.3 Learning 51
3.3 Experiments 54

3.3.1 Datasets . . . 54
3.3.2 Evaluation Metrics . . . 58
3.3.3 Architecture Details . . . 60
3.3.4 Training Details . . . 64
3.3.5 Unsupervised Object Discovery . . . 66
3.3.6 Object-Centric Scene Generation . . . 71
3.3.7 Semi-Supervised Learning . . . 75
3.3.8 Reconstruction Bottlenecks . . . 77
3.4 Conclusions . . . 81
3.4.1 Limitations . . . 84
3.4.2 Future Work . . . 85

4 Scaling Object-Centric Generative Models to Real-World Robotics Datasets with Differentiable Instance Colouring 86
4.1 Related Work . . . 89
4.2 GENESIS++ . . . 92
4.2.1 Instance Colouring Stick-Breaking Process . . . 94
4.2.2 Semi-Convolutional Embeddings . . . 97
4.2.3 Autoregressive Prior with Greedy Sorting . . . 100
4.3 Experiments . . . 101
4.3.1 Datasets . . . 102
4.3.2 Implementation Details . . . 105
4.3.3 Benchmarking in Simulation . . . 109
4.3.4 Real-World Decomposition and Generation . . . 116
4.4 Conclusions . . . 136
4.4.1 Limitations . . . 138
4.4.2 Future Work . . . 138

5 Conclusions 140

References 144

Acronyms

ADAM Adaptive Moment Estimation.

AP Average Precision.

ARI Adjusted Rand Index.

BN Batch Normalisation.

CNN Convolutional Neural Network.

DC-VAE Deconvolutional VAE.

ELBO Evidence Lower Bound.

ELU Exponential Linear Unit.

FC Fully-Connected.

FID Fréchet Inception Distance.

FLOP Floating-Point Operation.

GAN Generative Adversarial Network.

GECO Generalised ELBO with Constrained Optimisation.

GENESIS Generative Scene Inference and Sampling.

GLU Gated Linear Unit.

GN Group Normalisation.

HNM Hard Negative Mining.

IC-SBP Instance Colouring Stick-Breaking Process.

IN Instance Normalisation.

IODINE Iterative Object Decomposition Inference Network.

IOU Intersection Over Union.

KL Kullback-Leibler.

LSTM Long Short-Term Memory.

MAP Mean Average Precision.

MC Monte Carlo.

MLP Multilayer Perceptron.

MONET Multi-Object Network.

MSC Mean Segmentation Covering.

NMS Non-Maximum Supression.

OCGM Object-Centric Generative Model.

PGM Probabilistic Graphical Model.

PR Precision-Recall.

RELU Rectified Linear Unit.

RL Reinforcement Learning.

RNN Recurrent Neural Network.

SBD Spatial Broadcast Decoder.

SBD-VAE Spatial Broadcast Decoder VAE.

SBP Stick-Breaking Process.

SGD Stochastic Gradient Descent.

SGMM Spatial Gaussian Mixture Model.

STN Spatial Transformer Network.

SVM Support Vector Machine.

VAE Variational Autoencoder.

1 Introduction

"Sense, interpret, plan, act; repeat" is the fundamental operating cycle in robotics. This thesis concerns itself with the second of these steps: interpretation. Robots are typically equipped with sensors to gather data from their environment, such as cameras and depth-sensing devices. Raw sensory data, however, is of limited utility for the development of robotic systems. For example, while a standard camera image is meaningful to a person when viewed on a display or printout, it is merely a tensor of integers to a computer. Yet, we would like robots to plan their actions in a higher-level semantic space, for example: "put water into the pot, put the pot onto the stove, turn on the stove". This requires the semantic interpretation of sensory data from a robot's environment. Typically, this involves the detection or discovery of objects in a scene and the identification of their attributes and physical locations relative to the robot. We also refer to objects more generically as "scene components". Interpreting spatial data, images in particular, is the domain of "computer vision". In the last decade, computer vision has made large strides through the use of "deep learning" (e.g. LeCun et al. [1]), in particular deep Convolutional Neural Networks (CNNs) (LeCun et al. [2]). This development has benefited from GPU-accelerated software libraries (e.g. Collobert et al. [3], Vedaldi and Lenc [4], Al-Rfou et al. [5], Abadi et al. [6], and Paszke et al. [7]) and large corpora of labelled data for specific tasks (e.g. Everingham et al. [8], Lin et al. [9], Russakovsky et al. [10], and

Cordts et al. [11]). Robotics, however, comes with a set of additional challenges and opportunities. On the one hand, it is possible to leverage information from different types of sensing modalities to improve system performance, for example lidar (Geiger et al. [12] and Maddern et al. [13]) and radar sensors (Barnes et al. [14]). On the other hand, robotics requires real-time performance under tighter physical hardware constraints while maintaining a high degree of adaptability to changing environmental conditions and task definitions. As such, we often need to train machine learning models with as little supervision as possible. In other words: efficiency in terms of computation and supervision is of the essence.

In comparison to other sensing modalities available in robotics applications, lidar sensors have the special property that they directly provide 3D spatial information by recording the time-of-flight of photons being reflected from surfaces in the line of sight, which is utilised particularly in the context of autonomous driving (e.g. Geiger et al. [12] and Maddern et al. [13]). While CNNs were quickly adopted for object detection in 2D images (e.g. Ren et al. [15], Liu et al. [16], and Redmon et al. [17]), this had not yet happened for detecting objects in large-scale, real-world lidar point clouds at the time when work on this thesis began. We attribute this lack of progress to the increased computational burden of processing three-dimensional data with dense convolutions. For example, Maturana and Scherer [18] use dense three-dimensional convolutions which renders the approach too slow for processing full point clouds at practical detection speeds. Li et al. [19] attempt to circumvent this by projecting point clouds into a two-dimensional representation which is passed into a CNN, but this projection limits the generality of the solution while also potentially causing a loss of information. In contrast to these works and based on the fact that point clouds are spatially sparse so that applying a convolution to a discretised point clouds results in most operations being non-informative multiplications by zero, Wang and Posner [20] instead implement a sparse convolution operation via "feature-centric voting" to convolve a Support Vector Machine (SVM) (Cortes and Vapnik [21]) in a sliding window fashion across a point cloud to predict detection scores for objects of interest. While Wang and

Posner [20] provide a method for processing point clouds with efficient sparse convolutions, this alone is not sufficient for applying multi-layer CNNs to real-world 3D lidar point clouds. After all, even though the input to the CNN is sparse, the same is not necessarily true for the intermediate and output representations. Furthermore, even if we have machinery in place that allows these representations to be sparse in principle, each convolutional layer can dilate the locations of non-zero feature vectors spatially, thus potentially reducing sparsity with every layer in the CNN stack. In Chapter 2, we therefore propose solutions to these challenges and develop a new method for object detection in 3D point clouds which we call Vote3Deep, in reference to Vote3D from Wang and Posner [20].

Albeit useful, supervised models for object detection such as Vote3Deep require large corpora of manually annotated examples for training (e.g. (Ren et al. [15], Liu et al. [16], and Redmon et al. [17])). This paradigm is expensive and restrictive as it is necessary to collect labels for every possible object category of interest. This is practically infeasible in real-world robotics applications given that the types of objects that a robot might encounter during its lifetime is often unknown, with the exception of very tightly constrained applications. Nevertheless, previous works have demonstrated that it is possible to discover object-like entities in images in an unsupervised fashion using cues such as colour, shape, and motion (Huang and Murphy [22], Eslami et al. [23], and Kosiorek et al. [24]). These approaches frame the discovery of objects as learning a structured Object-Centric Generative Model (OCGM). While the images in these early works were visually quite simple, Burgess et al. [25] and Greff et al. [26] recently further developed this approach to decompose rendered images of simulated 3D environments into object-like scene components without supervision. Both Burgess et al. [25] and Greff et al. [26] frame their proposed models as Variational Autoencoders (VAEs) (Kingma and Welling [27] and Rezende et al. [28]), which should in principle allow the generation of images that look similar to the training data by sampling from a probabilistic prior. While Burgess et al. [25] and Greff et al. [26] are able to decompose images into interpretable components, their underlying generative model assumes that scene

components are independent. Consequently, individual components sampled from the associated prior distribution do not necessarily constitute a coherent scene. For example, we know that two objects cannot occupy the same physical space and sampling objects independently might lead to physically impossible scenes. This renders these methods unsuitable for applications such as, for example, training Reinforcement Learning (RL) agents in learned generative "world models" (Ha and Schmidhuber [29]), which is a promising paradigm for supervision-efficient learning of robotic skills. In order to achieve this, it is necessary to have a mechanism that models correlations between scene components in the generative model similar as in van Steenkiste et al. [30]. However, van Steenkiste et al. [30] formulate their method as a Generative Adversarial Networks (GANs) (Goodfellow et al. [31]), which is able to generate images of comparable complexity as considered in Burgess et al. [25] and Greff et al. [26] in an object-centric fashion, but lacks the ability to decompose a given input image. Thus, we develop a new model called Generative Scene Inference and Sampling (GENESIS) in Chapter 3 that is able to both decompose images from rendered images of 3D environments and generate coherent visual scenes in a principled fashion.

Since the development of GENESIS, OCGMs have experienced an explosion of interest and a range of works have been developed on the subject, for example Lin et al. [32], Jiang et al. [33], Yang et al. [34], Bear et al. [35], Locatello et al. [36], Nguyen-Phuoc et al. [37], and Ehrhardt et al. [38]. Yet, learning object-like decompositions of more complex real-world images rather than synthetic images remains a challenge. Some of these methods employ comparatively strong inductive biases. For example, Jiang et al. [33] use temporal information to identify objects. Similarly, Nguyen-Phuoc et al. [37] and Ehrhardt et al. [38] design explicit geometric constraints into the model to facilitate interpretable scene decomposition of real-world data in constrained scenarios. Instead, we explore whether we can improve GENESIS to learn an object-centric world model of real-world data without the adoption of such constraints in Chapter 4. In particular, we observe that the sequential inference mechanism in GENESIS imposes an unnatural ordering

on scene components. This is undesirable from a computational perspective as sequential inference cannot be parallelised and different scene components potentially receive gradients of different magnitude, which might cause the model to collapse to non-interesting solutions where an image is reconstructed as a single component. In contrast, Greff et al. [26] and Locatello et al. [36] infer object representations in a symmetric fashion without imposing an ordering. Greff et al. [26], however, utilise a computationally expensive iterative inference scheme and the method in Locatello et al. [36] requires the maximum number of scene components to be specified *a priori*, in a fashion that is reminiscent of k-means clustering (e.g. Bishop [39]). Thus, in Chapter 4 we develop a new method for inferring symmetric representations and—in contrast to Greff et al. [26] and Locatello et al. [36]—we again also consider the task of generating entire coherent scenes by sampling from a suitable prior, leading to an improved model which we call GENESIS++.

There has been an explosion of interest in the nascent fields of 3D computer vision with deep learning and object-centric generative modelling while work on this thesis was conducted. This complicates the framing of our contributions to some degree. While individual chapters are written from the vantage point of the state of the art at the time when the respective materials were published (Section 1.2), the respective results are contextualised against more recent works at the end of Chapters 2 to 4.

1.1 Contributions

This thesis makes a number of contributions in robot perception, 3D computer vision, and object centric-generative modelling to facilitate computation- and supervision-efficient object detection and discovery in real-world robotics applications:

Vote3Deep (Chapter 2)

- To the best of our knowledge, we present the first work to construct efficient multi-layer CNNs using sparse convolutions and apply them to the problem of detecting objects in real-world lidar point clouds *natively in 3D*, outperforming prior art on the KITTI Vision Benchmark Suite (Geiger et al. [12]) at

competitive detection speeds and by considerable margins, in some cases by more than 30% in terms of absolute Average Precision (AP).

- This is achieved by utilising Rectified Linear Unit (RELU) (Glorot et al. [40]) non-linearities in combination with non-positive filter biases to maintain sparsity throughout the entire CNN feature hierarchy, allowing us to leverage an efficient sparse convolution operation implemented via feature-centric voting (Wang and Posner [20]) in every layer of the CNN stack.

- Furthermore, we propose to add an L_1 sparsity penalty on the post-RELU activations to the training objective to further increase sparsity in the intermediate representations. We demonstrate that the proposed penalty further improves detection speeds with minimal impact on prediction accuracy.

GENESIS (Chapter 3)

- To the best of our knowledge, we present the first unsupervised object-centric generative model of entire visual scenes that models correlations between scene components to both decompose rendered images from simulated 3D environments and generate full, coherent scenes in an object-centric fashion.

- This is facilitated by a novel autoregressive prior that models correlations between scene components in the generative process to ensure the coherence of generated scenes.

- In addition, a quantitative and qualitative analysis of "reconstruction bottlenecks" is conducted, showing that they play a crucial role in determining whether the model learns interpretable scene decompositions or whether it collapses to a single component during training.

GENESIS++ (Chapter 4)

- We present a further improved unsupervised object-centric generative model of entire visual scenes that achieves better quantitative and qualitative results on two simulated datasets with three-dimensional structure as well as several considerably more challenging real-world datasets collected in the context of robotic manipulation.

- To this end, we propose a kernel-based algorithm to perform fully-differentiable soft clustering of instance embeddings, allowing us to extract object representations without imposing an unnatural ordering on components as in GENESIS and without requiring the *a priori* specification of the maximum number of possible scene components.

- Moreover, we utilise semi-convolutional embeddings from Novotny et al. [41] to implement a weak bias for inducing spatial locality of attention masks. Based on this, we derive principled initialisations for the scaling factor in different distance kernels that can be used in the proposed instance clustering algorithm.

- Given an unordered, symmetric set of object representations, we finally propose a greedy sorting procedure to ease the optimisation of a sequential, autoregressive prior that models correlations between scene components and facilitates the generation of entire coherent scenes.

1.2 Publications

Work on this thesis has led to a number of peer-reviewed publications. In particular, three articles discuss some of the materials presented in this thesis:

- Chapter 2: M. Engelcke et al. "Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2017) [42];

- Chapter 3: M. Engelcke et al. "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *International Conference on Learning Representations (ICLR)* (2020) [43];
- Chapter 3: M. Engelcke et al. "Reconstruction Bottlenecks in Object-Centric Generative Models". In: *ICML Workshop on Object-Oriented Learning* (2020) [44].

The results presented in Chapter 4 are currently unpublished. In addition to these works, several other articles were published while the research presented in this thesis was conducted:

- L. Yi et al. "Large-Scale 3D Shape Reconstruction and Segmentation from ShapeNet Core55". In: *arXiv preprint arXiv:1710.06104* (2017) [45];
- B. Graham et al. "3D Semantic Segmentation with Submanifold Sparse Convolutional Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018) [46].
- E. Wagstaff et al. "On the Limitations of Representing Functions on Sets". In: *International Conference on Machine Learning (ICML)* (2019) [47];
- A. L. Mitchell et al. "First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020) [48];
- S. Ehrhardt et al. "RELATE: Physically Plausible Multi-Object Scene Synthesis Using Structured Latent Spaces". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2020) [38];

According to the 2020 Google Scholar Metrics, Engelcke et al. [42] is one of the 20 most cited papers published at the IEEE International Conference on Robotics and Automation (ICRA) within the last five years. As of early October 2020, this work has been cited more than 270 times according to Google scholar. Furthermore, the method in Engelcke et al. [42] was filed as a patent by the University of Oxford

in: M. Engelcke et al. *A Neural Network and Method of Using a Neural Network to Detect Objects in an Environment*. EP Patent App. 20170777642. 2019 [49]. The code for the experiments in Engelcke et al. [43, 44] has been made publicly available under an open-source license at `https://github.com/applied-ai-lab/genesis`. Work on Yi et al. [45] and Graham et al. [46] was conducted during an internship at Facebook AI Research (FAIR). Yi et al. [45] is a technical report from the ICCV 2017 Large-Scale 3D Shape Reconstruction and Segmentation challenge where we entered the winning submission to the 3D segmentation track.

2

Native 3D Object Detection in Real-World Lidar Point Clouds with Efficient Spatially Sparse CNNs

Lidar sensors are commonly used in robotics applications such as autonomous driving (e.g. Geiger et al. [12] and Maddern et al. [13]). Unlike cameras, lidar sensors directly provide three-dimensional spatial information in the form of 3D point clouds by recording the time-of-flight and intensity of reflected photons. A basic capability of interest in robotics applications is the ability to detect objects that are relevant for downstream planning tasks. Given an input frame, this is typically formulated as a supervised learning problem with the aim to predict a set of bounding boxes that each contain a single object (Li et al. [19], Wang and Posner [20], and Behley et al. [50]). In recent years, the widespread adoption of Convolutional Neural Networks (CNNs) (LeCun et al. [2]) has considerably advanced the state of the art in image-based object detection (e.g Ren et al. [15], Liu et al. [16], and Redmon et al. [17]). Yet, so far CNNs have been applied to real-world 3D lidar point clouds only with limited success. We attribute this to the increased computational burden when applying CNNs to higher dimensional data. While some solutions attempt to alleviate this by, for example, projecting 3D points into a 2D view (Li et al. [19]), this leads to a less general solution where the choice

of view fundamentally affects qualitative and quantitative performance. In contrast to this, we develop a method for detecting objects in point clouds "natively in 3D", i.e. without projecting 3D points into lower dimensional representations.

We begin by observing that 3D point clouds are spatially sparse. For example, when discretising real-world point clouds such as from the KITTI Vision Benchmark Suite (Geiger et al. [12]) into a fixed-size grid with a resolution of 20cm, more than 99% of the resulting voxels—or "cells"—are unoccupied. Applying a dense 3D convolution to this sparse grid is thus computationally wasteful as a large number of multiply-add operations have trivial inputs and outputs of zero. Based on this insight, Wang and Posner [20] propose a "feature-centric voting scheme" that implements a "sparse convolution" operation and avoids uninformative multiply-add operations involving zeros. While the computational cost of a dense convolution scales linearly with the product of the spatial dimensions of the input grid, the cost of this voting scheme scales with the number of occupied cells. Moreover, Wang and Posner [20] show that a feature-centric voting is mathematically equivalent to a conventional dense convolution operation. This can therefore lead to considerable computational savings when the input is spatially sparse. Based on this, the method in Wang and Posner [20]—called Vote3D—outperforms prior art on KITTI object detection benchmark; which involves the detection of cars, pedestrians, and cyclists in a variety of road scenes. This is achieved by convolving Support Vector Machines (SVMs) (Cortes and Vapnik [21]) in a sliding window fashion across discretised point clouds at a processing time of only 0.5 seconds per frame.

In order to develop a general, accurate and efficient method for detecting object in 3D point clouds that leverages recent innovations in processing spatial data with CNNs (LeCun et al. [2]), we therefore define the following desiderata:

- convolutional filters should be applied natively in 3D, i.e. without projecting 3D points into lower dimensional representations;

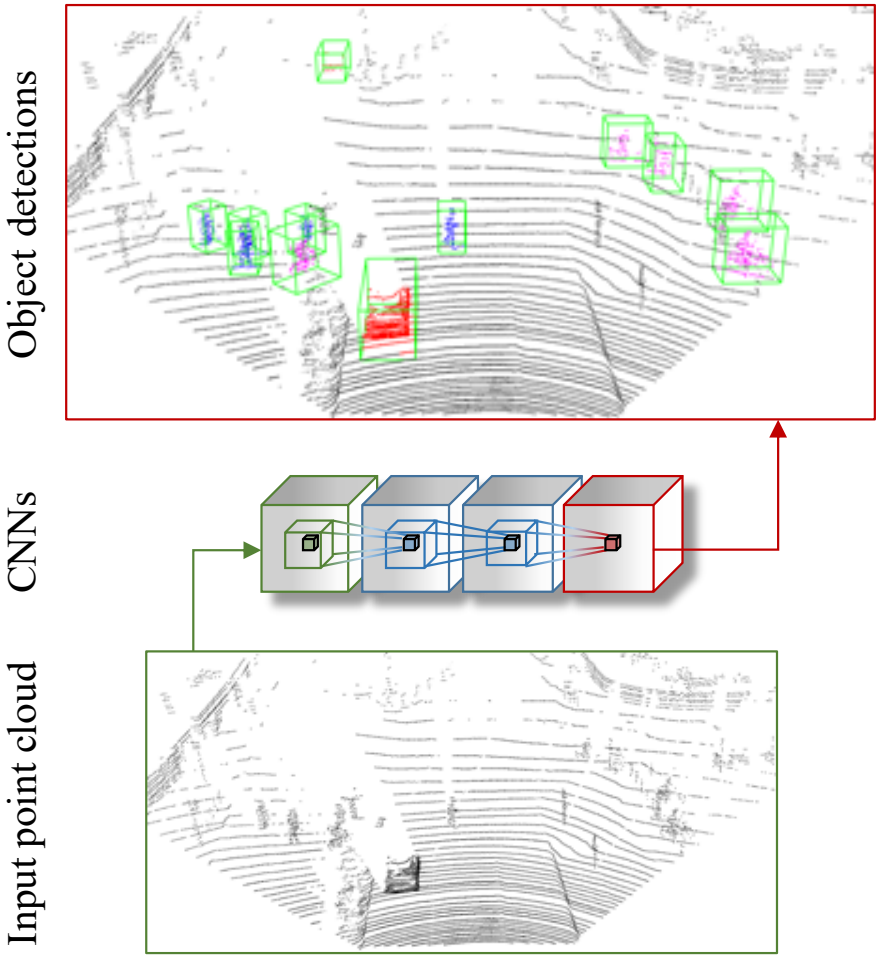
- uninformative and wasteful multiply-add operations should be avoided;

- the entire CNN feature hierarchy should be spatially sparse so that a sparse convolution operation can be leveraged in every layer of the CNN;

- representation sparsity should be explicitly included in the training objective, enabling the practitioner to tune the trade-off between higher sparsity and better accuracy.

Based on these considerations, we present a new method for detecting objects in real-world lidar point clouds, which we call "Vote3Deep" in reference to the use of feature-centric voting from Wang and Posner [20]. An overview of Vote3Deep is illustrated in Figure 2.1. Following prior work (Li et al. [19], Wang and Posner [20], and Behley et al. [50]), we validate the accuracy and efficiency of Vote3Deep on the challenging and competitive KITTI object detection benchmark (Geiger et al. [12]). We show that Vote3Deep outperforms prior art in point cloud based 3D object detection by considerable margins as evaluated in terms of Average Precision (AP), in some cases by absolute differences of more than 30% while maintaining a competitive detection speed of 1.1 seconds per frame on CPU. Moreover, Vote3Deep even outperforms methods that utilise both point cloud and image data for detecting objects.

Some of the materials presented in this chapter are published in Engelcke et al. [42]. Here, we provide a more comprehensive discussion of the subject, including additional experiments with multi-class detection models and an analysis of the sparsity of the intermediate representations. This chapter is structured as follows: Section 2.1 reviews existing literature related to the application of CNNs to three-dimensional data and also discusses other means of using sparsity to accelerate neural networks. Section 2.2 proceeds by providing an overview of Vote3Deep as well as describing its individual components in detail. Experimental results are presented in Section 2.3, including a description of the training details Section 2.3.1, a model comparison Section 2.3.2, a qualitative and quantitative analysis of representation sparsity Section 2.3.3, benchmarking results on KITTI Section 2.3.4, and an analysis of detection speed Section 2.3.5. Finally, Section 2.4 summarises the conclusions of this chapter.



(a) Vote3Deep overview



(b) Reference image

Figure 2.1: (a) Vote3Deep processes point clouds with CNNs natively in 3D to predict the locations of objects of interest. To perform this efficiently, we ensure that the entire CNN feature hierarchy is spatially sparse in order to leverage sparse convolutions in every layer of the CNN. (b) Camera image shown for reference.

2.1 Related Work

Several previous works have applied dense 3D convolutions to the closely related task of point cloud classification (e.g. Prokhorov [51] and Maturana and Scherer [52]). Maturana and Scherer [52] as well as many other works present results on the ModelNet benchmark (Wu et al. [53]), which has become a popular dataset for evaluating point cloud classification performance. The dataset is created by sampling points from the surfaces of CAD models of different types of objects, for example chairs and tables. However, classification of such individual objects is a considerably easier tasks than detecting objects in full real-world point clouds, as the latter requires the processing of much larger inputs. For example, Maturana and Scherer [52] achieve good performance on the ModelNet benchmark by discretising point clouds into a 30-by-30-by-30 occupancy grid, which would be much too coarse for accurate object detection in real-world lidar point clouds such as from the KITTI benchmark [12]. Moreover, while point clouds from ModelNet contain points on the full surface of an objects, lidar sensors only record points that are in the direct line of sight, i.e. there are not returns from occluded surfaces which further complicates the task of 3D object detection. In contrast to these works and in the context of a real-world robotics application, Maturana and Scherer [18] propose to detect landing zones for autonomous helicopters from lidar point clouds. Their method achieves a detection speed of 5ms per m^3. With this processing time, a direct application of their approach to full 3D lidar point clouds with typical dimensions of for example 50m $\times$ 50m $\times$ 5m takes about one minute per frame—far too slow for safety-critical real-time object detection in autonomous driving or other robotics applications.

An alternative approach is proposed in Li et al. [19] who benchmark their method on the KITTI object detection dataset as done in this work. Specifically, Li et al. [19] project 3D points into a frontal view and use a custom encoding for preserving height information of individual points. This avoids the need to perform computationally expensive dense 3D convolutions. Nevertheless, projecting 3D points into a 2D representation implies that multiple points can potentially be

projected into the same location, which therefore loses some of the information contained in the input. Moreover, points that are close in 2D image space might not be close in 3D metric space, for example at object boundaries. The network therefore needs to represent 3D information such as object scale within the two-dimensional feature maps. In this work, point clouds are processed natively in 3D so that each spatial location in a convolutional filter kernel is associated with a 3D volume in physical space of the same size. We posit that this can increase the re-use of individual filter weights, therefore improving generalisation to previously unseen inputs while also facilitating the use of smaller models.

Based on the insight that point clouds are spatially sparse, Wang and Posner [20] propose a feature-centric voting scheme to implement a sparse convolution operation, enabling them to outperform prior art in 3D object detection on the KITTI benchmark. Sparse convolutions via feature-centric voting are also utilised in this work and described in more detail in Section 2.3.3. Graham [54] and Graham [55] also propose to use sparse convolutions for processing spatially sparse data. These works pre-compute the locations of "active cells" in the next feature map in order to avoid performing computations for empty cells. Yet, in contrast to the formulation in Wang and Posner [20], this still requires the consideration of empty cells in the receptive field of the convolutional filters. Jampani et al. [56] propose to perform a sparse convolution on a "permutohedral lattice" that represents the data. The input point clouds considered in Graham [54], Graham [55], and Jampani et al. [56], however, are much smaller than those typically encountered in real-world robotics applications such as considered in this work. Orthogonal to the materials in this chapter, methods have also been developed to accelerate neural networks through sparse weight matrices which contain mostly zeros (Liu et al. [57] and Park et al. [58]). In contrast to these, this work leverages spatially sparse representations to accelerate CNNs. Nevertheless, sparse weight matrices could potentially be utilised as an additional mechanism for further improving detection speeds.

2.2 Vote3Deep

Point clouds are a set of continuous 3D coordinates, but inputs to CNNs need to be spatially discretised into a grid of pixels for 2D images or a grid of voxels for 3D spatial data. In the following, pixels and voxels are generically referred to as "cells" regardless of the number of spatial dimensions. To meet our desiderata, we need to discretise point clouds into a regular, fixed-size grid of cells. Rather than only considering the occupancy of cells as for example in Maturana and Scherer [52], we adopt the discretisation procedure from Wang and Posner [20] to reduce the loss of information during this operation. Specifically, given all 3D points falling into a specific cell, a hand-crafted six-dimensional feature vector is extracted. These six values correspond to (1) a binary occupancy indicator, (2–3) the mean and standard deviation of the lidar reflectance values, and (4–6) three "shape factors" that describe the statistics of the spatial coordinates of the points falling into the cell. This computation can be performed in parallel for all cells that contain at least one 3D point and no computation needs to be performed for empty cells.

After a point cloud is discretised, it can be processed by a series of convolutional layers that are interspersed by non-linear activation functions (see e.g. Krizhevsky et al. [59], Simonyan and Zisserman [60], and Szegedy et al. [61]). This structure allows models to learn complex, non-linear mappings. The convolutional layers can be efficiently implemented using the voting algorithm from Wang and Posner [20], which avoids performing uninformative operations that involve empty cells in the input grid to each layer. For completeness, this is later described in more detail in Section 2.2.1. Crucially, however, special care needs to be taken to ensure that intermediate representations are also spatially sparse—otherwise a sparse convolution could only be applied to the input and all later layers would need to perform regular, computationally expensive dense convolutions. These considerations are elaborated on in Section 2.2.2. Finally, a point cloud can be processed by a CNN in a "fully-convolutional" fashion (see e.g. Long et al. [62]),

which returns a sparse 3D grid of real-valued detection scores. These detection scores can then be further processed to obtain object bounding boxes.

In image-based object detection (e.g. Ren et al. [15], Liu et al. [16], and Redmon et al. [17]), the size of a bounding box for a physical object depends on how far away the object is from the camera. In 3D point clouds, though, the bounding box for an object in three-dimensional metric space is of the same size regardless of an object's location with respect to the sensor. Given that we are concerned with predicting bounding boxes for a fixed set of object categories which have limited variance in physical size, we make a simplifying assumption that all objects of a class are approximately of the same size and use fixed class-specific bounding box dimensions. Unlike recent image-based object detection methods that use CNNs, it is therefore sufficient to predict a grid of detection scores rather than also having to regress to bounding box dimensions. In other words, each cell location in the output grid with a high detection score corresponds to the centre of a predicted 3D object bounding box. In order to handle objects at different angular orientations with respect to the lidar sensor, we again follow Wang and Posner [20] and apply a CNN to a point cloud at N different angular orientations. These angular bins can be processed in parallel. Similar to image-based object detection methods, it is necessary to prune duplicate detections for the same object from spatially neighbouring cells or different angular bins. This can be done by using the well-established technique of Non-Maximum Supression (NMS). NMS greedily discards lower scoring detections whose bounding box overlap with a higher scoring detection is larger than a pre-determined threshold. This is performed separately for each class using a fixed class-specific threshold on the 3D Intersection Over Union (IOU). By using the 3D IOU for thresholding, an issue commonly encountered in image-based object detection can be avoided where two objects of the same class that heavily occlude each other cannot be discerned for a certain range of 2D IOU thresholds.

Detection models are trained as binary or multi-class classifiers on cropped positive and negative examples from the training data. This avoids running the detection models on full point clouds during training where gradients from many

regions are comparatively uninformative, therefore reducing the required training time. Initially, we randomly sample a set of negative examples for training. This set is periodically augmented with additional examples obtained through the well-established technique of Hard Negative Mining (HNM). The training procedure is described in more detail in Section 2.3.1. Moreover, care needs be taken to ensure that the training crops and the receptive field of a CNN are large enough to use most information available to decide whether a certain part of a 3D point cloud corresponds to an object or not. They should therefore be at least as large as the object itself. Further increasing the receptive field and crop size would allow the model to use more contextual information, for example cars are typically on the road or parked in a regular pattern. However, this increases the computational footprint of a forward pass through the model. In the interest of minimising detection time, we therefore choose to match the receptive field of a model and the size of training crops to the 3D dimensions of the object bounding boxes. Given that objects of interest might have different physical sizes, there are two main strategies. Firstly, a set of class-specific networks could be trained where the receptive field of each network closely matches the bounding box dimensions of the object category. Alternatively, a single multi-class network could be used to predict detection scores, whereby the receptive field of the network is set to the maximum bounding box dimension across the different classes along each spatial axis. Section 2.3.2 compares both of these strategies. Alternatively, there is also a third strategy which is a hybrid of the two, where a shared backbone network extracts features that are used for all object classes, and separate class-specific heads with tailored receptive fields predict the detection scores for each class. This hybrid solution is left for future work.

2.2.1 Sparse Convolutions via Voting

For completeness, we summarise the feature-centric algorithm from Wang and Posner [20] to implement a sparse convolution operation in this section. Nevertheless, the reader is referred to Wang and Posner [20] for a more complete treatment of the subject, including a formal proof that the voting scheme is computationally

equivalent to a conventional convolution operation. For simplicity and without loss of generality, we consider a three-dimensional input grid $\mathbf{h}^{c-1} \in \mathbb{R}^{L \times M \times N}$ with a single feature value per cell that is input to layer c in a CNN and where (L, M, N) are the spatial dimensions of the input grid. This grid is processed by a single convolutional filter with weights $\mathbf{w}^c \in \mathbb{R}^{(2I+1)\times(2J+1)\times(2K+1)}$ and filter bias $\mathbf{b}^c$. For odd-valued kernel dimensions as commonly used in the literature (e.g. Krizhevsky et al. [59], Simonyan and Zisserman [60], and Szegedy et al. [61]), the filter weights are indexed as $\mathbf{w}^c_{i,j,k}$ with $\{i, j, k \in \mathbb{Z} \mid i \in [-I, I], j \in [-J, J], k \in [-K, K]\}$. In general, however, feature-centric voting can be applied to inputs of arbitrary spatial dimensionality with multiple input features and using multiple convolutional filters. Considering the locations of a set of occupied cells $\mathbf{\Phi} = \{(l, m, n) \ \forall \ h^{c-1}_{l,m,n} \neq 0\}$ in the input grid, feature-centric voting computes the contribution of each occupied input cell to the activations in the output grid as:

$$z^c_{l+i,m+j,n+k} = z^c_{l+i,m+j,n+k} + w^c_{-i,-j,-k} \, h^{c-1}_{l,m,n} \,, \quad \forall \, i, j, k \,. \tag{2.1}$$

This is performed for all occupied input cells at locations $(l, m, n) \in \mathbf{\Phi}$ in order to obtain the full output grid z^c. Note that the weights $\mathbf{w}^c$ are indexed with $(-i, -j, -k)$ in Equation (2.1). Intuitively, this corresponds to flipping the convolutional filter weights $\mathbf{w}^c$ along each spatial dimension to obtain the associated "voting weights". In other words, rather than aggregating information in the receptive field of the convolutional filter as in a conventional dense convolution operation, the flipped weights cast votes to nearby cells in the output grid. Votes from different occupied input cells at locations $(l, m, n) \in \mathbf{\Phi}$ are aggregated additively in the output grid. Crucially, the computational cost of this algorithm is proportional to the number of occupied input cells rather than the product of the spatial dimensions of the input grid as for a dense convolution. This voting procedure is illustrated in Figure 2.2 using a 2D 5-by-5 example input grid where only two cells are occupied and a 3-by-3 convolutional filter kernel. Note that Equation (2.1) currently omits the bias term b^c, which is a scalar value for a single convolutional filter. The bias term requires special treatment to ensure that sparsity is preserved throughout the CNN hierarchy which

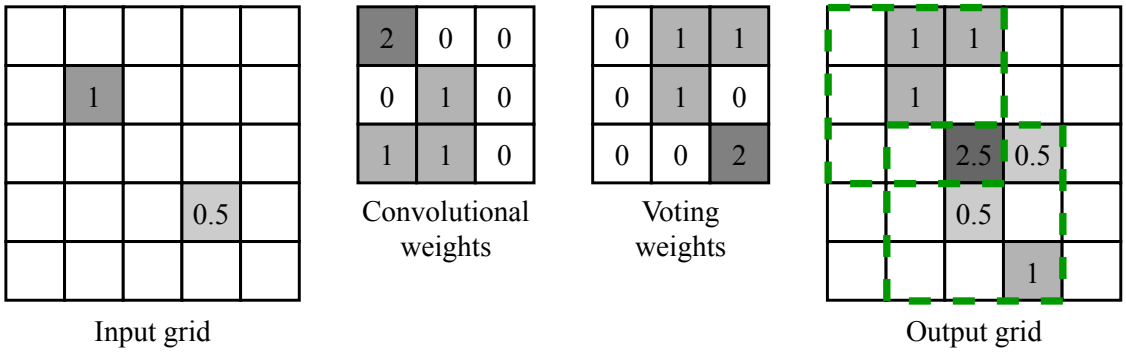


Figure 2.2: Illustration of voting with a sparse 2D input. The voting weights are obtained by flipping the convolutional weights along each dimension. The two green 3-by-3 squares indicate the votes from the occupied cells in the input. Votes are accumulated additively when an output cell receives multiple votes. For example, the cell in the centre of the output grid receives votes of "2" and "0.5", resulting in a value of "2.5".

consists of multiple convolutional layers implemented via feature-centric voting as defined in Equation (2.1). These considerations are discussed in Section 2.2.2.

In contrast to feature-centric voting in Equation (2.1), a standard dense convolution applies the convolutional filter at all $L \times M \times N$ locations in the output grid. For illustrative purposes, we assume input and output grids to be of equal spatial dimensions. The output $z^c_{l,m,n}$ at location (l, m, n) using a 3D dense convolution operation is therefore computed as:

$$z^c_{l,m,n} = \sum_{i=-I}^{I} \sum_{j=-J}^{J} \sum_{k=-K}^{K} w^c_{i,j,k}\, h^{c-1}_{l+i,m+j,n+k} + b^c\,. \tag{2.2}$$

If the input grid is sparse, however, applying the convolutional filter $\mathbf{w}^c$ at all $L \times M \times N$ locations in the output grid is computationally wasteful as most of the operations are trivial multiply-adds by zero. The dense convolution operation in Equation (2.2) therefore requires a significantly larger number of Floating-Point Operations (FLOPs) compared to feature-centric voting in Equation (2.1).

2.2.2 Maintaining Sparsity with Rectifiers

To maximally utilise the benefit of the spatial sparsity of the input data, we would like to leverage the sparse convolutions in all layers of a CNN, not just the input layer. Otherwise, any speed-ups achieved by exploiting spatial sparsity would be minimal. It is therefore necessary to ensure that intermediate feature maps are

also spatially sparse. We begin by noting that convolutional layers in a CNN are typically followed by a non-linear activation function (see e.g. Krizhevsky et al. [59], Simonyan and Zisserman [60], and Szegedy et al. [61]). For activation functions such as Exponential Linear Units (ELUs) (Clevert et al. [63]), the output will be non-zero unless the input element is exactly zero. This would effectively eliminate sparsity in higher layers if $\mathbf{b}^c \neq \mathbf{0}$. Otherwise, output cells that do not receive any votes are mapped to a non-zero value.

Rectified Linear Unit (RELU) non-linearities (Glorot et al. [40])—which correspond to the element-wise $\max(0, \cdot)$ operation—map negative values to zero and are therefore ideal for inducing sparsity. By also constraining the bias term b^c of the convolutional filters to be non-positive by setting $b^c = \min(0, b^c)$ after each parameter update during training, cells that do not directly receive votes remain empty. While this leads to a convolution operation that is slightly less general to a standard convolution, we conjecture that this should only have a small impact on the representational expressiveness of a CNN as a whole. Including the bias term b^c for a single convolutional filter which was excluded previously in Equation (2.1), the combined operation can be written as:

$$h^c_{l,m,n} = \max\left(0, z^c_{l,m,n} + b^c\right) , \tag{2.3}$$

where $z^c_{l,m,n}$ is the output the voting procedure at location (l, m, n) and $h^c_{l,m,n}$ is the output of the element-wise RELU non-linearity at the same location. Consequently, cells in the output grid that do not receive any votes remain empty by default. Moreover, even cells that do receive votes can potentially be considered as empty if all filter activations at a certain spatial location are rectified to be exactly zero. Later in Section 2.3.3, we observe that this behaviour indeed occurs.

2.2.3 Hinge-Loss for Output Sparsity

Due to the use of fixed bounding box dimensions as motivated earlier, the detection networks can be trained as binary or multi-class classifiers on point cloud crops that show positive and negative training examples for a certain class. When

applying these classifiers to full point clouds to predict cell-wise detection scores, we would like the resulting grid of scores to also be sparse to ease the demand for computational resources. For example, having a sparse grid of detection scores avoids the need to apply NMS to all scores in a dense grid where most cells correspond to low-scoring detections. This can be facilitated by using a hinge-loss when training the classifiers, as developed for training SVMs (Cortes and Vapnik [21]). Considering a classifier score $\hat{y} \in \mathbb{R}$ and class label $y \in \{-1, 1\}$ during training, then the hinge loss can be written as:

$$\mathcal{L}_{\text{classification}}(y, \hat{y}) = \max\left(0, 1 - \hat{y} \cdot y\right) . \tag{2.4}$$

This loss encourages classification and detection scores to be either larger than one for positive detections or smaller than negative one for negative detections. When applying the model to full point clouds, this allows us to threshold detection scores at zero or some other selected value to discard low scoring detections.

2.2.4 Sparsity-Inducing Regularisation

While RELU activations and non-positive biases as discussed in Section 2.2.2 facilitate sparsity throughout the CNN feature hierarchy, it is also possible to explicitly encourage sparsity through a suitable regulariser in the training objective. To this end, we propose to use an $\mathcal{L}_1$ loss on the post-RELU activations of the intermediate feature maps. In particular, we normalise this loss term by the spatial dimensions of each feature map so that the resulting gradient magnitude is independent of the absolute dimensions of point cloud crops during training. With C denoting the total number of layers in a CNN, including the output layer, this normalised $\mathcal{L}_1$ loss can therefore be written as:

$$\mathcal{L}_{\text{sparsity}} = \sum_{c=1}^{C-1} \frac{1}{L^c \times M^c \times N^c} \left\| \mathbf{h}^c \right\|_1 . \tag{2.5}$$

This additional loss term can be added to the classification loss and weighted by a scaling factor λ to tune its relative importance. Thus, the full training objective can be written as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{classification}} + \lambda \, \mathcal{L}_{\text{sparsity}} . \tag{2.6}$$

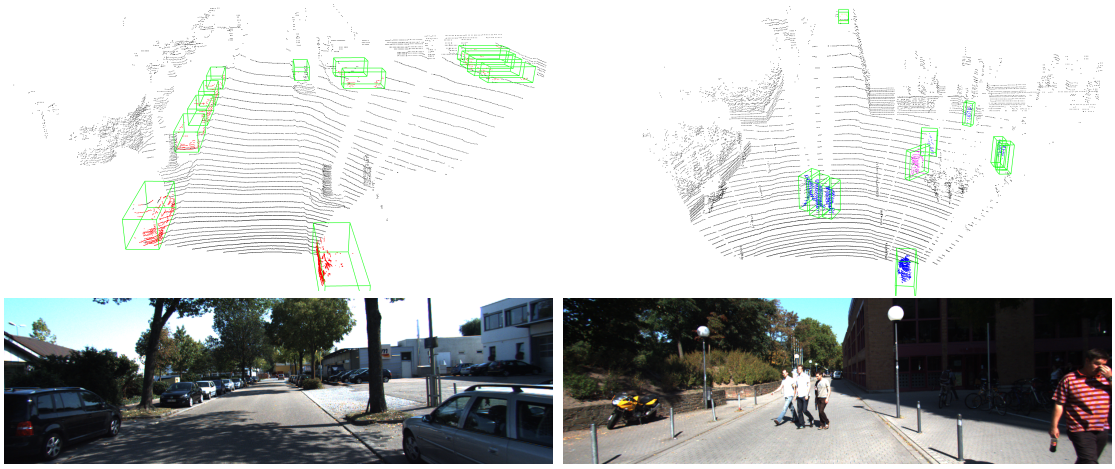


Figure 2.3: Two frames with ground truth labels from the KITTI training set illustrate the diversity of scense encountered in the KITTI Vision Benchmark Suite (Geiger et al. [12]). Images only shown for reference.

2.3 Experiments

To test the utility of Vote3Deep for robotics applications, we need a dataset of 3D point clouds collected with a lidar sensor in the context of a real-world applications such as autonomous driving. Furthermore, a reasonable number of frames needs to have been annotated with ground truth labels for objects of interest so that detection models can be trained and evaluated. The publicly available and popular KITTI Vision Benchmark Suite (Geiger et al. [12]) which contains outdoor scenes captured with a lidar sensor mounted on a moving vehicle in different environments is ideally suited for this purpose. The training set consists of 7,481 annotated frames with 28,742; 4,487; and 1,627 labels for cars, pedestrians, and cyclists, respectively. We set aside 20% of the training frames for validation purposes. The varying numbers of unique positive training examples result from the different frequencies of occurrence of these object categories in the collected data. Two example frames are shown in Figure 2.3, highlighting the broad range of object and background appearances in the dataset.

For evaluation, the object detection benchmark considers the AP of the predicted detections for three-classes: cars, pedestrians, and cyclists. For a prediction to be registered as a positive detection, the IOU of the predicted and the ground truth

bounding box in 2D image space has to exceed a class-dependent threshold. This IOU threshold is 70% for cars and 50% for pedestrians and cyclists. The labels for the test set are not accessible and the benchmark evaluation requires submitting predictions on the 7,518 test frames to a remote test server. The frequency of submission to the test server from a certain user account is limited to prevent hyperparameter tuning on the test set. The benchmark evaluation distinguishes between three difficulty levels: easy, moderate, and hard. The hard test case considers the largest number of positive ground truth objects. For the moderate and easy test cases, labels are ignored based on the object's bounding box size, truncation at the edges of an image, and occlusion through other objects in the scene. The official ranking is determined by the AP on the moderate test case.

The reader is referred to Geiger et al. [12] for further details regarding the dataset and the benchmark evaluation. The complete list of published benchmark submissions can be viewed on the official website.[1] It should be noted that due to an update to the official KITTI benchmark evaluation script in 2017, the results presented here—which are based on the website—are slightly different compared to the ones in Engelcke et al. [42] and other works. For the purpose of participating in the benchmark, we project the predicted 3D bounding boxes into the 2D image plane using the sensor calibration matrix for each frame. We also scale the height and width of the 2D bounding boxes by a class-specific factor which is tuned manually to maximise the AP on the validation set. Any detections that fall outside of the image are discarded. This also allows us to restrict the point clouds to the field of view of the camera when predicting detections, therefore avoiding unnecessary computation.

2.3.1 Training Details

As described in Section 2.2, the models can be trained as binary or multi-class classifiers on point cloud crops which either show an object of interest or background. In addition to positive training examples which are perfectly centred and aligned with the annotated ground truth bounding boxes, we augment the set of positive

[1] Available at `http://www.cvlibs.net/datasets/kitti/eval_object.php`.

examples by randomly translating and rotating examples by a small amount to reduce overfitting on the training data and to counteract the discretisation effects introduced by the finite resolution of the spatial grid and the angular bins. The amount of random translation and rotation is smaller than the size of the grid cells and the angular resolution, respectively. When training several single-class models, we augment the positive ground truth objects by a factor of ten for pedestrians and cyclists and a factor of one for cars. We employed a smaller factor for cars as the number of unique training labels for cars is much larger. For the multi-class models, positive car examples are still augmented by a factor of one, but the augmentation factor for the other two classes is chosen so that the total number of training examples per class is approximately the same for each class. An equal number of initial class-specific negative examples is randomly extracted from the training data at locations that do not overlap with any positive ground truth labels of that class so that the initial ratio of positives and negatives is $1 : 1$.

The set of negative training examples is extended by performing class-specific HNM every ten epochs, effectively constituting a form of "curriculum" (Bengio et al. [64]). For each frame in the training set, we add the ten highest scoring false positives to the training data for single-class models and the three highest scoring false positives for multi-class models per class. The number of mined hard negatives is reduced for the multi-class models to maintain a training time similar to the largest single-class models. This is done in a fashion so that the total number of negatives seen by a multi-class model is comparable to the number of negatives seen by each individual single-class model. A binary hinge loss as described in Section 2.2.3 is used for training single-class models. When training multi-class models, a one-versus-all hinge loss is used for each object class. In this case, a single training crop only contains a positive or negative label for one class; the other classes are ignored and no loss is backpropagated for them. This allows us to mine hard negatives for a class that can potentially coincide with true positives from one of the other classes, for example a cyclist might be a hard negative for pedestrians and vice versa.

Table 2.1: Kernel dimensions for the five architectures considered here. "RF" in the output layer is set according to the desired receptive field for each class.

Architecture	Layer 1	Layer 2	Layer 3
A	RF_A	-	-
B	3×3×3	RF_B	-
C	5×5×5	RF_C	-
D	3×3×3	3×3×3	RF_D
E	5×5×5	3×3×3	RF_E

The models are trained with Stochastic Gradient Descent (SGD), a batch size of 16, a learning rate of 10^{-3}, momentum of 0.9, and $\mathcal{L}_2$ weight decay of 10^{-4}. The filter weights are initialised as proposed in He et al. [65] and models are trained for 100 epochs where an epoch is defined as processing each of the current training crops exactly once. Note that as HNM expands the set of training examples every ten epochs, an individual epoch becomes longer as training progresses. The AP of the current model is computed on the validation set every ten epochs and the model checkpoint that achieves the largest AP on the validation set is selected for evaluation unless otherwise noted. A separate NMS threshold is used for each object class and the thresholds are tuned on the validation set. Training and testing were implemented in C++ with all operations being executed on CPU. Depending on the size of the model, training takes approximately between half a day to four days on a server node with 16 CPU cores, whereby gradients for each training example in a batch are computed in a separate thread.

2.3.2 Model Comparison

Before comparing Vote3Deep against other methods on the official KITTI benchmark, we compare different network architectures for both single and multi-class models in this section to make an informed decision about which variant should be entered to the benchmark. We use the validation set for comparing the models and consider their performance on the moderate test cases to mirror the official KITTI benchmark ranking. Table 2.1 shows the details of the architectures considered here. The networks have up to three layers with 8, 16, or 32 filters per layer. Inspired by

Table 2.2: Class-specific 3D bounding box dimensions and receptive fields of the CNN architectures in Table 2.1 (length × width × height).

Model type	Receptive field
Car	4.8m × 2.0m × 2.0m
Pedestrian	1.4m × 1.4m × 2.2m
Cyclist	2.4m × 2.0m × 2.4m
Multi-class	4.8m × 2.0m × 2.4m

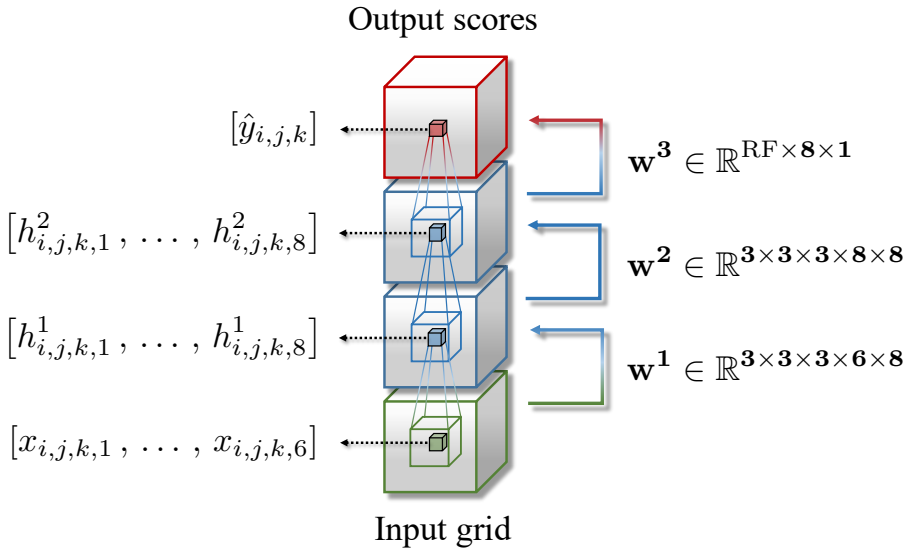


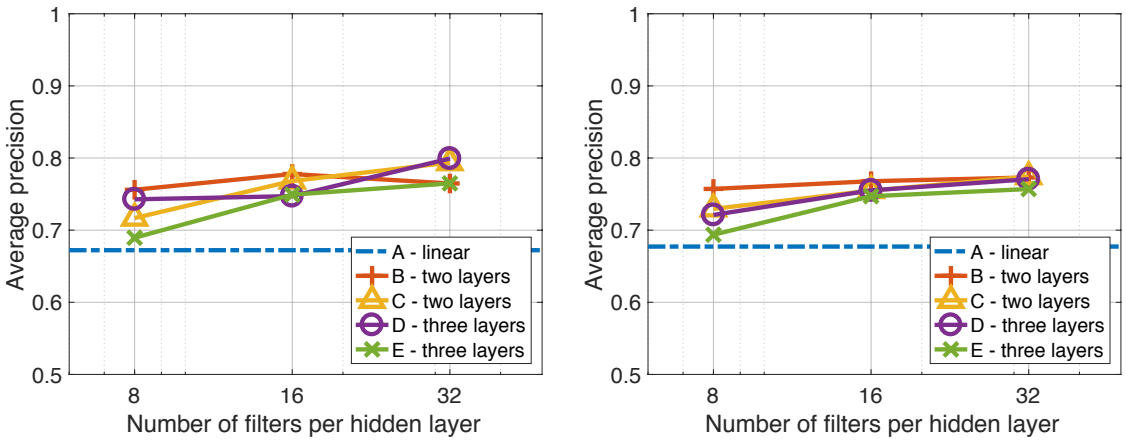
Figure 2.4: Illustration of architecture "D" from Table 2.1. The input $\mathbf{x}$ (green), intermediate $\mathbf{h}^c$ (blue), and output $\hat{\mathbf{y}}$ (red) representations are sparse 3D grids. Solid cubes indicate occupied cells that contain a feature vector in the early layers and a vector of detection scores in the output layer.

the networks commonly used in image-based CNNs (Simonyan and Zisserman [60], Szegedy et al. [61], and He et al. [66]), we limit ourselves to small 3×3×3 and 5×5×5 kernels in the intermediate layers. The receptive fields of the models are summarised in Table 2.2 and are kept the same across network architectures by adjusting the kernel dimensions of the output layer. Similar to Wang and Posner [20], we use a cell size of 20cm in all experiments. A smaller cell size gives a more fine-grained input representation at the cost of having a larger total number of cells, hence deteriorating detection speed. Figure 2.4 provides an illustration of architecture "D".

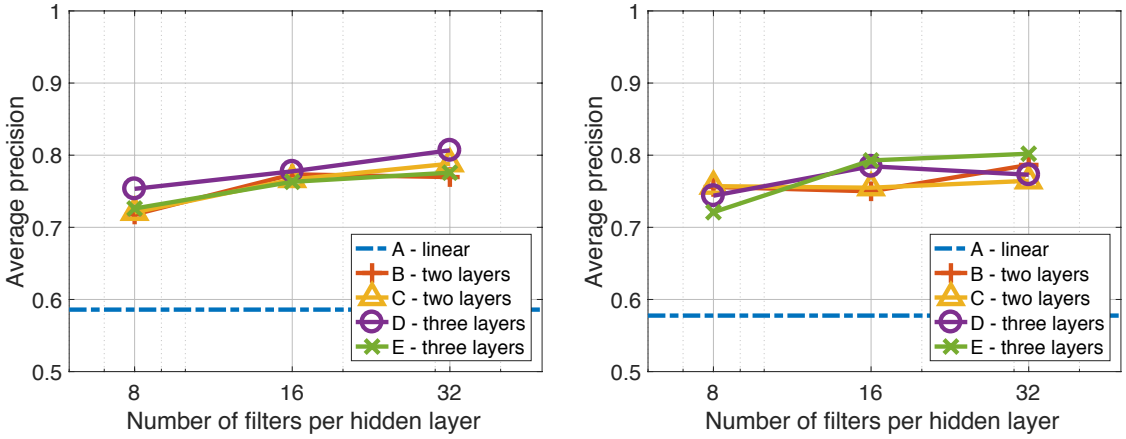
Figure 2.5 shows the AP scores for the single- and multi-class models for each of the three object classes, respectively. These are computed using the validation

set and the results show the performance on the moderately difficult test case. In all scenarios, the non-linear models significantly outperform the linear baselines. Changing the dimensions of the filter kernels or using two vs. three layers does not have a significant impact on performance. While increasing the width of the layers has a positive effect on performance, the returns are diminishing and improvements are moderate in comparison to the considerable gap between the linear baseline and the models with only eight filters in the hidden layers. This might be caused by the larger models starting to overfit. An alternative explanation is that the models are already very close to capturing all the information in the current hand-crafted input representation from Wang and Posner [20]. These observations apply to both the single- and multi-class models. Notably, the latter achieve comparable performance to the single-class models despite seeing 70% fewer hard negatives per class during training. This demonstrates that features in the intermediate representations are re-used effectively. Whereas image-based CNNs typically have a number of parameters in the order of 10^7 to 10^8 (Krizhevsky et al. [59], Simonyan and Zisserman [60], and Szegedy et al. [61]), Vote3Deep is very parameter efficient by achieving high accuracy with fewer layers and a number of parameters in the order of 10^3 to 10^4.

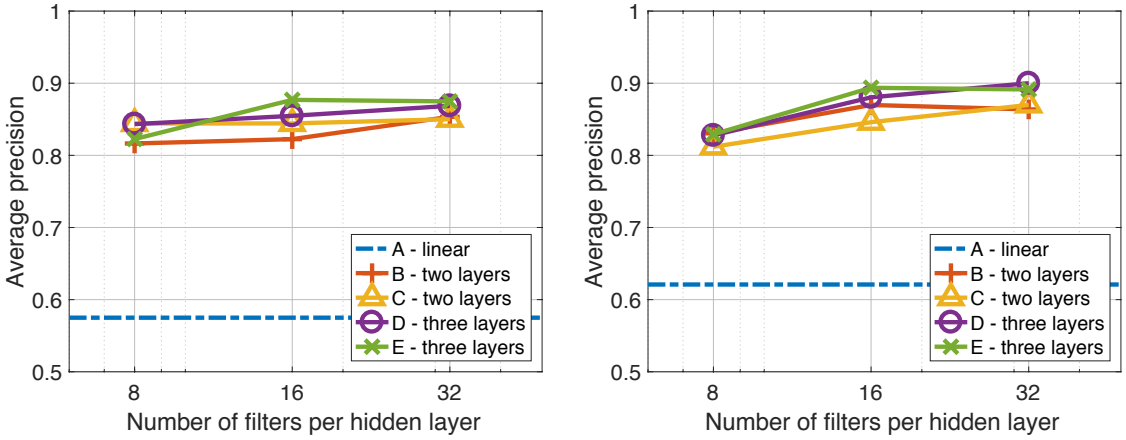
Considering that the specifics of the kernel dimensions do not appear to exhibit a consistent influence on model performance, at least as far as can be told from a single random seed, we use small 3×3×3 convolutional filters in the intermediate layers in all subsequent experiments, which has been shown to be effective in image-based CNNs (Simonyan and Zisserman [60]). Furthermore, the models with eight filters per hidden layer are used for the rest of this chapter unless noted otherwise to avoid the slower detection speeds of the larger models at comparatively small improvements in performance. During deployment, the single-class models can be run in parallel and the overall detection speed is determined by the slowest model. Regarding pedestrian and cyclist detection, we select the three-layer "D" architecture which gives the highest AP for both object classes when using only eight filters per layer. The two layer "B" architecture actually performs best for car detection with eight filters per layer and is therefore selected for further experiments.



(a) Cars: single-class **(b)** Cars: multi-class



(c) Pedestrians: single-class **(d)** Pedestrians: multi-class



(e) Cyclists: single-class **(f)** Cyclists: multi-class

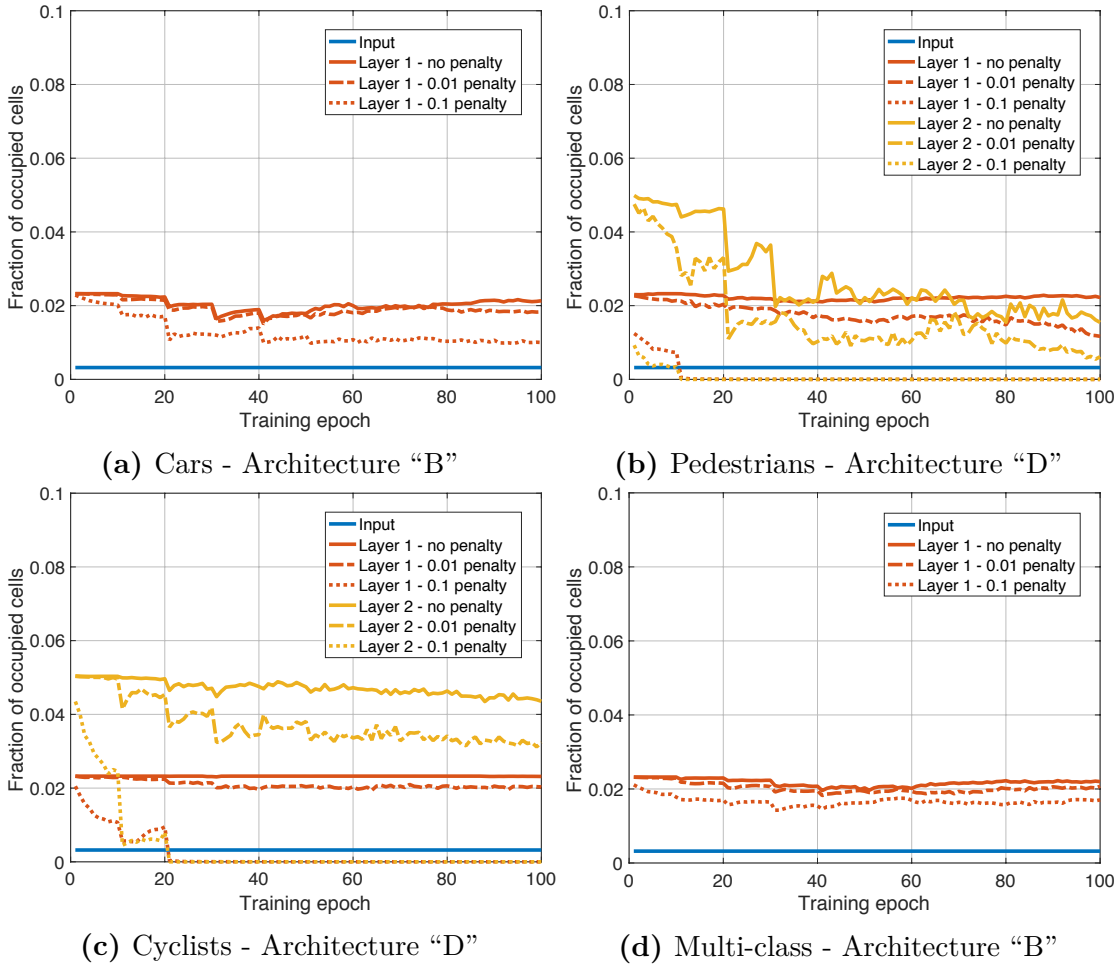
Figure 2.5: Model comparison showing the AP on the validation set (moderate difficulty level) using the architectures in Table 2.1. The multi-layer models significantly outperform the linear baselines.

The speed considerations for different classes do not apply to the multi-class models. Here, we select the "B" architecture for any further experiments as it performs well across all classes and different numbers of filters per layer. Considering the large receptive field of the multi-class models, it is also in the interest of detection speed to choose a model with only two layers.

2.3.3 Representation Sparsity

The main concern when using fast, sparse convolutions is that each convolutional layer can increase the number of occupied cells from layer to layer, even when using RELU non-linearities and non-positive filter biases to encourage sparsity. In particular, all features in a cell have to be zero so that it can be considered as empty. At worst, even a small 3×3×3 filter kernel could increase the number of occupied cells by a factor of $3^3 = 27$ if none of the votes fall into the same cells. To investigate this concern, Figure 2.6 shows the average sparsity as the fraction of occupied cells after each convolutional layer during training for the models selected in Section 2.3.2. This was measured on 100 frames from the validation set at the end of every epoch. Given an example input, Figure 2.7 shows the locations of occupied cells after the first two convolutional layers of the pedestrian models from Figure 2.6b when training with and without the sparsity loss.

As can be seen in Figure 2.6, even without applying the $\mathcal{L}_1$ sparsity loss, the increase in the fraction of occupied cells is considerably smaller than the worst case scenario for all models. Specifically, slightly less than 0.5% of the cells are occupied in the input grid. Unsurprisingly, the first convolutional layer dilates the occupied cells. Considering that 3×3×3 filters are used in all layers, however, up to $3^3 \times 0.5\% = 13.5\%$ of cells could be occupied after the first convolutional layer, which is far more than the $2-3\%$ of cells which are in fact occupied. While there are some newly occupied cells around the edges for the pedestrian model as can be seen when comparing Figure 2.7a and Figure 2.7b, a significant proportion of votes fall into overlapping cells so that the decrease in sparsity is only moderate. Intriguingly, for the pedestrian models, the second intermediate representation is actually sparser



(a) Cars - Architecture "B"

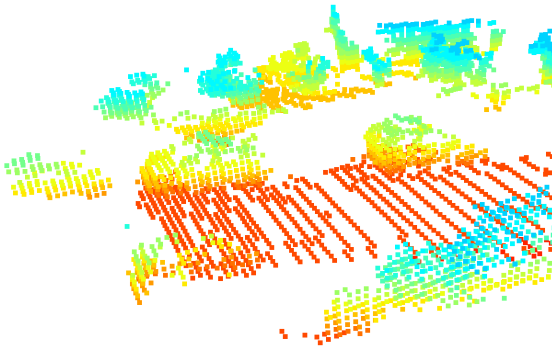
(b) Pedestrians - Architecture "D"

(c) Cyclists - Architecture "D"

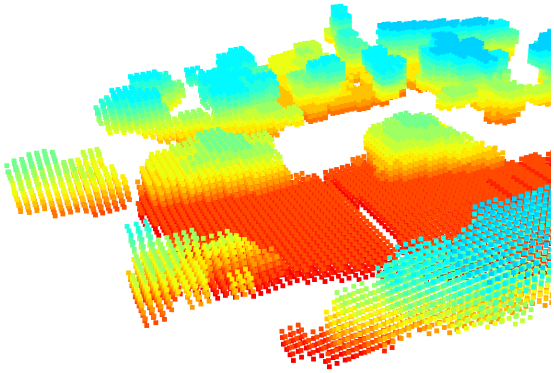
(d) Multi-class - Architecture "B"

Figure 2.6: Fraction of occupied cells averaged for different weightings of the $\mathcal{L}_1$ sparsity loss. The models maintain a high sparsity ratio even without the sparsity loss, for example the second layer of the pedestrian model is sparser than the first layer, but the loss reduces the fraction of occupied cells even further. In some cases where the weighting is too large, the models collapse and all activations become zero.

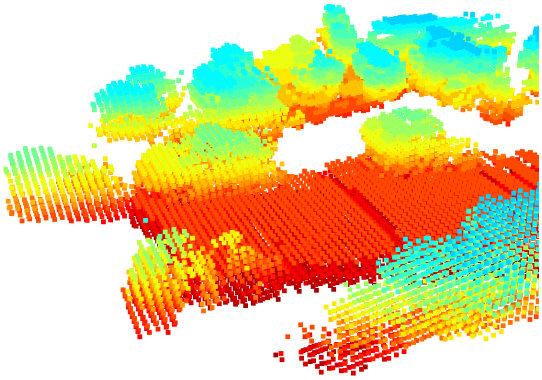
than the first one as seen in Figure 2.6b, indicating that the models proficiently discard features in physically unoccupied locations even without an explicit sparsity loss being included in the training objective. This is also confirmed when comparing Figure 2.7b and Figure 2.7c. A further observation is that during the early training epochs, the fraction of occupied cells drops rapidly every ten epochs, i.e. after every round of HNM. We hypothesise that after HNM a large number of previously high-scoring positives is suddenly presented to the model as negative training examples, so the model temporarily becomes overly cautious at making positive predictions and hence pushes activations to zero before it becomes more confident again.



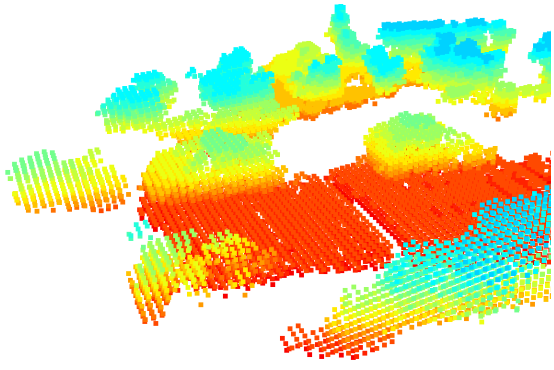
(a) Input



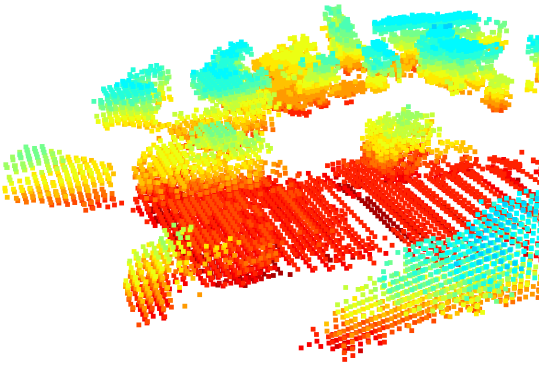
(b) First layer, no penalty



(c) Second layer, no penalty



(d) First layer, 0.01 penalty



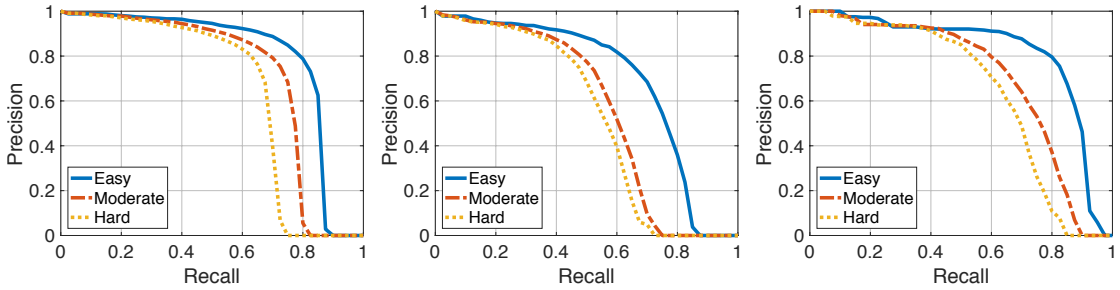
(e) Second layer, 0.01 penalty

Figure 2.7: Occupancy of a discretised point cloud segment from the validation set and the associated occupied spatial locations in the intermediate representations of the pedestrian model with architecture "D", eight filters, and after 100 epochs of training both without and with the $\mathcal{L}_1$ sparsity loss. It can be seen that the sparsity loss effectively reduces the number of occupied cells.

When using the $\mathcal{L}_1$ sparsity loss during training, the fraction of occupied cells is consistently lower throughout the feature hierarchy, as can be seen in Figure 2.6. This effect becomes stronger as the weighting of the sparsity loss is increased. If the penalty is too large, however, the activations collapse to zero as can be seen for the pedestrian and cyclist models in Figure 2.6b and Figure 2.6c respectively. More specifically, this collapse occurs right after HNM at epochs 10 and 20, where the model is confronted with previously unseen and difficult examples that push the activations to zero. We conjecture that the car and multi-class models do not suffer from this collapse for the same weighting of the sparsity loss because their receptive fields are much larger. Consequently, the loss is spread over a larger number of cells and discarding some of them will still allow gradients to be propagated through the network. Notably, the fraction of occupied cells after the first convolutional layer is lower for the car models than for the multi-class models, which is not a surprising observation as the multi-class models are expected to require more information for detecting three object classes instead of just one. A direct, qualitative comparison of the representation sparsity when training without and with the $\mathcal{L}_1$ sparsity loss can be made between Figure 2.7b and Figure 2.7d as well as Figure 2.7c and Figure 2.7e. With the sparsity loss, the model discards, for example, features on the ground which are highly redundant as well as features far out on the right-hand whose appearance does not resemble any objects of interest. Overall, the sparsity in the intermediate representations can possibly be further increased by performing a more fine-grained search for the relative weighting of the sparsity loss during training.

2.3.4 KITTI Benchmark Results

To compare Vote3Deep to pior art, the single-class models selected in the model comparison were submitted to the official KITTI test server. The corresponding Precision-Recall (PR) curves are shown in Figure 2.8. In addition, we also submitted a multi-class model with 32 filters in the hidden layers, which we refer to as Vote3DeepXL. Table 2.3 compares both models against all other methods that only utilise laser data and are listed on the official ranking table at the time when



(a) Cars **(b)** Pedestrians **(c)** Cyclists

Figure 2.8: PR curves obtained with Vote3Deep on the official KITTI test set.

Table 2.3: AP in % on the KITTI test set for detecting objects in point clouds.

		Cars			Pedestrians			Cyclists		
	Speed	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Vote3Deep	1.1s	**76.95**	**68.39**	**63.22**	**67.94**	**55.38**	**52.62**	**76.49**	**67.96**	**62.88**
Li et al. [19]	1.0s	70.68	53.45	46.90	-	-	-	-	-	-
Wang and Posner [20]	**0.5s**	56.66	48.05	42.64	44.47	35.74	33.72	41.45	31.24	28.60
Plotkin [67]	3.5s	35.24	26.13	22.69	-	-	-	-	-	-
Behley et al. [50]	10s	37.63	23.76	18.44	44.36	31.37	30.62	28.19	21.62	20.93
Vote3DeepXL	5.7s	**77.56**	**71.64**	**66.98**	**69.30**	**60.60**	**58.40**	**79.67**	**71.06**	**65.39**

Table 2.4: AP in % on the KITTI test set for approaches that use both point cloud and image data as indicated with *.

		Cars			Pedestrians			Cyclists		
	Speed	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Vote3Deep	**1.1s**	**76.95**	68.39	**63.22**	67.94	55.38	**52.62**	**76.49**	**67.96**	**62.88**
Gonzalez et al. [68]*	4s	76.49	**69.92**	57.47	**73.05**	**56.59**	49.63	51.46	42.61	37.42
Premebida et al. [69]*	30s	-	-	-	59.38	46.67	42.05	-	-	-
Vote3DeepXL	5.7s	**77.56**	**71.64**	**66.98**	**69.30**	**60.60**	**58.40**	**79.67**	**71.06**	**65.39**

Engelcke et al. [42] was submitted for publication. Vote3Deep outperforms the other methods by significant margins across all classes and difficulty levels at a processing time of about 1 second per frame on a 2.5 GHz Intel Core i7 CPU. Vote3DeepXL further improves upon this at the expense of detection speed. The improvements are particularly large for cyclist detection where the AP is increased by more than a factor of two on the moderate and hard test cases, even though the cyclist detectors have the smallest number of only 1,627 annotated positive examples available for training. We conjecture that cyclists are more distinctive than pedestrians and cars as they are less easily confused with common shapes such as straight poles and walls. Furthermore, the parameter efficiency of Vote3Deep—as discussed in Section 2.3.2—

allows models to generalise well even when learning from a small number of positive training examples. Finally, Table 2.4 shows a comparison of Vote3Deep models against methods which utilise both laser and image data for detection. Even without using image data, Vote3Deep achieves a higher accuracy in most test cases at a better detection speed compared to prior art. We conjecture that using an optimised GPU implementation of sparse convolutions as well as an improved input representation could lead to models that achieve even better performance while remaining light-weight in terms of computation and the number of parameters.

2.3.5 Detection Speed

As fast detection speeds are paramount in robotics, in particular in safety-critical applications such as autonomous driving, we compare the three single-class models from the KITTI benchmark submission as well as the the "B" multiclass model—all with eight filters in each hidden layer—with regards to detection speed and accuracy when training them with different weightings for the $\mathcal{L}_1$ sparsity loss. The mean and variance of the detection speeds as measured on the first 200 frames of the validation set are summarised together with the AP on the moderate cases from the validation set in Table 2.5 for single-class models and Table 2.6 for multi-class models.

Overall, pedestrians have the fastest detection times which can be attributed to the fact that they have the smallest receptive field. The two-layer car model is faster than the three-layer cyclists model despite the former having a larger receptive field. Interestingly, the two-layer multi-class model is slightly slower than a sequential combination of the single-class models even though the multi-class model shares computation in the first layer. In particular, the receptive field of the single-class car model is only marginally smaller compared to the multi-class model but runs more than three times faster. We attribute this to the very large receptive field of 22×8×10 of the multi-class model which dominates the computation time. More specifically, the last layer of the multi-class filter has three instead of one output filter, hence increasing the number of parameters and the computational footprint of the layer by a factor of three. This shortcoming could be addressed

Table 2.5: $\mathcal{L}_1$ sparsity loss versus detection speed in milliseconds and AP in % on the validation set for the single-class models.

Class	Penalty	Run-time	Easy	Mod.	Hard
Cars	0	784±205	88.85	75.60	70.01
	10^{-3}	822±222	89.68	**76.45**	**70.19**
	10^{-2}	749±191	89.25	76.13	69.58
	10^{-1}	**491±109**	**89.75**	74.30	67.83
Pedestrians	0	538±133	82.90	75.33	68.70
	10^{-3}	558±132	**83.04**	**76.81**	**70.06**
	10^{-2}	**399±88**	78.68	72.56	66.48
	10^{-1}	-	-	-	-
Cyclists	0	1033±304	**91.78**	**84.33**	81.96
	10^{-3}	1005±289	90.91	83.62	**82.47**
	10^{-2}	**822±224**	90.20	83.55	81.92
	10^{-1}	-	-	-	-

Table 2.6: $\mathcal{L}_1$ sparsity loss versus detection speed in milliseconds and AP in % on the validation set for the multi-class models.

		Cars			Pedestrians			Cyclists		
Penalty	Run-time	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
0	2684±754	88.00	75.72	70.60	82.27	75.59	69.98	88.32	83.05	81.21
10^{-3}	2759±810	87.98	74.55	68.88	81.20	74.87	69.51	88.90	83.18	81.50
10^{-2}	2622±753	**89.32**	75.58	70.02	82.89	76.38	**71.52**	87.45	82.00	80.35
10^{-1}	**2068±600**	89.30	**76.15**	**71.28**	**83.24**	**77.33**	71.27	**91.75**	**85.34**	**83.63**

by designing a model with several class-specific output layers, each with dedicated kernel dimensions so as to minimise computation cost; a task left for future work.

For both single- and multi-class models, including the $\mathcal{L}_1$ sparsity loss in the training objective leads to considerable improvements in detection speed, approximately 40% for cars and 25% for the other three categories for the largest admissible loss term. This matches the observations from Section 2.3.3 that the $\mathcal{L}_1$ sparsity loss indeed reduces the fraction of occupied cells. The smallest loss term of 10^{-1} does not have a large impact on detection speeds. For the single-class models, the AP deteriorates slightly for larger loss terms. Yet, for the multi-class model, the highest Mean Average Precision (MAP) is obtained with the largest loss term, demonstrating that the $\mathcal{L}_1$ sparsity loss can also have a beneficial effect on generalisation in some cases, as far as can be told from these results which are obtained with a single random seed.

2.4 Conclusions

This chapter introduces Vote3Deep, a novel method for detecting objects in 3D lidar point clouds. This is achieved without projecting 3D points into a lower dimensional view as in previous works. Instead, the computational cost of applying dense convolutions to 3D data is avoided by employing the feature-centric voting scheme from Wang and Posner [20] to implement a sparse convolution operation which is used as the main building block of CNN-based detection models. Sparsity throughout the feature hierarchy is maintained using a combination of RELU non-linearities and non-positive filter biases. In addition, a sparsity inducing $\mathcal{L}_1$ loss is proposed to further increase sparsity. Vote3Deep is validated on the KITTI object detection benchmark, outperforming prior art for object detection in point clouds by large margins. Furthermore, Vote3Deep even outperforms methods that utilise both point cloud and image data. A set of timing experiments reveals that the proposed $\mathcal{L}_1$ penalty is indeed beneficial for inducing higher sparsity and thus further increasing detection speeds at minimal impact on detection accuracy. This is also corroborated by a qualitative and quantitative investigation of the sparsity in intermediate representations. With regards to the objectives of this thesis, these results demonstrate that in comparison to prior art, Vote3Deep is able to reliably detect objects of interest in real-world lidar point clouds in a computation-efficient manner and even performs exceptionally well at detecting cyclists where only 1,627 manually labelled positive training examples are available.

2.4.1 Limitations

The main technical limitations of Vote3Deep as presented in this chapter are the fixed bounding box dimensions, the finite number of angular bins, and the use of hand-crafted feature vectors to capture the statistics of lidar point falling into individual grid cells. These three components all limit the maximum accuracy that can be achieved by the detection models. When visualising false negative detections, it can be observed that the model often predicts a bounding box that roughly corresponds to a missed ground truth annotation, but IOU between the ground

truth and predicted bounding box is below the required threshold for the detection to be registered as a true positive. Vote3Deep also appears to struggle in particularly with detecting objects that are further away from the lidar sensor. Furthermore, the fact that detection performance only improves marginally as the number of layers and filters is increased (Figure 2.5) possibly indicates that too much information is lost when only using a six-dimensional vector for representing each input cell. Finally, it is arguably not desirable to apply the $\mathcal{L}_1$ sparsity penalty equally to all cells in the intermediate feature representations as some cells are expected to remain occupied.

2.4.2 Future Work

In order to avoid fixed size bounding boxes and using discrete angular bins, a bounding box regression head could be used commonly done in the literature on image-based object detection (e.g. Ren et al. [15], Liu et al. [16], and Redmon et al. [17]). Moreover, rather than extracting fixed, hand-crafted feature vectors for the points in each grid cell, these representations could be jointly learned; for example using symmetric aggregation functions (Zaheer et al. [70] and Qi et al. [71]), Kd-networks (Klokov and Lempitsky [72]), or OctNets (Riegler et al. [73]). This would avoid any information being lost during discretisation as long as the embedding dimensions is large enough (Wagstaff et al. [47]). These changes, however, would also make the model more expensive to run, making it less impractical for real-world applications. This could be overcome by using GPU-accelerated sparse convolutions, as for example already explored in Yan et al. [74]. Several software libraries that offer this capability have appeared since this work was published in Engelcke et al. [42], for example as presented in Graham et al. [46], Ren et al. [75], Choy et al. [76], Hackel et al. [77], and Tang et al. [78]. In particular, Graham et al. [46] and Ren et al. [75] also propose a structured approach to maintaining sparsity throughout the CNN. Graham et al. [46] constrain the sparsity pattern in an output grid to be identical to the pattern in the associated input grid, so that information flows along lower-dimensional "submanifolds" while Ren et al. [75] use binary masks to differentiate between empty and occupied regions in point clouds. Moreover, Han

et al. [79] utilise sparse convolutions for the related task of instance segmentation in 3D point clouds and it might be possible to build upon insights from this work. An alternative approach for 3D detection without point cloud discretisation is presented in Qi et al. [80]. Recently, Tang et al. [78] proposed to combine sparse convolutions as used in this chapter and point-based operations as proposed in Zaheer et al. [70] and Qi et al. [71] to combine the relative merits of the two approaches.

3

Unsupervised Object Discovery and Object-Centric World Modelling with Autoregressive Priors

As we saw in Chapter 2 in the context of autonomous driving, robotic tasks require interactions with object-like entities. In particular, we investigated the detection of other road users in the form of cars, pedestrians, and cyclists. In this well-defined task, it is possible to adopt a supervised learning approach and to use a large dataset with detailed object annotations (Geiger et al. [12]). This approach, however, is ill-suited for applications that routinely involve interactions with novel types of objects that have not been encountered before. For example, when learning robotic skills from human demonstration (e.g (Yu et al. [81])) in small-batch manufacturing applications, a human demonstrator might show the robot how to do a novel task that involves tools or components that the robot has not encountered before, for example a custom 3D printed manufacturing component. Yet, we would like the perception module to recognise this as a physical object and treat it appropriately: for example, the object will most likely not suddenly disappear or fall apart without cause, we should avoid breaking it, or we can put another object on top of it.

While humans are able to seamlessly partition their environment into suitable objects or components, "objectness" as such is not uniquely defined; what we might

identify as an object is both context and task dependent. For example, while a car might be a viewed as a single object when driving on a road, we would want to view it as a collection of parts when performing repairs on it. Given the fluency of how object attribution changes, we would similarly like to have a means for a robot to identify object-like entities without supervision in the context of its domain of operation. Given the unsupervised nature of this problem, this lends itself to a generative modelling approach.[1] The fundamental idea is to generate images—as we consider in this chapter—while structuring the generation process in a way so that object-like building blocks can emerge (e.g. (Huang and Murphy [22], Eslami et al. [23], and Greff et al. [82])). We therefore refer to these types of models as Object-Centric Generative Models (OCGMs). Moreover, generative models are not only suitable for discovering structure in data, but their ability to capture a concise representation of the training data also allows them to be used as a "world model" for skill acquisition in a learned simulation, offering the potential of considerably increasing the data efficiency of skill acquisition with Reinforcement Learning (RL) algorithms (Ha and Schmidhuber [29]). Based on the central role of objects in human and robotic skills, we posit that such world models should consequently also be structured in an object-centric fashion to ensure that dynamics, interactions, and physical constraints are properly represented. With regards to the objectives of this thesis, OCGMs thus present a promising avenue for increasing supervision efficiency in robotics tasks.

A mathematically principled way to formulate a generative model and to discover structure in data is to frame this problem as learning a probabilistic latent-variable model. This is based on the assumption that individual examples in a dataset are actually samples from a probability density function $p(\mathbf{x})$ and the goal is to recover this generative distribution with a model $p_\theta(\mathbf{x})$ parameterised by θ. A probabilistic latent variable model further assumes that $p_\theta(\mathbf{x})$ can be factorised in

[1]The notion that object understanding should be unsupervised was discussed by Klaus Greff in his talk "What are Objects" at the ICML 2020 Workshop on Object-Oriented Learning (OOL). Workshop page: `https://oolworkshop.github.io/index.html`. Recording of the talk: `https://slideslive.com/38930701/what-are-objects`.

terms of unknown probabilistic latent variables $\mathbf{z}$ which captures hidden attributes of each example in the dataset. The generative distribution $p_\theta(\mathbf{x})$ is retrieved by marginalising out the uncertainty with respect to $\mathbf{z}$ so that:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) \, \mathrm{d}\mathbf{z} \, . \tag{3.1}$$

Furthermore, OCGMs make the particular assumption that these latent variables can be sub-divided into a set of latents $\mathbf{z}_{1:K}$, whereby each $\mathbf{z}_k$ represent a single object-like component and K is the total number of components:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}_{1:K}) p_\theta(\mathbf{z}_{1:K}) \, \mathrm{d}\mathbf{z}_{1:K} \, . \tag{3.2}$$

While the distribution $p_\theta(\mathbf{x} \mid \mathbf{z}_{1:K})$ describes the generative process of how an image $\mathbf{x}$ is obtained given its description in latent space in the form of $\mathbf{z}_{1:K}$, it is possible to formulate an associated inference model $p(\mathbf{z}_{1:K} \mid \mathbf{x})$ that determines the underlying latent variables from given image, hence allowing us to "decompose" the image into its constituent scene components. The true latent posterior $p(\mathbf{z}_{1:K} \mid \mathbf{x})$, though, is generally intractable. One solution to this is to formulate variational approximation $q_\phi(\mathbf{z}_{1:K} \mid \mathbf{x})$ with parameters ϕ which are optimised to minimise the difference between the true and the approximate latent posterior according to a measure such as the Kullback-Leibler (KL) divergence (Jordan et al. [83]).

A range of works have explored this paradigm in recent years and manage to identify object-like entities in simple visual scenes using attributes such as colour, shape, and movement (Huang and Murphy [22], Eslami et al. [23], Kosiorek et al. [24], Burgess et al. [25], Greff et al. [26, 82, 84], and van Steenkiste et al. [85]). Early incarnations of this approach are fairly limited in terms of the visual complexity of input images they can handle, for example with experiments being conducted on datasets showing multiple MNIST digits (LeCun et al. [86]) on a uniformly-coloured background (Huang and Murphy [22], Eslami et al. [23], and Greff et al. [82]). Important breakthroughs are presented in Burgess et al. [25] and Greff et al. [26] who propose two models called Multi-Object Network (MONET) and Iterative Object Decomposition Inference Network (IODINE), respectively. These are the

first works of this paradigm that are able to learn object-like scene decompositions from rendered images of simulated 3D environments. Both of these approaches, however, have the same two limitations: Firstly, both MONET and IODINE have a fully-factorised prior that assumes that scene components are independent. The underlying generative model can therefore not capture relationships between components, such as physical constraints that prohibit two objects from occupying the same space. It is therefore not possible to generate coherent scenes in an object-centric fashion, therefore making these models unsuitable as world models for facilitating robot skill acquisition. We dedicate this chapter to develop a solution to this problem. Secondly, both works only demonstrate the ability to learn object-like representations from simulated data. Yet, practical robotics applications require learning such representations from more complex real-world data, which we are going to explore in Chapter 4.

Specifically, in this chapter we introduce a novel model called Generative Scene Inference and Sampling (GENESIS), which is both able to infer interpretable object-centric representations from rendered images of simulated 3D environments and to generate novel-scenes in an object-centric fashion. This is achieved by using an autoregressive prior to capture relationships between scene components in the generative model. The model is designed so that, unlike MONET and IODINE, any recurrent operations are executed in a low-dimensional latent space with all convolutional operations being parallelised across scene components to facilitate better GPU utilisation. Furthermore, we conduct an investigation of "reconstruction bottlenecks" and show that these critically affect model behaviour. Some of the materials in this chapter are published in Engelcke et al. [43] and Engelcke et al. [44]. In particular, Engelcke et al. [43] presents the empirical results in Sections 3.3.5 to 3.3.7 and Engelcke et al. [44] presents the results in Section 3.3.8. This chapter provides a unified and more extensive discussion of this material. For reproducibility and benefit to the community, a repository including code for training all models as well as checkpoints of pre-trained models is publicly available at `https://github.com/applied-ai-lab/genesis` under an open-source license.

As of early October 2020, the repository has received 25 "GitHub stars". After reviewing relevant prior work in Section 3.1 and describing the details of the model in Section 3.2, GENESIS is evaluated both qualitatively and quantitatively in Section 3.3 on four well-established datasets: Multi-dSprites (Burgess et al. [25]), GQN rooms-ring-camera (Eslami et al. [87]), ShapeStacks (Groth et al. [88]), and ObjectsRoom (Burgess et al. [25]). While Multi-dSprites is a visually simple toy dataset, the latter three consist of rendered images of simulated 3D environments containing different types of foreground objects and background appearances as might be encountered in robotic manipulation applications. Finally, Section 3.4 summarises the conclusions of this chapter.

3.1 Related Work

The field of OCGMs is still nascent with the earliest pioneering works only having been published about 5 years ago (Huang and Murphy [22], Eslami et al. [23], and Greff et al. [82]). The small number of available works can be categorised according to their learning objective: there are reconstruction-based latent variable models which are typically formulated as Variational Autoencoders (VAEs) (Kingma and Welling [27] and Rezende et al. [28]) and adversarial models based on Generative Adversarial Networks (GANs) (Goodfellow et al. [31]). GENESIS belongs to the former domain and is also formulated as a VAE. VAEs have the advantage that both an inference and a generative model are jointly learned in a mathematically principled fashion without suffering from "mode collapse" as often encountered when training GAN-based models. Reconstruction-based models can be further subdivided into the mechanism used for distinguishing object instances. There are approaches which use Spatial Transformer Networks (STNs) (Jaderberg et al. [89]) for extracting bounding boxes that correspond to objects (Huang and Murphy [22], Eslami et al. [23], Kosiorek et al. [24], Xu et al. [90], Stelzner et al. [91], and Crawford and Pineau [92]). Other works adopt "spatial mixture models" where objects are distinguished by instance segmentation masks (Burgess et al. [25], Greff et al. [26, 82, 84], and van Steenkiste et al. [85]).

Huang and Murphy [22] and Eslami et al. [23] are the first two works to frame unsupervised learning of object-centric representations as structured VAEs with STNs. Following these seminal works, Kosiorek et al. [24] extend Eslami et al. [23] to handle temporal sequences. Stelzner et al. [91] accelerate learning in a model similar to Eslami et al. [23] by making use of "Sum-Product Networks" and Crawford and Pineau [92] take inspiration from single-shot object detection models (Redmon et al. [17]) to infer a grid of object latents instead of inferring latents sequentially as in Eslami et al. [23]. Finally, Xu et al. [90] propose an extension to Eslami et al. [23] to generate better images, but their approach is still limited to simple images with white objects on a black background. In parallel to Huang and Murphy [22] and Eslami et al. [23], Greff et al. [82] propose to learn object-centric representations without supervision by framing the problem as inference in a spatial mixture model. This approach was subsequently refined in Greff et al. [26, 84] and van Steenkiste et al. [85]. While these works use amortised iterative inference, Burgess et al. [25] instead propose to use an attention network to directly predict segmentation masks that are used to infer the approximate posterior. Unlike the models above which are formulated as VAEs, Kosiorek et al. [93] frame unsupervised object-centric learning as a "Capsule Network" (Sabour et al. [94]). GENESIS as presented in this chapter utilises a spatial mixture formulation rather than STNs. The former is arguably more general than the later as objects can take arbitrary shapes and sizes, whereas the fixed-size sampling grid of the latter imposes a strong bias on the size of discovered objects. It might be possible to overcome this with a multi-scale formulation, but this would involve other challenges such as choosing the number and distribution of scales. The approaches most similar to GENESIS are MONET (Burgess et al. [25]) and IODINE (Greff et al. [26]). However, the main difference to these works is that GENESIS features an autoregressive prior which can capture relationships between scenes components, thus facilitating unconditional object-centric generation of entire coherent scenes in addition to being able to infer object-like scene components from a given image. The architecture of GENESIS

is more similar to MONET than IODINE and avoids the iterative, gradient-based inference mechanism of IODINE which is comparatively expensive in terms of computation. Unlike MONET, however, GENESIS performs any recurrent operations in a low-dimensional latent space so that component-wise convolutional modules can be efficiently parallelised on GPU. Importantly, these differences render GENESIS more suitable for robotics applications, both in terms of computationally efficient inference and as a world model for supervision-efficient skill acquisition.

VAEs as utilised here fall into the group of explicit density models as they explicitly approximate the underlying generative distribution by a model $p_\theta(\mathbf{x})$ with parameters θ. In contrast, GANs (Goodfellow et al. [31]) are an implicit density model. While GANs lack an inference model to evaluate the likelihood of an input image, it is still possible to generate synthetic images. Several works have explored GANs for either learning to segment objects in images without supervision as well as for generating novel images in an object-centric fashion. Given that an object's appearance is somewhat independent of its context, Chen et al. [95] learn to segment objects by first predicting a mask and then training a generator that tries to fill in the mask to create a realistic image. Bielski and Favaro [96] build on the idea that if an object in an image is segmented perfectly, then translating the object pixel-wise should result in a perturbed image which still looks realistic. This is taken one step further in Arandjelović and Zisserman [97] who propose to learn to cut out an entire object from one image and paste it into another image to create a third realistic looking image. This copy-and-paste procedure should only result in a realistic looking image if the copied object is well segmented. A similar approach is taken in Azadi et al. [98] who train a model to combine two images that either each contain a single object, or one shows an object and the other a background scene. Lastly and most similar to this work, van Steenkiste et al. [30] consider a more general setting of generating images with a potentially large number of objects, but their approach is limited to simulated datasets such as CLEVR (Johnson et al. [99]). In contrast to GENESIS, most of these adversarial models only segment a single foreground object in an image, with van Steenkiste et al. [30]

being a notable exception. However, the "relational stage" in van Steenkiste et al. [30] has a computational complexity of $\mathcal{O}(K^2)$ compared to the $\mathcal{O}(K)$ complexity of the autoregressive prior of GENESIS.

3.2 GENESIS

In order to learn object-centric representations, GENESIS adopts a latent variable formulation as in Equation (3.2). Given that we would like to both decompose a given image as well as generate synthetic images in an object-centric fashion, this requires the formulation of appropriate inference and generative models. Crucially, while prior art in Burgess et al. [25] and Greff et al. [26] assumes that individual components are independent in the generative model, i.e. $p_\theta(\mathbf{z}_{1:K}) = \prod_k p_\theta(\mathbf{z}_k)$, we instead introduce an autoregressive prior to model correlations between scenes components, which can be written as:

$$p_\theta(\mathbf{z}_{1:K}) = \prod_{k=1}^{K} p_\theta(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})\,. \tag{3.3}$$

This enables GENESIS to generate entire coherent scenes which is necessary if we want to use it as a world model as in Ha and Schmidhuber [29].

3.2.1 Generative Model

We begin by defining the generative model. Consider an image $\mathbf{x}$ of height H, width W, with C channels, and pixel values in the interval $[0, 1]$ to be a three-dimensional tensor $\mathbf{x} \in [0, 1]^{H \times W \times C}$. This chapter is only concerned with RGB images where $C = 3$ but other input modalities with a different number of channels, for example depth images, could also be considered. We would like scene components to correspond to object-like entities and we would like to be able to assign each pixel in an image to one of these components. This can be formulated as obtaining a set of segmentation masks $\pi \in [0, 1]^{H \times W \times K}$ with $\sum_k \pi_{i,j,k} = 1$ for all pixel coordinate tuples (i, j) in an image. To achieve this, we follow Burgess et al. [25] and Greff et al. [26] and formulate the image likelihood $p_\theta(\mathbf{x} \mid \mathbf{z}_{1:K})$ as a Spatial Gaussian

Mixture Model (SGMM). Assuming pixels are independent and a fixed standard deviation σ_x, this can be written as:

$$p_\theta(\mathbf{x}_{i,j} \mid \mathbf{z}_{1:K}) = \sum_{k=1}^{K} \pi_{i,j,k}(\mathbf{z}_{1:K})\, \mathcal{N}\Big(\mu_{i,j}(\mathbf{z}_k),\, \sigma_x^2\Big)\,, \tag{3.4}$$

where $\mu \in [0,1]^{H \times W \times C}$ are the means of the object colours. Note that both $\mathbf{x}_{i,j}$ and $\mu_{i,j}$ are vectors of length C, corresponding the C channels at pixel location (i,j).

We choose to encode the segmentation masks π and the means μ by two separate sets of latent variables so that they can be processed by separate, dedicated sub-networks. In particular, we assume that a set of latent "mask variables" $\mathbf{z}_k^m \in \mathbb{R}^{D_m}$ encodes the segmentation masks π and that a set of latent "component variables" $\mathbf{z}_k^c \in \mathbb{R}^{D_c}$ encodes the means μ of the object colours. The image likelihood can therefore be written as:

$$p_\theta(\mathbf{x}_{i,j} \mid \mathbf{z}^m_{1:K}, \mathbf{z}^c_{1:K}) = \sum_{k=1}^{K} \pi_{i,j,k}(\mathbf{z}^m_{1:K})\, \mathcal{N}\Big(\mu_{i,j}(\mathbf{z}^c_k),\, \sigma_x^2\Big)\,. \tag{3.5}$$

To investigate whether having separate latent variables for mask and component appearances is indeed sensible, we also consider a simplified variant called GENESIS-S which only has a single set of latent variables $\mathbf{z}^s_{1:K}$.

Considering that components in visual scenes are correlated, for example two solid objects cannot occupy the same physical space or the walls of a room are usually not disconnected from the floor, we assume—in contrast to the works in Burgess et al. [25] and Greff et al. [26]—that a mask variable $\mathbf{z}_k^m$ depends on all other mask variables $\mathbf{z}^m_{<k}$ associated with components that are already present in a scene, leading to an autoregressive formulation:

$$p_\theta(\mathbf{z}^m_{1:K}) = \prod_{k=1}^{K} p_\theta\Big(\mathbf{z}^m_k \mid \mathbf{z}^m_{1:k-1}\Big)\,. \tag{3.6}$$

While Equation (3.6) does not guarantee that, e.g., physical constraints on object locations are satisfied, it enables GENESIS to represent and learn such correlations from the statistics of the training data. This is the key component that is missing in Burgess et al. [25] and Greff et al. [26], allowing the object-centric generation of entire coherent scenes and rendering GENESIS suitable as world model for data-efficient

robotic skill acquisition. Specifically, new scenes can be generated by first sampling $\mathbf{z}_1^m$ from $p_\theta(\mathbf{z}_1^m)$ which can either be learned or fixed and then the remaining $\mathbf{z}_k^m$ are sampled from $p_\theta\left(\mathbf{z}_k^m \mid \mathbf{z}_{1:k-1}^m\right)$ in an autoregressive fashion until a certain stopping criterion is reached. In practice, we run this for a fixed number of K steps.

Following the adoption of two sets of latent variables, $\mathbf{z}_{1:K}^m$ and $\mathbf{z}_{1:K}^c$, we also need to model the correlation between them so that the appearance of a component matches the space that it occupies in an image, i.e. pixel values associated with an objects depend its the spatial location as encoded by $\mathbf{z}_k^m$. Ignoring surface reflectance, however, the appearance of several objects is largely independent. Consequently, we assume that latent component variables are conditionally independent given $\mathbf{z}_k^m$:

$$p_\theta(\mathbf{z}_{1:K}^c \mid \mathbf{z}_{1:K}^m) = \prod_{k=1}^{K} p_\theta(\mathbf{z}_k^c \mid \mathbf{z}_k^m)\,. \tag{3.7}$$

Based on these assumptions, the complete generative model of GENESIS can be written as:

$$p_\theta(\mathbf{x}) = \iint p_\theta(\mathbf{x} \mid \mathbf{z}_{1:K}^c, \mathbf{z}_{1:K}^m)\, p_\theta(\mathbf{z}_{1:K}^c \mid \mathbf{z}_{1:K}^m)\, p_\theta(\mathbf{z}_{1:K}^m)\, \mathrm{d}\mathbf{z}_{1:K}^m\, \mathrm{d}\mathbf{z}_{1:K}^c\,. \tag{3.8}$$

An analogous expression can be written for GENESIS-S:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}_{1:K}^s)\, p_\theta(\mathbf{z}_{1:K}^s)\, \mathrm{d}\mathbf{z}^s\,. \tag{3.9}$$

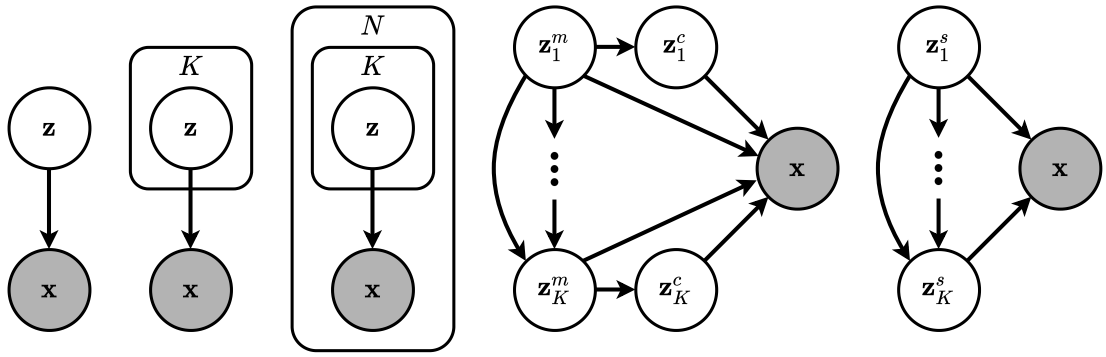
Following Burgess et al. [25] and Greff et al. [26], we assume the individual terms in the factorised prior distribution to be diagonal Gaussians. Any unconditional terms are assumed to have a mean of zero and unit variance. The resulting Probabilistic Graphical Models (PGMs) of GENESIS and GENESIS-S are shown in comparison to MONET, IODINE, and a standard VAE in Figure 3.1.

Finally, we need to ensure that the segmentation masks π in Equation (3.4) sum to one for each pixel. To this end, we follow Burgess et al. [25] and utilise a Stick-Breaking Process (SBP), which can be written as:

$$\pi_1 = \alpha(\mathbf{z}_1^m)\,, \tag{3.10}$$

$$\pi_k = \left(1 - \sum_{k'=1}^{k-1} \pi_{k'}\right) \odot \alpha(\mathbf{z}_k^m)\,, \tag{3.11}$$

$$\pi_K = \left(1 - \sum_{k'=1}^{K-1} \pi_{k'}\right)\,, \tag{3.12}$$



(a) VAE (b) MONET (c) IODINE (d) GENESIS (e) GENESIS-S

Figure 3.1: Graphical model of GENESIS and GENESIS-S compared to other methods. N denotes the number of refinement iterations in IODINE.

where subscripts for pixel coordinates are omitted and $\alpha \in [0,1]^{H \times W}$ denotes the unnormalised "alpha mask" obtained after decoding an individual mask variable. Alternatively, it would also be possible to normalise the masks with a softmax operation.

3.2.2 Inference Model

The true latent posterior distribution $p(\mathbf{z} \mid \mathbf{x})$ is generally intractable. Instead, we employ a variational approximation $q_\phi(\mathbf{z} \mid \mathbf{x})$ with parameters ϕ (Kingma and Welling [27] and Rezende et al. [28]). We amortise inference of the approximate posterior by reparameterising the latent variables as in Kingma and Welling [27] and Rezende et al. [28] and predict the sufficient statistics of the approximate posterior as a function of the input $\mathbf{x}$ using a dedicated inference network. As for the generative model, the individual terms in the joint posterior distribution are also assumed to be Gaussian. Mirroring the generative model, we assume that the posterior mask variables depend on the mask variables belonging to components that are already present in a scene:

$$q_\phi(\mathbf{z}^m_{1:K} \mid \mathbf{x}) = \prod_{k=1}^{K} q_\phi\Big(\mathbf{z}^m_k \mid \mathbf{x}, \mathbf{z}^m_{1:k-1}\Big)\,. \tag{3.13}$$

Thus, the inference model is effectively required to learn to sequentially "look" at different parts of an image one at a time. Similarly, we assume that the posterior

component variables are conditionally independent given the associated segmentation masks π_k, which are themselves a function of $\mathbf{z}^m_{1:k}$ as in the generative model:

$$q_\phi(\mathbf{z}^c_{1:K} \mid \mathbf{x}, \mathbf{z}^m_{1:K}) = \prod_{k=1}^{K} q_\phi(\mathbf{z}^c_k \mid \mathbf{x}, \mathbf{z}^m_{1:k}) \,. \tag{3.14}$$

Intuitively, the model therefore first needs to decide "where" to look and then figure out "what" it is looking at in a certain region of an image. Combining these two assumptions, the full posterior distribution can be written as:

$$q_\phi(\mathbf{z}^c_{1:K}, \mathbf{z}^m_{1:K} \mid \mathbf{x}) = q_\phi(\mathbf{z}^m_{1:K} \mid \mathbf{x})\, q_\phi(\mathbf{z}^c_{1:K} \mid \mathbf{x}, \mathbf{z}^m_{1:K}) \,. \tag{3.15}$$

For GENESIS-S, $\mathbf{z}^s_k$ is treated like $\mathbf{z}^m_k$ in the first term in Equation (3.15) and the second term can be dropped.

3.2.3 Learning

Now that we have defined the generative and the inference models, GENESIS can be trained as a VAE as in Kingma and Welling [27] and Rezende et al. [28]. The flow of information in the resulting autoencoder architectures is illustrated in Figures 3.2a and 3.2b for GENESIS and GENESIS-S, respectively. Instantiations of these models can be trained by optimising one of several alternative learning objectives. The standard method is to train VAEs by maximising the Evidence Lower Bound (ELBO) on the marginal log-likelihood $\log p_\theta(\mathbf{x})$ (Kingma and Welling [27] and Rezende et al. [28]).[2] Omitting the subscripts with respect to the component indices, the ELBO for GENESIS can be written as:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}^c, \mathbf{z}^m \mid \mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z}^c, \mathbf{z}^m) p_\theta(\mathbf{z}^c, \mathbf{z}^m)}{q_\phi(\mathbf{z}^c, \mathbf{z}^m \mid \mathbf{x})}\right] \tag{3.16}$$

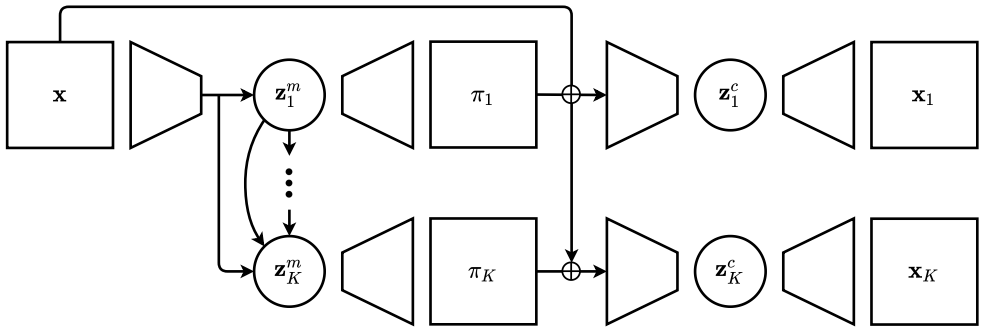
$$= \mathbb{E}_{q_\phi(\mathbf{z}^c, \mathbf{z}^m \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z}^c, \mathbf{z}^m)] - \mathrm{KL}\left(q_\phi(\mathbf{z}^c, \mathbf{z}^m \mid \mathbf{x}) \,||\, p_\theta(\mathbf{z}^c, \mathbf{z}^m)\right) \,. \tag{3.17}$$

Again, a similar expression can be derived for GENESIS-S:

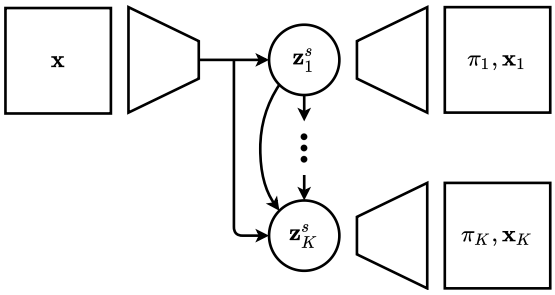
$$\mathcal{L}_{\text{ELBO}}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}^s \mid \mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z}^s) p_\theta(\mathbf{z}^s)}{q_\phi(\mathbf{z}^s \mid \mathbf{x})}\right] \tag{3.18}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}^s \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z}^s)] - \mathrm{KL}\left(q_\phi(\mathbf{z}^s \mid \mathbf{x}) \,||\, p_\theta(\mathbf{z}^s)\right) \,. \tag{3.19}$$

[2] When training neural networks, we typically talk about minimising a training loss, i.e. we would minimise the negative marginal likelihood. To avoid talking about a "negative lower bound", we stick with the view that we want to maximise the marginal likelihood in this section.



(a) GENESIS



(b) GENESIS-S

Figure 3.2: GENESIS and GENESIS-S autoencoder architectures. Illustration does not include the prior networks. Given an image $\mathbf{x}$, GENESIS infers mask latent variables $\mathbf{z}^m_{1:K}$ in a sequential fashion. These are decoded in parallel into the segmentation masks $\pi_{1:K}$. The masks concatenated with the image to infer the component latent variables $\mathbf{z}^c_{1:K}$ and to reconstruct the image components $\mathbf{x}_{1:K}$. Similarly, GENESIS-S infers the latent variables $\mathbf{z}^s_{1:K}$ in a sequential fashion, which encode both the segmentation masks $\pi_{1:K}$ and the image components $\mathbf{x}_{1:K}$.

The first term in Equations (3.17) and (3.19) corresponds to the pixel likelihood—or the reconstruction loss in the language of autoencoders—and the second term is the KL divergence between the approximate latent posterior and the prior. This second term regularises the reconstructions by encouraging the amortised posterior to be similar to the prior, hence facilitating the generation of novel images by sampling from the prior distribution. We approximate the expectation by drawing a single Monte Carlo (MC) sample from the posterior distributions $q_\phi(\mathbf{z} \mid \mathbf{x})$ (Kingma and Welling [27] and Rezende et al. [28]). It should be noted that MONET from Burgess et al. [25] is actually not trained with a valid ELBO, but with a loss function that resembles an ELBO. Yet, it would still be possible to compute an ELBO with MONET by using the reconstructed masks rather than the masks from the attention network for reconstructing an image.

In practice, the objective in Equation (3.17) above does not necessarily lead to the desired trade-off between the two loss terms that govern reconstruction quality and sample fidelity. Thus, it has become common practice to re-weight the KL divergence term by a scalar value β to tune this trade-off as per task requirements (Higgins et al. [100]):

$$\mathcal{L}_\beta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] \; - \; \beta \cdot \mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \;||\; p_\theta(\mathbf{z})\right) . \tag{3.20}$$

This has been extended by designing curriculum-like schedules for β to further improve the optimisation procedure (Sønderby et al. [101]). While these schemes can achieve reasonable empirical results, they involve a somewhat unintuitive tuning of the schedule for β and lack a theoretical justification. As an alternative, Rezende and Viola [102] propose an optimisation objective called Generalised ELBO with Constrained Optimisation (GECO) which is both intuitive and justified from a theoretical perspective. GECO involves setting a goal value $C \in \mathbb{R}$ for the reconstruction error which can be tuned manually by considering the reconstruction quality for different goal values. Given a specific "reconstruction goal", GECO minimises the KL divergence subject to a constraint on the reconstruction quality:

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} \mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \;||\; p_\theta(\mathbf{z})\right), \tag{3.21}$$

$$\text{s.t.} \quad \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] \geq C \, . \tag{3.22}$$

This optimisation can be performed by incorporating the constraint as a Lagrange multiplier λ. This leads to the same training objective as in Equation (3.20) with $\beta = \lambda^{-1}$. Note that the constraint is on the reconstruction quality so we need to take the inverse if we want to re-weight the KL divergence in the training objective. β is updated according to the moving average E of the reconstruction error:

$$E = \alpha_g \cdot \epsilon + (1 - \alpha_g) \cdot \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] \, , \tag{3.23}$$

$$\beta = \beta \cdot e^{\eta(E-C)} \, , \tag{3.24}$$

where $\alpha_g \in [0, 1]$ and $\eta \in \mathbb{R}^+$ are fixed momentum and step size hyperparameters, respectively. Intuitively, when the reconstructions are poor at the beginning of

training, the constraint is not satisfied and $E < C$. This causes β to become smaller, decreasing the effect of the KL regularisation and putting a larger emphasis on the reconstruction loss. As training progresses, the reconstructions improve so that the constraint is eventually satisfied, i.e. $E > C$, and β becomes larger. This increases the emphasis on KL minimisation to maximise sample quality at a reconstruction quality of $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] \approx C$.

3.3 Experiments

To verify that GENESIS is suitable for unsupervised object discovery and as an object-centric world model, we evaluate its ability to decompose scenes into object-like entities (Section 3.3.5) and to generate entire visually plausible scenes in a object-centric fashion (Section 3.3.6) both quantitatively and qualitatively. Additionally, to better understand and characterise the properties of GENESIS, we probe the information in the learned object representations in a set of semi-supervised tasks (Section 3.3.7) and we also conduct an investigation of "reconstruction bottlenecks", showing that these critically determine model performance (Section 3.3.8). Before presenting the empirical results, this section proceeds by introducing the datasets (Section 3.3.1) and metrics (Section 3.3.2) used for evaluation, by providing the architecture specifications (Section 3.3.3), and by describing the training details (Section 3.3.4).

3.3.1 Datasets

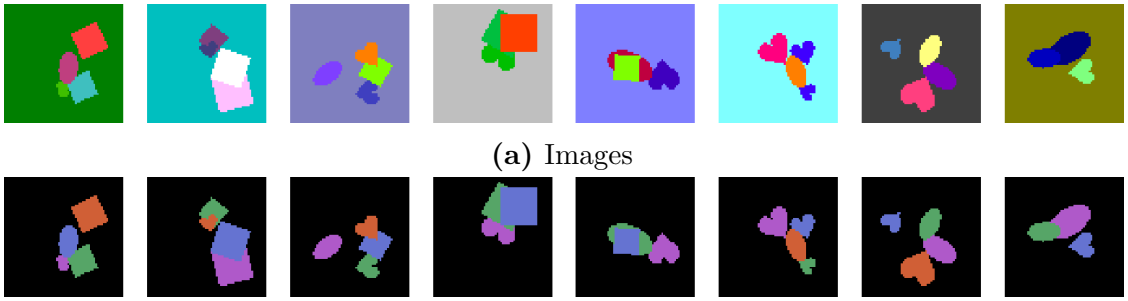
Prior art in Burgess et al. [25] and Greff et al. [26] is evaluated on the Multi-dSprites (Burgess et al. [25]), Tetrominoes Greff et al. [26], CLEVR (Johnson et al. [99]), and ObjectsRoom (Burgess et al. [25]) datasets. Multi-dSprites and Tetrominoes are visually simple datasets with 2D structure; CLEVR and ObjectsRoom contain rendered images of simulated 3D environments and are therefore closer to real-world applications. Apart from CLEVR, the remaining datasets were not publicly available at the time when our work on the results in Sections 3.3.5 to 3.3.7 began. To allow a comparison with the prior work in Burgess et al. [25] and Greff et al.

[26], we generate our own version of the Multi-dSprites dataset. This is the visually simplest of these four datasets and was therefore selected as a means for verifying basic model functionality. Furthermore, we conduct experiments with the GQN dataset (Eslami et al. [87]) which is visually similar to ObjectsRoom and more interesting than the remaining two datasets; Tetrominoes only has 2D structure and all CLEVR images have the same plain white background. To go beyond the experiments in Burgess et al. [25] and Greff et al. [26], we also conduct experiments on the more challenging ShapeStacks dataset (Groth et al. [88]). At a later point in time, Kabra et al. [103] released official versions of the Multi-dSprites, Tetrominoes, CLEVR, and ObjectsRoom datasets as used in Burgess et al. [25] and Greff et al. [26]. We use the ObjectRoom dataset released along with Kabra et al. [103] for additional experiments in Section 3.3.8. Similar to Burgess et al. [25] and Greff et al. [26], we use a resolution of 64-by-64 for all datasets. The datasets are described in more detail in the following.

Multi-dSprites (Burgess et al. [25]) Images from the Multi-dSprites dataset consist of one to four coloured two-dimensional "sprites" super-imposed on a uniformly coloured background. These sprites are obtained from the "dSprites" dataset (Matthey et al. [104]) which were originally introduced for the purpose of evaluating the ability of unsupervised models to "disentangle" the ground truth factors of variations in a dataset.[3] Using these sprites, we generate a dataset of 50,000 training images; 10,000 validation images; and 10,000 test images with values for the three colour channels being sampled uniformly from the set $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Example images are shown in Figure 3.3.[4] When training GENESIS and our re-implementation of MONET, we set the maximum number of scene components to $K = 5$ as there are up to four foreground sprites and a single background component.

[3]The dSprites dataset is available at `https://github.com/deepmind/dsprites-dataset`.

[4]Our script for generating the Multi-dSprites dataset is available at `https://github.com/applied-ai-lab/genesis`. An "official" version of the Multi-dSprites dataset is now also available at `https://github.com/deepmind/multi_object_datasets` (Kabra et al. [103]).



(a) Images

(b) Ground truth segmentation masks

Figure 3.3: Example images from the re-implemented Multi-dSprites dataset.

GQN (Eslami et al. [87]) Images from the GQN dataset correspond to different camera viewpoints capturing a simulated three-dimensional environment that contains one to three objects in a square room. These objects vary in terms of colour, shape, and size and the room appearance varies in terms of floor and wall textures. We choose to use the most basic "rooms-ring-camera" dataset in which viewpoints correspond to a camera hovering in a circle above the room. There are 10,800,000 training images of which 180,000 are set aside for validation and 1,200,000 test images. Example images are shown in Figure 3.4.[5] Ground truth segmentations of the foreground objects are not available for this dataset. We set the maximum number of scene components to $K = 7$ as there are up to three foreground objects per image and up to four background components: two walls, the sky, and the floor.

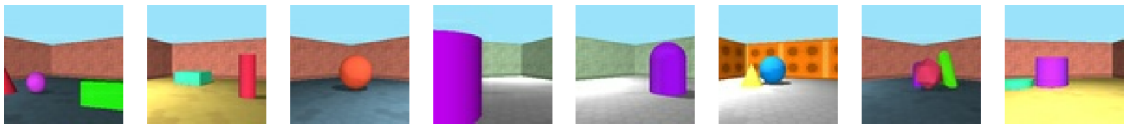
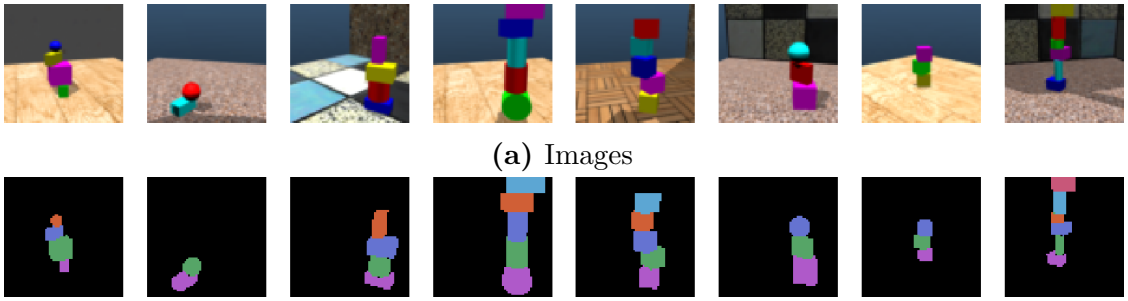


Figure 3.4: Example images from the GQN dataset (Eslami et al. [87]).

ShapeStacks (Groth et al. [88]) Images from the ShapeStacks dataset depict towers of block-like objects on a flat surface in a square room. Each tower consists of at least two blocks and can be up to six blocks tall. Some of these towers are statically stable while others are unstable. Scenes vary in terms of the colour, shape, and size of the foreground objects; textures of the floor and the walls; lighting

[5]The GQN dataset is available at `https://github.com/deepmind/gqn-datasets`.



(a) Images

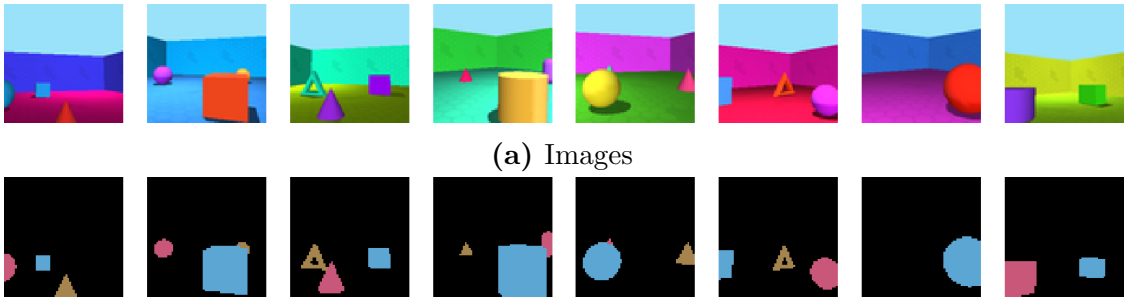
(b) Ground truth segmentation masks

Figure 3.5: Example images from the ShapeStacks dataset (Groth et al. [88]) after cropping and resizing.

conditions; and camera viewpoints. Annotations for these factors of variation are provided which make this dataset well suited for semi-supervised experiments (Section 3.3.7). The raw images have a resolution of 224-by-224. A centre crop of size 196 is extracted before resizing the images to a resolution of 64-by-64 using bilinear interpolation. There are 217,888; 46,656; and 46,656 images in the training, validation, and test sets, respectively. Example images are shown in Figure 3.5.[6] For ShapeStacks, we use $K = 9$ in all experiments as there are up to six blocks in a tower and up to three background components: two walls and the floor.

ObjectsRoom (Burgess et al. [25] and Kabra et al. [103]) The ObjectsRoom dataset is very similar to the GQN dataset. The main visual differences are different floor and wall textures. In contrast to the GQN dataset, ground truth segmentation masks of the foreground objects are available for the ObjectsRoom dataset. The official "training" set consists of 1,000,000 images. 10,000 images are set aside from this split for validation and testing each. We do not use the official evaluation sets as they are designed to test model generalisation to out-of-distribution images, for example empty scenes or scenes with more objects than encountered during training. Example images are shown in Figure 3.6.[7] As for the GQN dataset, the maximum number of scene components is set to $K = 7$ as the ObjectsRoom dataset has the same main characteristics.

[6]The ShapeStacks dataset is available at `https://shapestacks.robots.ox.ac.uk/`.
[7]The ObjectsRoom dataset is available at `https://github.com/deepmind/multi_object_datasets`.



(a) Images

(b) Ground truth segmentation masks

Figure 3.6: Examples from the ObjectsRoom dataset (Kabra et al. [103]).

3.3.2 Evaluation Metrics

To assess the suitability of GENESIS for unsupervised object discovery and as an object-centric world model, we need to evaluate both the quality of the object segmentations and the quality of generated scenes. The former can be achieved by comparing the predicted segmentations to annotated ground truth segmentations, if available. Instance segmentation methods are commonly evaluated using the Average Precision (AP) based on an individual or a range of Intersection Over Union (IOU) thresholds (e.g. (He et al. [105])). However, computing the AP requires some form of confidence scores which are not available in this unsupervised setup. Following Greff et al. [26], we therefore use the Adjusted Rand Index (ARI) (Hubert and Arabie [106]) of pixels belonging to foreground objects to evaluate the quality of the unsupervised scene segmentations with respect to ground truth labels. We observe, however, that the ARI does not penalise the undersegmentation of foreground objects, which is later demonstrated in Figure 3.10. Inspired by Arbelaez et al. [107], we consequently propose the Mean Segmentation Covering (MSC) of foreground objects as an additional metric. For evaluating the quality of generated scenes, we utilise the Fréchet Inception Distance (FID) (Heusel et al. [108]) as commonly employed in the literature on GANs (Goodfellow et al. [31]). In the following, we describe these three metrics in more detail.

Adjusted Rand Index (ARI) Hubert and Arabie [106] consider the problem of quantitatively assessing the similarity of two different partitions, U and V, of a set

of elements S. Intuitively, the ARI considers whether all possible pairs of elements in S have been grouped into the same or different classes in U and V, respectively. As such, the ARI is a symmetric measure, i.e. $\text{ARI}(U, V) = \text{ARI}(V, U)$. The ARI as proposed in Hubert and Arabie [106] is normalised to fall in the interval $[-1, 1]$ and it is adjusted so that random partitions give a score close to zero.

Mean Segmentation Covering (MSC) Considering ground truth masks U and predicted masks V, we compute the MSC as the mean segmentation covering of U by V:

$$\text{MSC}(U \leftarrow V) = \frac{1}{|U|} \sum_{u \in U} \max_{v \in V} \text{IOU}(u, v), \tag{3.25}$$

whereby u and v are individual object masks in U and V, respectively, and $|U|$ is the number of ground truth foreground objects. In contrast to the ARI, the MSC is not a symmetric metric. The original formulation in Arbelaez et al. [107] weights IOU scores by the number of pixels belonging to the reference mask u. Analogous to for example the AP for evaluating object detection methods, we argue the metric should be independent of object size. We therefore take an unweighted mean over the IOU scores of different segmentation pairs instead.

Fréchet Inception Distance (FID) The FID is computed by comparing the statistics of the feature maps from real and generated images, extracted by an inception-style Convolutional Neural Network (CNN) (Szegedy et al. [109]). This CNN is trained on the ImageNet dataset (Russakovsky et al. [10]) which shows real-world images of centred objects from 1,000 different categories.[8] Even though the distribution of images from ImageNet is quite different to the images considered in this chapter, we observe in Section 3.3 that qualitatively better generated images indeed correlate with smaller FID scores.

[8]We use the PyTorch implementation available at `https://github.com/mseitzer/pytorch-fid` for computing FID scores.

3.3.3 Architecture Details

After discussing suitable metrics for conducting a quantitative evaluation, this section describes the architecture implementation details of GENESIS, GENESIS-S, and several baselines. We primarily compare scene decomposition and generation quality to our own re-implementation of MONET (Burgess et al. [25]).[9] Although relevant, we do not compare to IODINE (Greff et al. [26]) as the authors report that training on CLEVR images (Johnson et al. [99]) with a resolution of 128-by-128 takes about one week on eight NVIDIA V100 GPUs compared to training GENESIS and MONET on a single GPU within a few days on the datasets considered in this chapter. Such large computational requirements are rather impractical for real-world applications where practitioners might want to frequently re-train models with new data. To assess the quality of generated scenes in comparison to more conventional reference points, we also consider two standard VAEs as additional baselines. The same architecture details are across datasets. Qualitative and quantitative results can likely be improved at loss of generality by specifically tuning various hyperparameters for each dataset. However, this should not significantly affect the conclusions: unlike MONET and IODINE, GENESIS is designed to facilitate object-centric scene generation to make it suitable as a world model for learning robotic skills. The purpose of the experiments is therefore to show that (1) GENESIS is able to achieve commensurate scene decomposition performance to prior work, (2) that GENESIS is indeed able to generate entire coherent scenes in contrast to prior work, and (3) to conduct further experiments to better understand the attributes of GENESIS.

GENESIS Following the description of GENESIS in Section 3.2, the following neural network modules need to be specified: the image encoder for mask variables, the posterior Recurrent Neural Network (RNN) for mask variables, the decoder for masks, the image encoder for component variables, the associated decoder, the autoregressive prior RNN for mask variables, and the conditional prior for

[9] Our re-implementation of MONET (Burgess et al. [25]) is also available at `https://github.com/applied-ai-lab/genesis`.

Table 3.1: GENESIS mask encoder.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
64	3	64	5×5	1	Conv	BN-GLU
64	32	64	5×5	2	Conv	BN-GLU
32	32	128	5×5	1	Conv	BN-GLU
32	64	128	5×5	2	Conv	BN-GLU
16	64	128	5×5	1	Conv	BN-GLU
-	16×16×64	512	-	-	FC	GLU
-	256	$2\times D_m$	-	-	FC	-

Table 3.2: GENESIS mask decoder.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
-	D_m	16×16×128	-	-	FC	GLU
16	64	128	5×5	1	Deconv	BN-GLU
16	64	64	5×5	2	Deconv	BN-GLU
32	32	64	5×5	1	Deconv	BN-GLU
32	32	64	5×5	2	Deconv	BN-GLU
64	32	64	5×5	1	Deconv	BN-GLU
64	32	1	1×1	1	Conv	-

component variables. The convolutional mask encoder (Table 3.1) and decoder (Table 3.2) are inspired by Berg et al. [110]. This is selected based on its good empirical performance in the context of normalising flows (Rezende and Mohamed [111]) on datasets of comparable visual complexity. The encoder and decoder consist of five convolutional and deconvolutional layers, respectively, followed by non-linear Gated Linear Unit (GLU) activations (Dauphin et al. [112]). The only difference to Berg et al. [110] is that we apply Batch Normalisation (BN) (Ioffe and Szegedy [113]) before the GLU activations as suggested in Sønderby et al. [101]. This was found to stabilise and accelerate training in preliminary experiments. After encoding the images, the posterior statistics of the mask variables of the first object are directly computed with a fully-connected linear layer. For later components, the image encoding is concatenated with the mask variables from the previous step and passed into a Long Short-Term Memory (LSTM) cell (Hochreiter and Schmidhuber [114]),

Table 3.3: GENESIS mask posterior RNN.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
-	$256+D_m$	128	-	-	LSTM	-
-	128	$2\times D_m$	-	-	FC	-

Table 3.4: GENESIS component encoder.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
64	3+1	32	3×3	2	Conv	ELU
32	32	32	3×3	2	Conv	ELU
16	32	64	3×3	2	Conv	ELU
8	64	64	3×3	2	Conv	ELU
-	4×4×64	256	-	-	FC	ELU
-	256	$2\times D_c$	-	-	FC	-

whose memory keeps track of the history of previous mask variables. The output of the LSTM is then passed into a fully-connected linear layer to produce the posterior statistics of the remaining mask variables (Table 3.3) one step at a time. Finally, the mask variables are decoded in parallel with a deconvolutional decoder to produce the alpha masks for the SBP (Section 3.2.1). All experiments use a mask latent dimension of $D_m = 64$, i.e. $\mathbf{z}_k^m \in \mathbb{R}^{64}$. Component encoder and decoder specifications (Tables 3.1 and 3.5) are identical to MONET (Burgess et al. [25]) with the only difference of employing Exponential Linear Units (ELUs) (Clevert et al. [63]) instead of Rectified Linear Units (RELUs) (Glorot et al. [40]) as non-linear activation functions. This appeared to accelerate training in preliminary experiments. In particular, MONET and GENESIS use a Spatial Broadcast Decoder (SBD) as proposed in Watters et al. [115], which is designed to improve the "disentanglement" of the underlying factors of variation in the training data. The main experiments use a component latent dimension of $D_c = 16$, i.e. $\mathbf{z}_k^c \in \mathbb{R}^{16}$. The autoregressive prior on the mask variables is also implemented as an LSTM (Table 3.6), but the number of hidden units is doubled compared to the mask posterior LSTM, as the latter has access to the current image encoding as an additional input. Lastly,

Table 3.5: GENESIS component decoder.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
-	D_c	$2+D_c$	-	-	Broadcast	-
72	$2+D_c$	32	3×3	1	Conv	ELU
70	32	32	3×3	1	Conv	ELU
68	32	32	3×3	1	Conv	ELU
66	32	32	3×3	1	Conv	ELU
64	32	3	1×1	1	Conv	-

Table 3.6: GENESIS mask prior RNN.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
-	D_m	256	-	-	LSTM	-
-	256	$2\times D_m$	-	-	FC	-

the prior on component variables takes as input the associated mask variables and is implemented as a feed-forward MLP with two hidden layers (Table 3.7). In contrast to GENESIS, GENESIS-S only has a single set of latent variables $\mathbf{z}_k^s$ that encode both component masks and appearances. These are encoded with the same convolutional encoder and autoregressive posterior network as used for the mask variables in GENESIS (Tables 3.1 and 3.3). Two separate decoders are used for masks variables (Table 3.2) and component variables (Table 3.5). The autoregressive prior for $\mathbf{z}_k^s$ is implemented with the same architecture as for $\mathbf{z}_k^m$ (Table 3.6).

MONET We compare the decomposition and generation performance of GENESIS to our own re-implementation of MONET (Burgess et al. [25]). The architecture details in Burgess et al. [25] are followed as closely as possible. Two aspects, however, are not fully specified: the numbers of filters per layer in the UNet-like attention network (Ronneberger et al. [116]) and how reconstructed masks are normalised. To approximately match the attention network specifications of GENESIS, we use [32, 32, 64, 64, 64] filters in the convolutional encoder of the attention network and the reverse in the decoder. The skip connections of the UNet are implemented by concatenating the features from the lower resolutions. Rather than employing

Table 3.7: GENESIS component prior Multilayer Perceptron (MLP).

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
-	D_m	256	-	-	FC	ELU
-	256	256	-	-	FC	ELU
-	256	$2 \times D_c$	-	-	FC	-

an SBP, the reconstructed masks are normalised using a softmax. Preliminary experiments indicated that this trains in a more stable fashion. Given that Burgess et al. [25] do not provide any quantitative results, a rigorous verification of our re-implementation is not possible apart from ensuring that qualitative results look reasonable. For the purposes of this investigation, however, an exact reproduction of results is not necessary as our objectives are to (1) verify that GENESIS achieves comparable and not necessarily better scene decomposition performance and (2) illustrate that scenes generated by MONET are incoherent, which will not be affected by minor implementation differences.

VAEs In the sections on image generation (Section 3.3.6) and semi-supervised learning (Section 3.3.7), GENESIS and MONET are compared to two VAE baselines: a Spatial Broadcast Decoder VAE (SBD-VAE) and a Deconvolutional VAE (DC-VAE). Both variants have the same convolutional encoder as in Berg et al. [110], without the additional BN layers as used for GENESIS (Table 3.1). The SBD-VAE uses the same component decoder as GENESIS (Table 3.5) and the DC-VAE uses the deconvolutional decoder as used for the masks in GENESIS, again without the additional BN layers (Table 3.2).

3.3.4 Training Details

All models are trained by minimising the GECO Rezende and Viola [102] as described in Section 3.2.3. The standard deviation the pixel likelihood in Equation (3.4) is set to $\sigma_x = 0.7$. The value used for σ_x is notably larger than in Burgess et al. [25] and Greff et al. [26]. Instead, it is set to the final value of σ_x in Eslami et al. [87] at the end of their annealing schedule. In a regular VAE, modifying σ_x merely changes

the relative weighting of the pixel likelihood and the KL divergence. With a spatial Gaussian mixture formulation as in Equation (3.4), however, changing σ_x affects the reconstruction likelihood in a non-linear fashion, unless the segmentation masks are binary, i.e. they are exactly zero or one. Preliminary experiments indicated that the models start differentiating individual objects earlier during training with a slightly larger value of σ_x than used in Burgess et al. [25] and Greff et al. [26], but a more rigorous empirical investigation would be required to substantiate this. For comparability, the re-implementation of MONET was also trained with this larger value of σ_x, rather than the original setting in Burgess et al. [25].

The GECO reconstruction goal is set to the summed negative log-likelihood value per pixel and per channel of 0.5655. This reconstruction goal was tuned manually and set so as to produce reconstructions of reasonable visual quality. Regarding the two GECO hyperparameters, the default $\alpha = 0.99$ is used as in Rezende and Viola [102] and the step size for updating β is set to 10^{-5}. This comparatively large step size causes β to become very small early during training and it is therefore clamped to a minimum value of 10^{-10}. This undershooting behaviour is mitigated by increasing the step size to 10^{-4} when the reconstruction constraint is satisfied. It should be highlighted that MONET is also trained using GECO and same optimisation hyperparameters for comparability. This is different to the setup in Burgess et al. [25] who optimise an objective of the form as in Equation (3.20). It would be interesting to investigate the effect of these differences, but this does not affect the conclusions of this work.

All models are trained with the Adaptive Moment Estimation (ADAM) optimiser (Kingma and Ba [117]), a learning rate of 10^{-4}, and a batch size of 32 for 500,000 iterations. On an NVIDIA Titan RTX, training GENESIS and MONET takes around two days. 500,000 training iterations with a batch size of 32 is smaller compared to Burgess et al. [25] who train for 1,000,000 iterations with a batch size of 64 and IODINE (Greff et al. [26]) who train for the same number of updates with a batch size of 32. While 500,000 iterations are not sufficient to train the models "to convergence", changes in loss are usually small at this

point and the shorted training time allows for more comprehensive experimentation given finite computational resources.

3.3.5 Unsupervised Object Discovery

While the primary innovation of GENESIS is its autoregressive scene prior for object-centric image generation, this can only work as intended if the the model is able to segment input images into object-like components. After establishing the experimental details, we now proceed to evaluate the ability of GENESIS to discover objects without supervision in comparison to our re-implementation of MONET. To this end, we present results on the GQN and ShapeStacks datasets to determine whether GENESIS is able to decompose rendered images of 3D scenes into object-like components. Overall, we find that GENESIS achieves better qualitative and quantitative results than our re-implementation of MONET. Furthermore, we also confirm that both models are able to distinguish multiple objects in an image with identical colour and we illustrate the benefit of utilising the MSC rather than the ARI for quantifying the decomposition performance.

3.3.5.1 Decomposition of GQN Images

Figure 3.7 shows a direct side-by-side comparison of the component-by-component reconstructions of GENESIS and MONET given two previously unseen images from the GQN dataset. It can be seen that both models follow a similar segmentation strategy that is also consistent across the two different input images. GENESIS first reconstructs the floor and the sky at $k = 1$, followed by the individual foreground objects at steps $k = \{2, 3, 4\}$ from right to left, and finally the two walls in the background in steps $k = \{5, 6, 7\}$. For the first input image, no dedicated scene component is segmented at $k = 5$, only an artefact at the boundary between the walls and the sky. For the second input image, the third slot dedicated to foreground objects at $k = 4$ remains empty as there are only two foreground objects in the input scene. A possible explanation for the emergence of this strategy is that the uniformly coloured floor and sky have a large number of pixels and are easy to reconstruct. The loss minimisation might therefore be driven by them early during

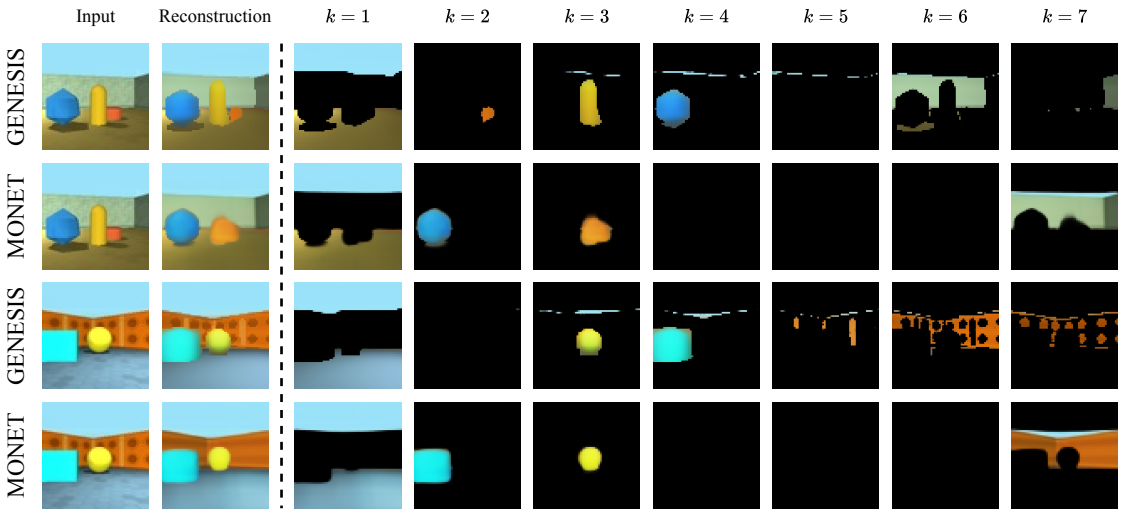


Figure 3.7: Decomposition and reconstruction of two previously unseen images from the GQN dataset by GENESIS and MONET. Both models segment the images into object-like components. Unlike MONET, GENESIS is able to segment all three foreground objects in the first example.

training and given the SBP formulation for mask prediction, the first slot receives the most direct learning signal. While MONET follows a similar strategy, in the first image it fails to disambiguate the three foreground objects: the yellow cylinder and the orange block are instead merged into a single blob at $k = 3$. For the second input image, MONET is unable to capture the dotted pattern on the walls in the background and the reconstruction instead shows blurred horizontal stripes.

Considering that both models are trained on static frames without temporal information or explicit inductive biases on 3D geometry, segmentations need to be inferred from object appearance alone. Especially for the simulated datasets as considered in this chapter, pixel colour is a very strong cue for assigning pixels to object slots. An immediate question therefore is whether GENESIS differentiates between different objects of the same colour or whether such objects are grouped into a single slot. Burgess et al. [25] present experiments to show that MONET manages to differentiate multiple objects of the same colour. Inspired by this, Figure 3.8 shows the reconstructions and decompositions by GENESIS and MONET given an image from the GQN dataset with three objects, whereby two objects have both a similar colour and a similar shape. It can be observed that both GENESIS

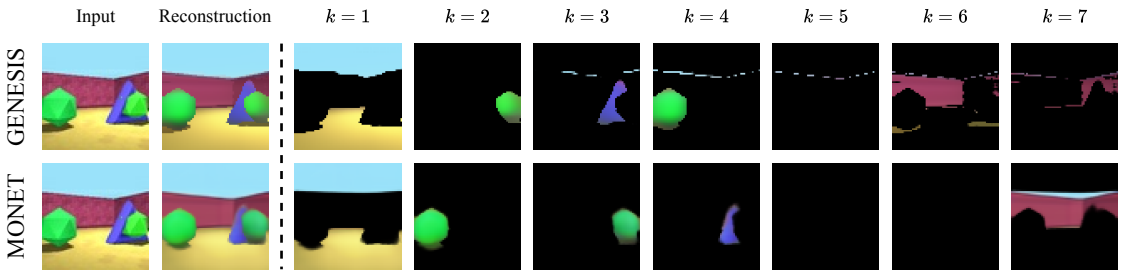


Figure 3.8: Decomposition of two previously unseen images from the GQN dataset with two objects of similar colour and shape by GENESIS and MONET. Both models manage to separate the two green shapes despite their similar appearance.

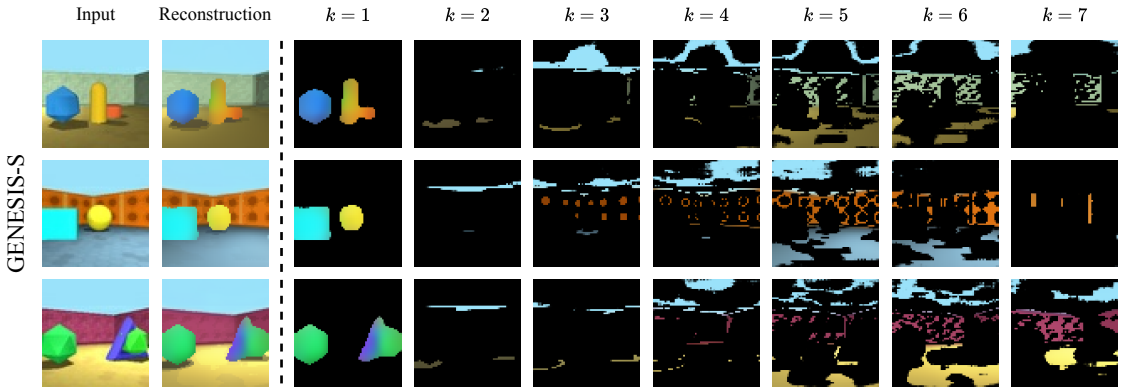


Figure 3.9: Decomposition and reconstruction of three previously unseen images from the GQN dataset by GENESIS-S. In contrast to GENESIS, GENESIS-S does not segment the images into object-like components.

and MONET differentiate the three individual objects and produce qualitatively similar results, corroborating the findings in Burgess et al. [25] and also showing that GENESIS extracts interpretable, object-centric representations for more challenging input scenes such as this one. A conjecture for explaining this behaviour is that during training, both models mostly see scenes with objects of different colours and slots therefore specialise in encoding and reconstructing individual objects. Thus, even when a scene with multiple objects with similar appearances is encountered, the models still segment out individual objects.

To evaluate whether GENESIS-S is also able to segment scenes into object-like components, Figure 3.9 shows the outputs of a trained model on the same scenes from the GQN dataset as used in Figures 3.7 and 3.8. It can be observed that all foreground objects are reconstructed at $k = 1$ and the rest of the image is reconstructed in later steps without slots corresponding to interpretable scene

components. This indicates that GENESIS-S does not exhibit suitable inductive biases for learning object-centric representations, but an extensive analysis to explain this qualitative behaviour is beyond the scope of this work.

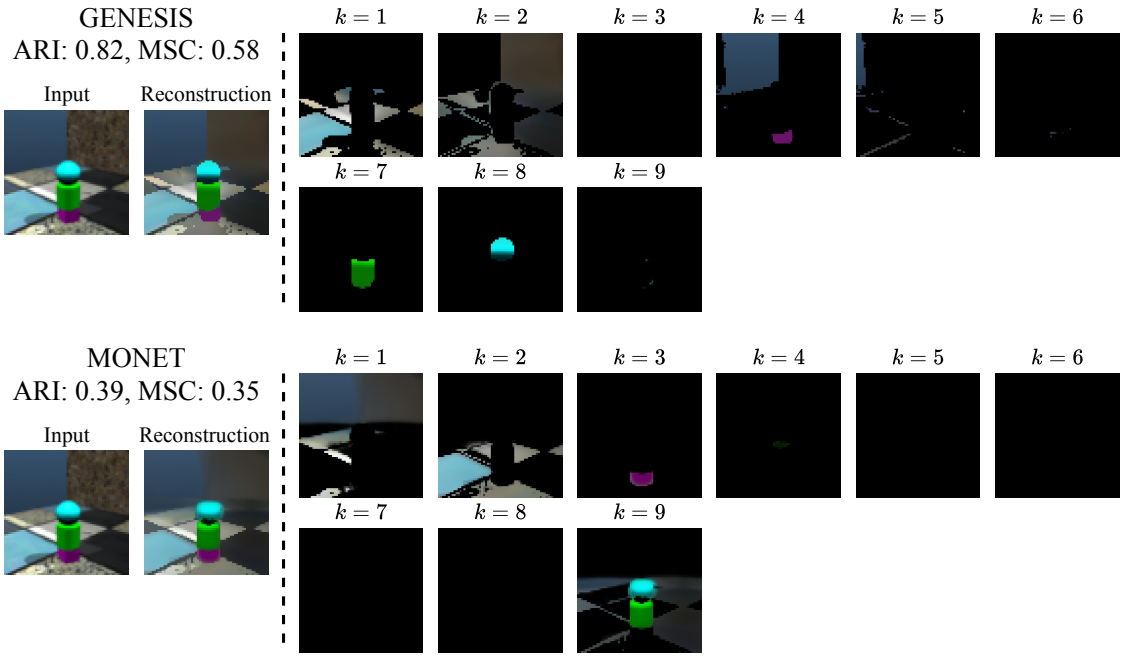
3.3.5.2 Decomposition of ShapeStacks Images

Given that ground truth segmentations are available for the ShapeStacks dataset, we follow a similar protocol as Greff et al. [26] and compute segmentation metrics on 300 test images. In the following, we report the means and standard deviations from five models trained with different random seeds. The ShapeStacks dataset was chosen for this evaluation as it is the visually most interesting dataset considered in this chapter. The results are summarised in Table 3.8, indicating that GENESIS outperforms our re-implementation of MONET both in terms of ARI and MSC.

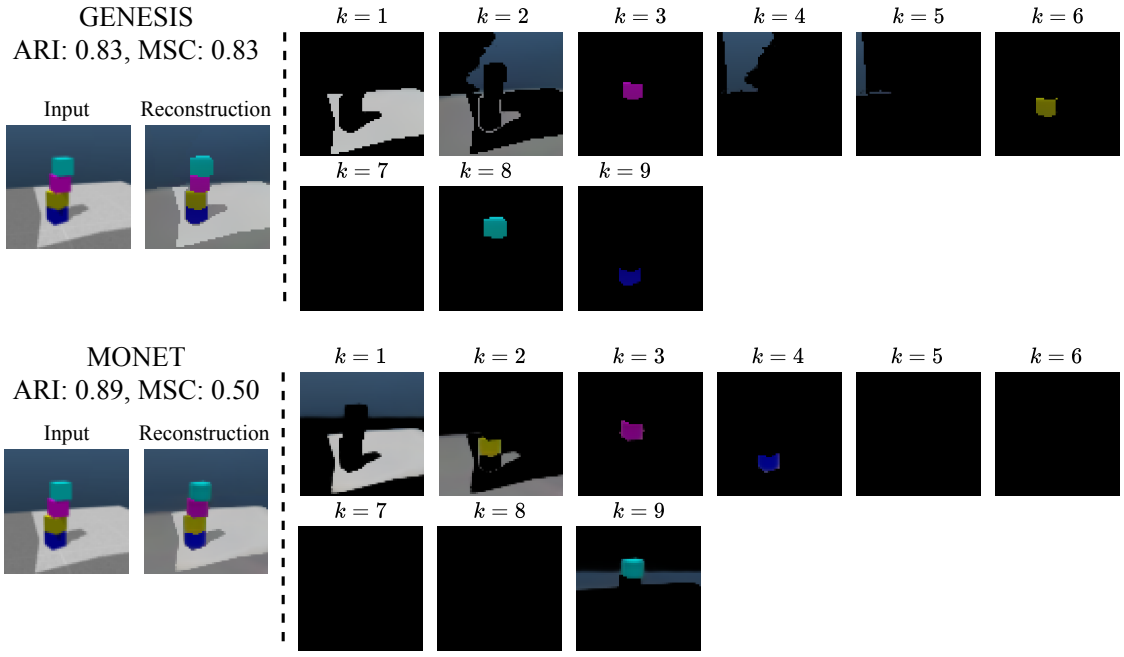
Table 3.8: Means and standard deviations of the segmentation metrics for GENESIS and MONET on ShapeStacks from five random seeds.

Model	ARI	MSC
GENESIS	0.73 ± 0.03	0.60 ± 0.09
MONET	0.63 ± 0.07	0.49 ± 0.09

To provide a visual indication of the meaning of these metrics, Figure 3.10 shows the decompositions and reconstructions of two images from the ShapeStacks dataset by GENESIS and MONET, including the quantitative scores for the ARI and MSC. In Figure 3.10a, both models struggle to perfectly separate the foreground objects. While GENESIS reconstructs objects in separate slots, the slot at $k = 4$ contains both the purple block and a part of the blue background wall. Nevertheless, GENESIS still fares better both quantitatively and qualitatively then MONET. The latter is not able to separate the green and the cyan shape; both of them are reconstructed at $k = 9$. In Figure 3.10b, GENESIS perfectly separates the foreground objects while MONET still struggles to do so. As expected, GENESIS achieves a better MSC score, but MONET achieves a better ARI score even though its decomposition is qualitatively inferior: the objects at $k = \{2, 9\}$ are reconstructed together with parts of the background. This is a consequence of the



(a) First example with a tower of height three.



(b) Second example with a tower of height four.

Figure 3.10: Decomposition and reconstruction of two previously unseen images from the ShapeStacks dataset with GENESIS and MONET. In both cases, GENESIS obtains qualitatively better results. Yet, MONET achieves a better ARI score in the second example, even though its segmentation looks qualitatively worse, illustrating a drawback of this metric.

fact that the ARI of foreground pixels does not penalise the undersegmentation of objects as discussed in Section 3.3.2.

3.3.6 Object-Centric Scene Generation

MONET (Burgess et al. [25]) and IODINE (Greff et al. [26]) are designed to decompose scenes into object-like constituent parts and both of them are formulated as VAEs so that it should in principle be possible to generate new images by sampling from the variational prior. Due to the assumption of independence between scene components in the prior, however, there is no mechanism to ensure that individual generated components actually compose a reasonable and coherent scene. In this section, we show that the autoregressive prior of GENESIS which is the principal innovation of this chapter overcomes these limitations, hence enabling the model to generate coherent scenes in an object-centric fashion. Figure 3.11 shows the component-wise generation of novel scenes with GENESIS and MONET after the models were trained on the GQN dataset, including the individual components at the intermediate generation steps. Figure 3.12 shows a larger number generated images for both models to give a better sense of the variety of generated scenes, omitting the illustration of individual components. It can be observed that GENESIS generates crisp and coherent scenes with different numbers of foreground objects. Scenes generated by MONET, however, are not coherent and distinctively different to the training data. While there are some vaguely distinguishable components such as floors and walls, component boundaries are not clearly defined, colours are blurred, and shapes are distorted. As can be seen in Figure 3.11, GENESIS follows the same strategy for generation as for inference: floor and sky are generated in the first step, followed by up to three foreground objects from right to left, and the last three components constitute different elements of the walls in the background. Even though GENESIS-S did not perform satisfactorily in scene decomposition, for completeness Figure 3.13 shows scenes generated in an object-centric fashion by GENESIS-S after training on the GQN dataset and Figure 3.14 shows a wider range of scenes, omitting the individual components. Generated scenes are fairly

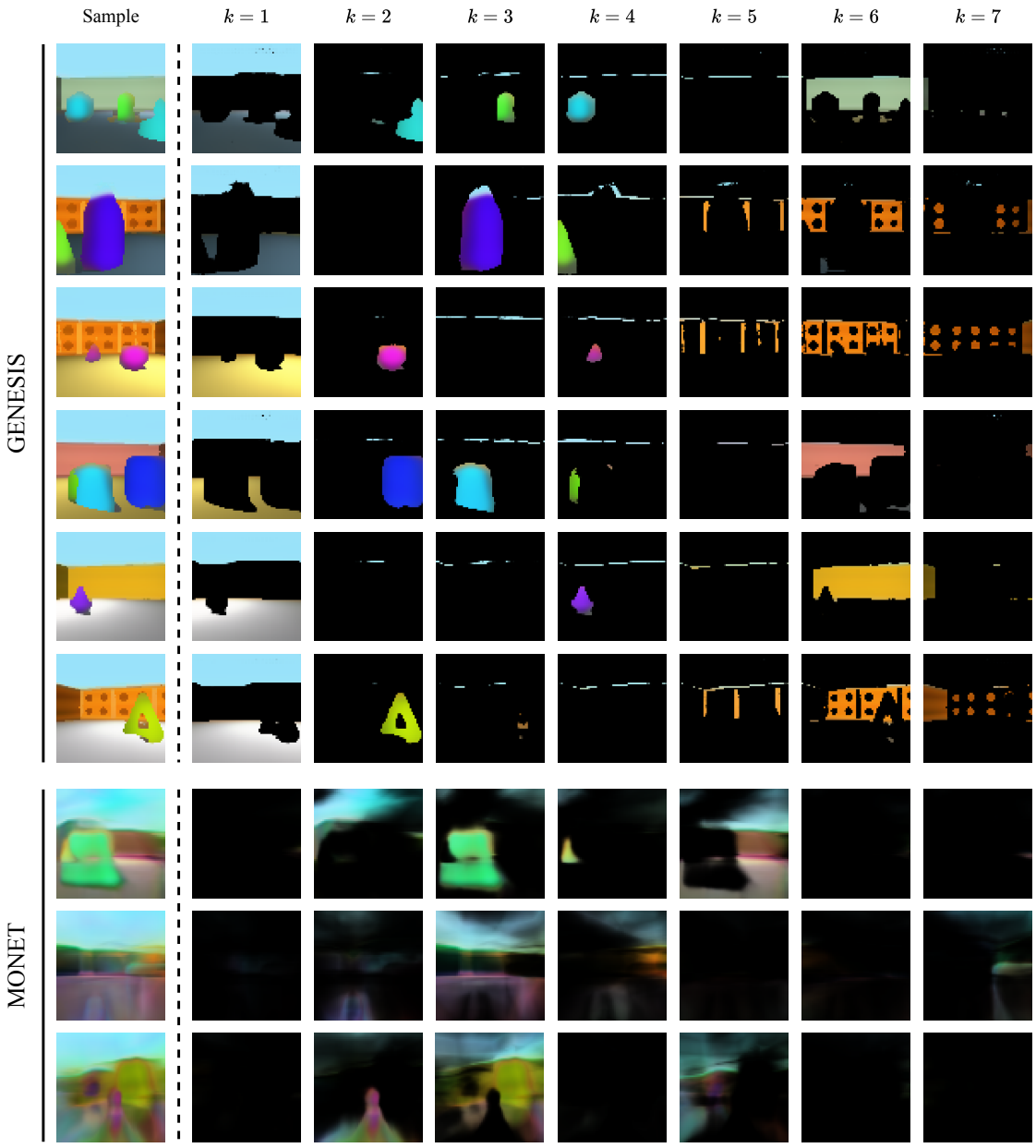


Figure 3.11: Object-centric scene generation with GENESIS and MONET after training on the GQN dataset. In contrast to MONET, GENESIS successfully generates entire coherent scenes in an object-centric fashion.

crisp and sensible compared to MONET, but foreground objects sometimes take odd shapes and colours. As for the inferred components, generated components are also not very interpretable. To provide context for these qualitative results, Figure 3.15 shows images generated by two standard VAEs, one with a spatial broadcast decoder and the other with a convolutional decoder as described in Section 3.3.3. The generated images are coherent and very typical for VAEs. The

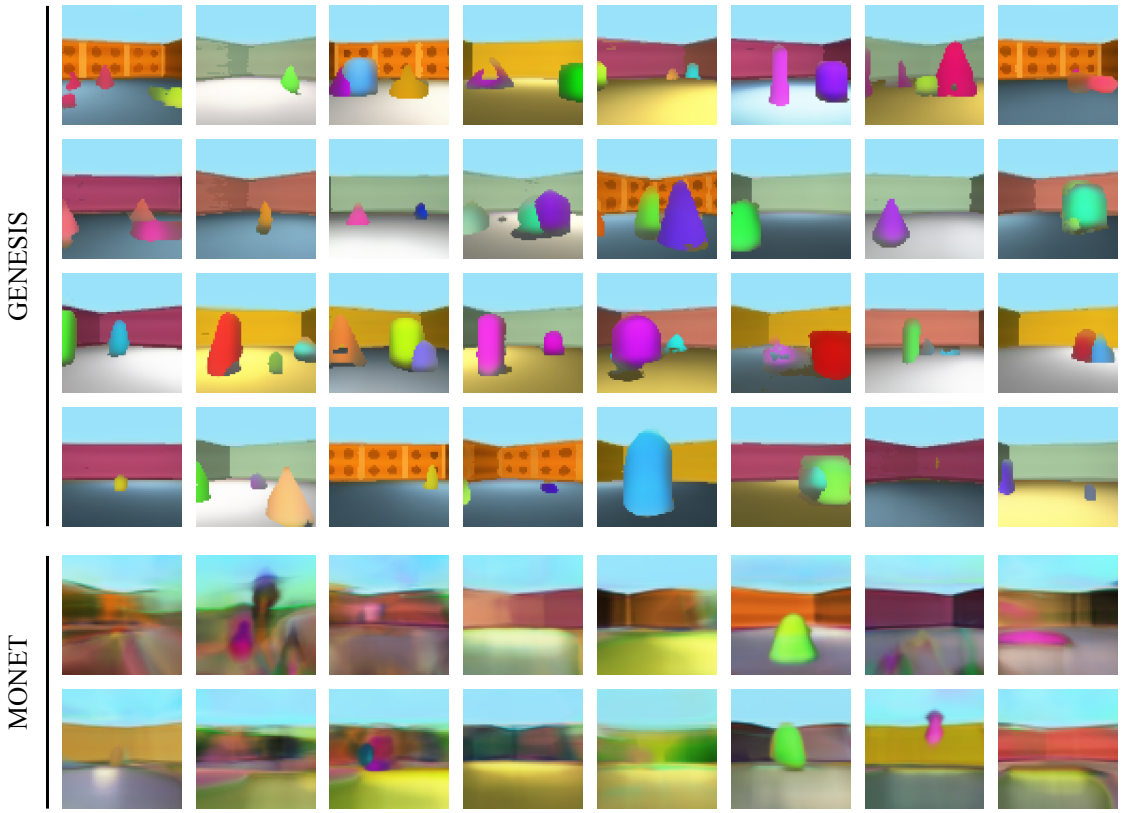


Figure 3.12: A variety of scenes generated by GENESIS and MONET after training on the GQN dataset. GENESIS generates plausible scenes while samples from MONET are largely incoherent.

general appearance of the samples resembles the training data, but edges and foreground objects are blurry. The DC-VAE captures some finer details compared to the SBD-VAE, for example the dots on some of the background walls, indicating that the architecture is more flexible. In contrast to these, interestingly, GENESIS generates images with much crisper object boundaries.

Finally, we evaluate the quality of generated scenes quantitatively by computing FID scores using 10,000 images generated by a model and 10,000 images from the test set as in Heusel et al. [108]. Results for GENESIS, GENESIS-S, MONET, and the two VAE baselines are summarised in Table 3.9 for the Multi-dSprites and the GQN datasets. GENESIS is not able to produce good samples on ShapeStacks following the optimisation procedure described in Section 3.3.4. The ShapeStacks sample quality can likely be improved by training for longer and better tuning the architecture. As can be seen in Table 3.9, GENESIS and GENESIS-S achieve

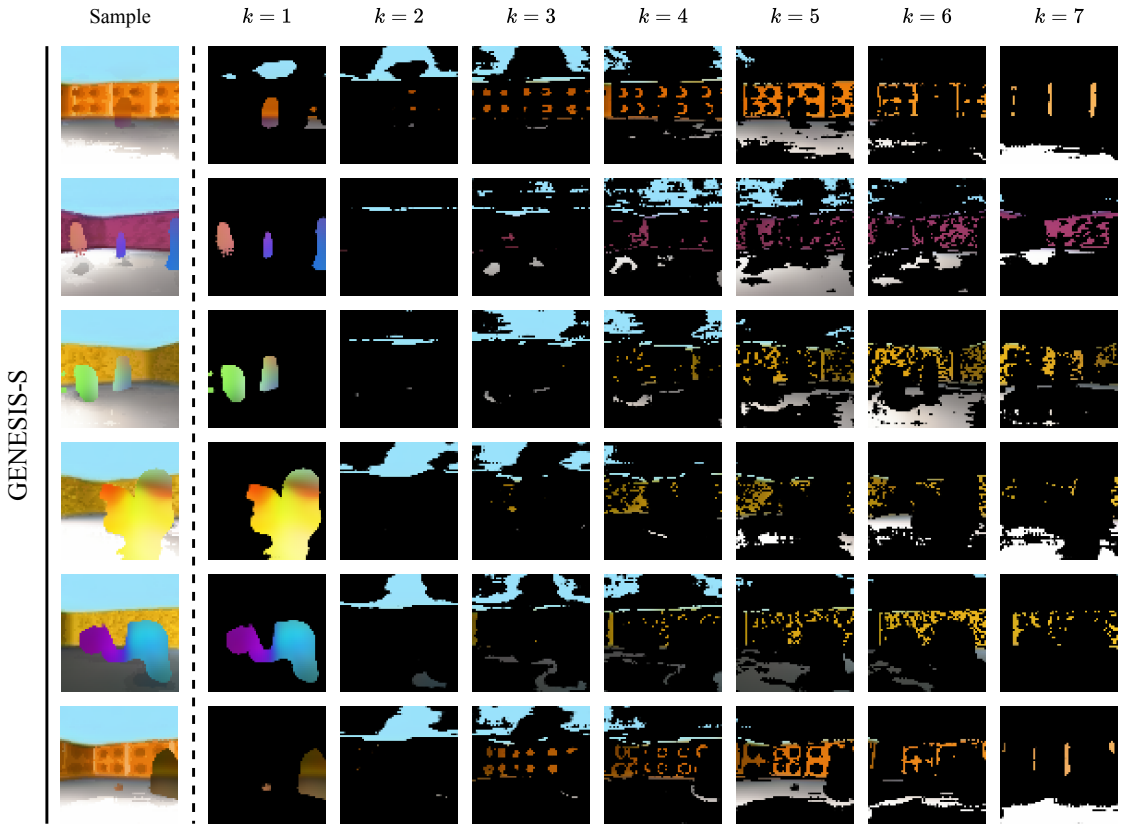


Figure 3.13: Scenes generated with GENESIS-S after training on the GQN dataset are plausible but individual components lack interpretability.

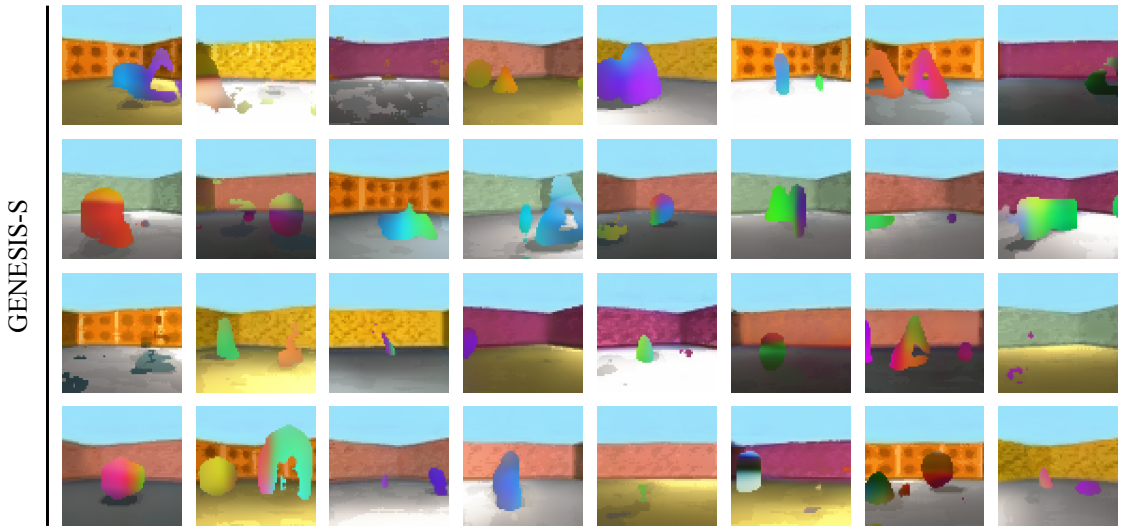


Figure 3.14: A variety of scenes generated by GENESIS-S after training on the GQN dataset. The backgrounds tend to show a high level of visual detail but foreground objects are often distorted.

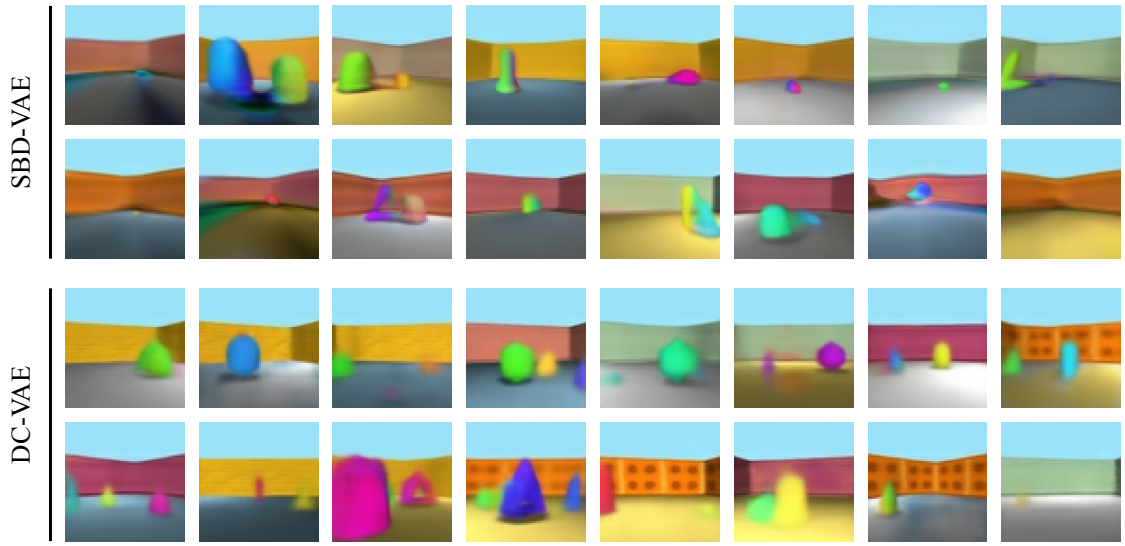


Figure 3.15: After training on the GQN datasets, scenes generated by the SBD-VAE lack visual details while samples from a DC-VAE are fairly good.

the best scores overall. Interestingly, GENESIS-S achieves a better score than GENESIS on the GQN dataset, even though individual components are not very interpretable. This might be caused by the fact that GENESIS-S dedicates more slots for reconstructing texture details of the background. Unsurprisingly, the FID scores for MONET are comparatively large as expected from the qualitative results. Nevertheless, MONET is able to achieve a score that is competitive with the two VAEs on Multi-dSprites, possibly because individual sprites are randomly superimposed on a uniform background without any interactions. Hence, as long as not more than one background component is generated by MONET, the resulting scene is likely to be somewhat similar to the training data.

Table 3.9: FID scores for GENESIS and the baselines. Smaller is better.

Dataset	GENESIS	GENESIS-S	MONET	SBD-VAE	DC-VAE
Multi-dSprites	**24.9**	28.2	92.7	89.8	100.5
GQN	80.5	**70.2**	176.4	145.5	82.5

3.3.7 Semi-Supervised Learning

After verifying the main competencies of GENESIS, this section investigates the content of the learned representations and their utility in downstream applications.

Table 3.10: Classification accuracy in % on the semi-supervised ShapeStacks tasks using images from the test set.

Task	GENESIS	GENESIS-S	MONET	SBD-VAE	DC-VAE	Random
Stability	**64.0**	63.2	59.6	60.1	59.0	50.0
Height	80.3	80.8	**88.4**	78.6	67.5	22.8
View	99.3	**99.7**	99.5	**99.7**	99.1	6.25

This is framed as a set of semi-supervised learning tasks using the scene annotations available for the ShapeStacks images. After training a set of generative models on all unlabelled images, a limited number of labelled training data is used to train a set of classifiers on the latent representations of the generative models to predict three scene properties. The first two tasks require the model to infer both scene content as well as to conduct basic physical and relational reasoning. The third tasks requires the model to infer information about the geometry of a scene:

1. Is a tower of blocks stable or unstable?
2. How many blocks are there in a tower?
3. From which of the 16 camera viewpoints is the scene shown?

We implement this by training feed-forward classifiers on the latent encodings of different models to predict the ground truth scene annotations. These classifiers are implemented as MLPs with a single hidden layer and 512 units, an ELU non-linearity, and a fully-connected layer followed by a softmax operator to obtain classification scores. For GENESIS, GENESIS-S, and MONET, all latent variables are concatenated into a single, fixed-length vector as input to the classifiers. The generative models are first trained as described in Section 3.3.4 on the whole ShapeStacks training set. Subsequently, their weights are fixed and classifiers are trained on the extracted latent encodings on 50,000 labelled examples. A cross-entropy loss is minimised with the ADAM optimiser (Kingma and Ba [117]), a learning rate of 10^{-4}, and a batch size of 128 for 100 epochs.

The results for GENESIS, GENESIS-S, MONET, the two VAE baselines, as well as a "random baseline" are summarised in Table 3.10. The random baseline results correspond to random predictions that are uniformly distributed across

classes. The values in Table 3.10 are obtained from the frequency of occurrence of individual classes in the training set, which is assumed to be representative of the test set. For the stability and height prediction class, each class has the same number of examples in the training set. However, the distribution of tower height in the training set is non-uniform. Both GENESIS variants outperform all four baselines at predicting tower stability. MONET approximately performs on par with the two VAE baselines on stability prediction and is the only model to outperform GENESIS on tower height prediction. It is unclear why MONET performs so well on this particular task, but still falls behind GENESIS on the stability prediction task. Notably, the SBD-VAE achieves a better accuracy than the DC-VAE on all three tasks. This might be explained by the fact that the spatial broadcast decoder of the SBD-VAE was specifically designed for facilitating the disentanglement of the underlying factors of variation in a dataset (Watters et al. [115]). All models perform very well on predicting the camera viewpoint, indicating that this task is fairly easy.

It should be noted that none of these models are able to compete with the accuracies achieved for stability prediction in Groth et al. [88]. This is not surprising, however, given that Groth et al. [88] train a large Inception-v4 classifier (Szegedy et al. [118]) on all labelled images while also using a higher image resolution and data augmentation. Nevertheless, the purpose of these experiments is not to achieve better performance, but to investigate the utility of object-centric representations for downstream tasks. To this end, it can be observed that the object-centric models consistently outperform the other baselines in the first two tasks which require reasoning about the foreground objects in the scene.

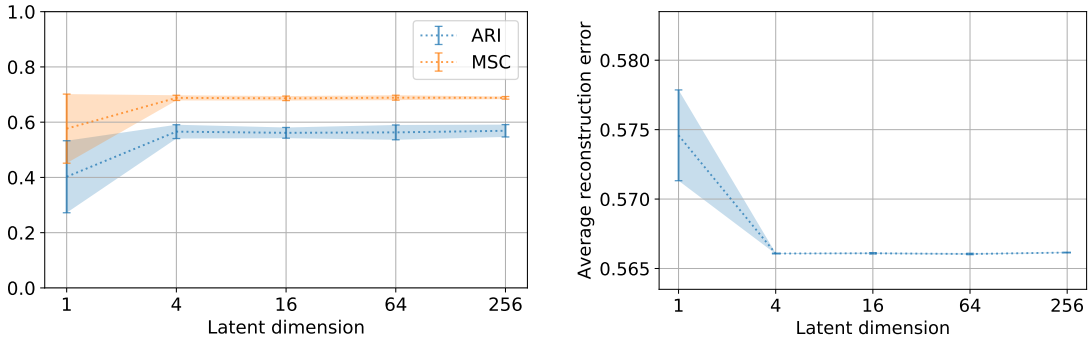
3.3.8 Reconstruction Bottlenecks

When developing GENESIS, we observed that decomposition and reconstruction quality critically depend on "reconstruction bottlenecks". On the one hand, if the component VAE is too flexible, then the model reconstructs an image as a single component without learning to segment it into object-like components. If on the

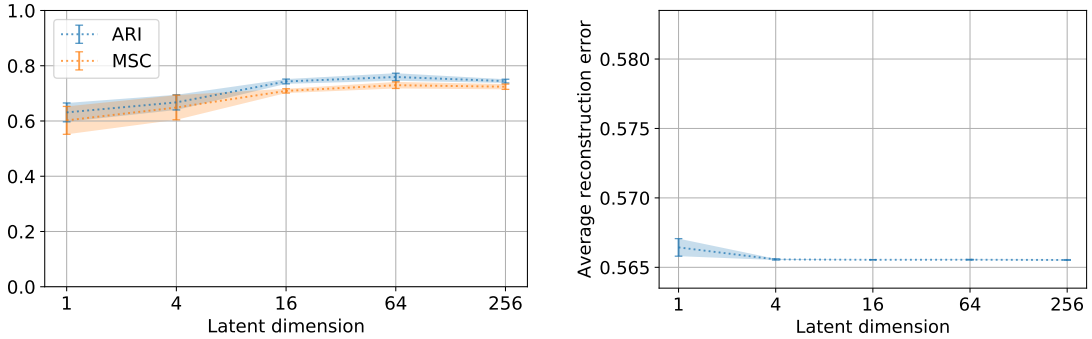
other hand the component VAE is not flexible enough, then reconstruction quality as well as segmentation quality suffer as the model struggles to accurately capture object boundaries. In this section, we investigate this phenomenon systematically by changing the reconstruction bottleneck in two ways: firstly, by varying the dimensionality of $\mathbf{z}_k^c$ and secondly, by modifying the architecture. The standard GENESIS architecture as described in Section 3.3.3 is "asymmetric", with the mask network and the component network having different specifications. As an alternative, a "symmetric" architecture is also considered to examine the importance of this design choice. This is achieved by using convolutional encoders and decoders in the component network that are similar to the corresponding counterparts in the mask network (Table 3.1 and Table 3.2). The only difference is that the second Fully-Connected (FC) layer in the encoder was removed so that the first FC-GLU layer directly maps to the sufficient statistics of the latent posterior. Models are trained with component latent dimensions D_c of {1, 4, 16, 64, 256} for the asymmetric architecture and {1, 2, 4, 8, 16} for the symmetric architecture. For the latter, preliminary experiments showed that the model only learns to decompose images for smaller latent dimensions so the range of values was adjusted to increase the resolution in the more interesting hyperparameter range. Performance is quantified using the ARI and the MSC, computed on 320 images from the validation set. The means and standard deviations are reported for three models trained with different random seeds. Additionally, the expressiveness of the encoder and decoder is also examined separately by setting $K = 1$, effectively turning the model into a standard VAE for both the asymmetric and the symmetric architecture.

Asymmetric Architecture ARI and MSC scores as well as pixel reconstruction errors for component latent dimensions D_c of {1, 4, 16, 64, 256} are shown in Figure 3.16 for the Multi-dSprites, ShapeStacks, and ObjectsRoom datasets. For this architecture choice, the models decompose the images across the full range of latent dimensions, i.e. both the ARI and the MSC are significantly larger than zero in all cases. It can be observed that both segmentation and reconstruction

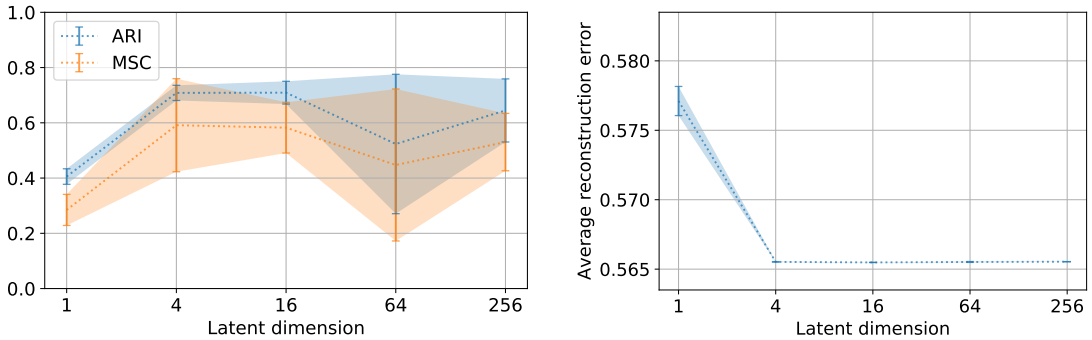
qualities slightly drop when the latent dimension is very small, indicating that the models struggle to capture finer details in the images. Figure 3.17 shows qualitative results of the inferred segmentation masks for the component latent dimensions {1, 16, 256}. The segmentations are very similar in the later two cases. When only a single latent dimension is used, however, the bottom two blocks in the tower



(a) Multi-dSprites



(b) ShapeStacks



(c) ObjectsRoom

Figure 3.16: Segmentation and reconstruction metrics for the standard asymmetric GENESIS architecture. The models learn to decompose the images even for large latent dimensions. Segmentation performance drops when the latent dimension is very small.

Table 3.11: Training iterations needed for the asymmetric GENESIS architecture to reach the GECO reconstruction goal when $K = 1$. None of the models is able to reach the GECO goal, so GENESIS with $K > 1$ is unlikely to collapse to a single component.

	Multi-dSprites		ShapeStacks		ObjectsRoom	
D_c	Goal iter	500k err	Goal iter	500k err	Goal iter	500k err
1	-	0.5885	-	0.5852	-	0.6064
2	-	0.5820	-	0.5819	-	0.5921
4	-	0.5763	-	0.5785	-	0.5820
8	-	0.5730	-	0.5754	-	0.5767
16	-	0.5697	-	0.5730	-	0.5715
32	-	0.5676	-	0.5709	-	0.5703
64	-	0.5675	-	0.5712	-	0.5698
128	-	0.5676	-	0.5715	-	0.5700
256	-	0.5674	-	0.5710	-	0.5700

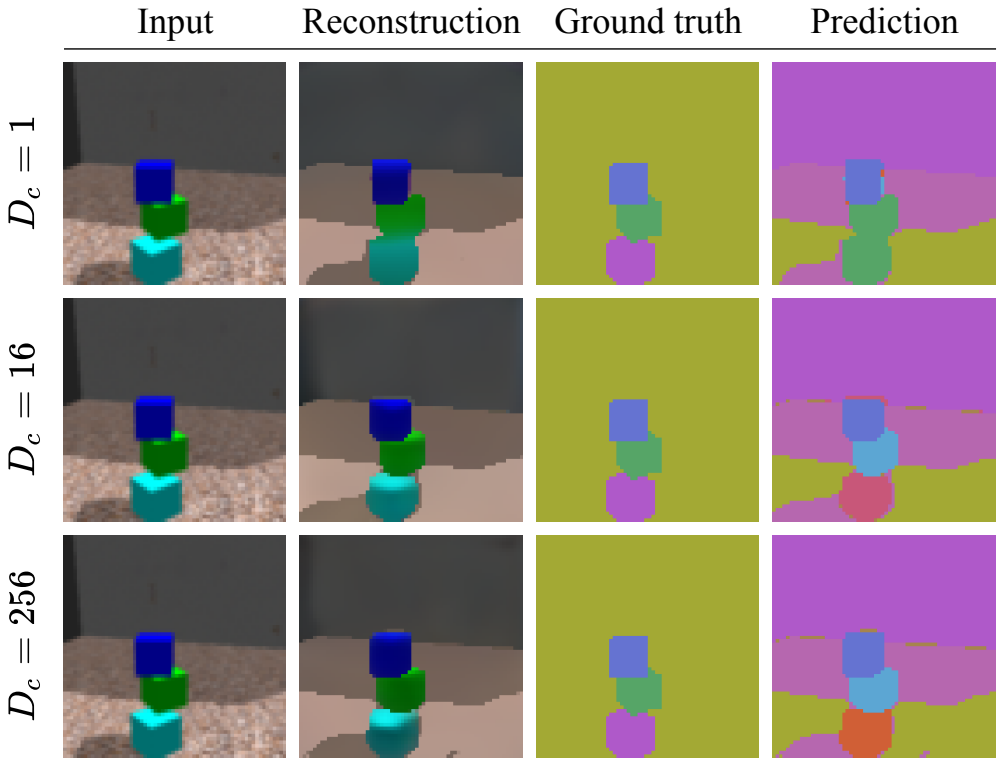


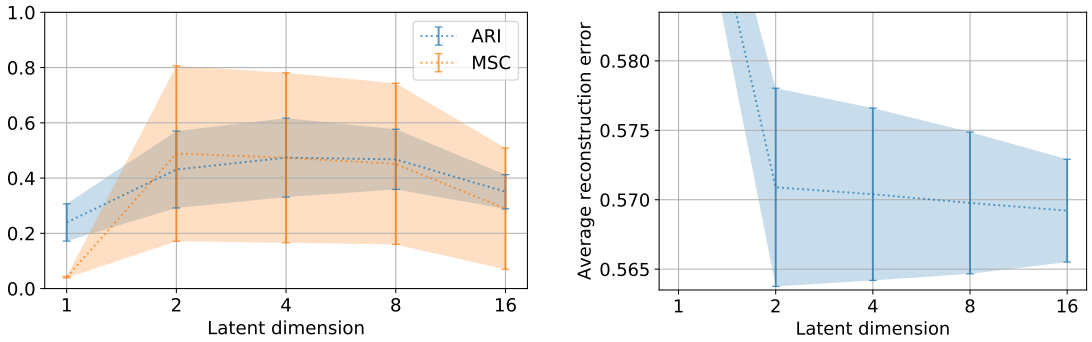
Figure 3.17: Qualitative segmentation results with the architecture on ShapeStacks for latent dimensions of {1, 16, 256} corroborate the observations from Figure 3.16.

are not disambiguated in the segmentation and their colours flow into each other in the reconstruction. To understand why the models do not collapse to a single component even for a comparatively large latent dimension, another set of models is trained with $K = 1$, practically turning the model into a standard VAE. For the different latent dimensions, either the number of iterations required to reach the GECO reconstruction goal or the final moving average of the reconstruction error is recorded in Table 3.11. Notably, it can be seen that the model with $K = 1$ never reaches the GECO goal on its own even for comparatively large latent dimensions, thus prohibiting the model from collapsing to a single component when $K > 1$.

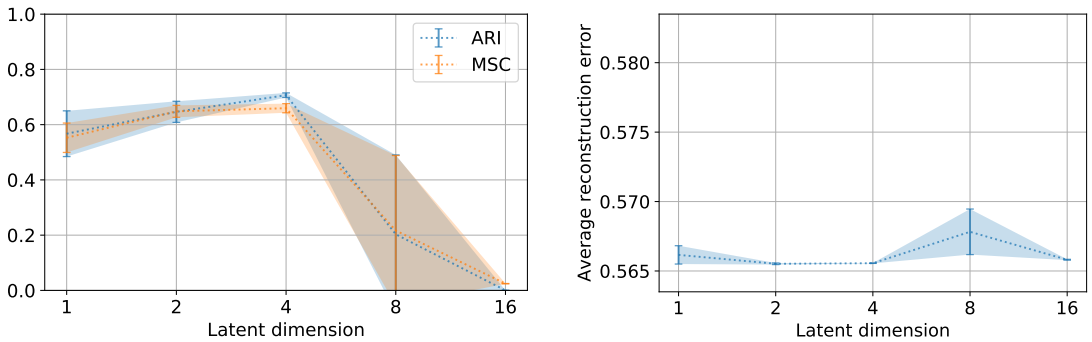
Symmetric Architecture ARI and MSC scores as well as pixel reconstruction errors for component latent dimensions D_c of $\{1, 2, 4, 8, 16\}$ are shown in Figure 3.18 for the Multi-dSprites, ShapeStacks, and ObjectsRoom datasets. For this architecture choice, the models decompose the images only for a small range of latent dimensions. To understand why the models collapse to a single component for larger latent dimensions, another set of models was again trained with $K = 1$. These results are summarised in Table 3.12. The resulting model always reaches the GECO goal on its own for latent dimensions larger than 16. When $K > 1$, the models with the symmetric architecture can therefore still produce good reconstructions without decomposing the input image as can be seen in the qualitative results for the component latent dimensions $\{1, 4, 16\}$ in Figure 3.19. This is interesting from a practitioners point of view: the results indicate that the reconstruction quality of a VAE can be used to gauge its utility when used in an OCGM such as GENESIS.

3.4 Conclusions

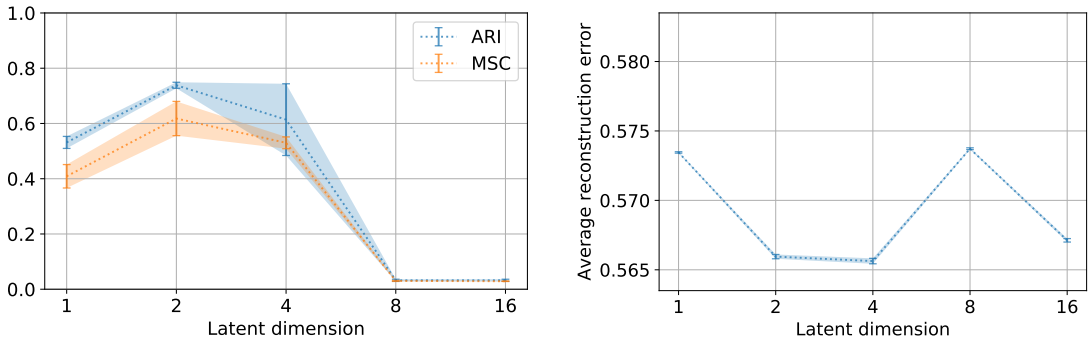
In this chapter, a novel unsupervised object-centric generative model called GENESIS was introduced. Similar to prior works in Burgess et al. [25] and Greff et al. [26], GENESIS learns to decompose visual scenes into object-like entities. In contrast to these works, GENESIS differentiates itself by featuring an autoregressive prior that captures relationships between scene components, enabling the model to generate



(a) Multi-dSprites



(b) ShapeStacks



(c) ObjectsRoom

Figure 3.18: Segmentation and reconstruction metrics for the more flexible symmetric GENESIS architecture. The models do not decompose the images for large latent dimensions. Segmentation performance again drops as in Figure 3.16 when the latent dimension is very small.

Table 3.12: Training iterations needed for the symmetric GENESIS architecture to reach the GECO reconstruction goal when $K = 1$. The models start reaching the goal when $D_c > 8$, so GENESIS with $K > 1$ is likely to collapse to a single component.

	Multi-dSprites		ShapeStacks		ObjectsRoom	
D_c	Goal iter	500k err	Goal iter	500k err	Goal iter	500k err
1	-	0.6008	-	0.5820	-	0.6082
2	-	0.5808	-	0.5766	-	0.5904
4	-	0.5753	-	0.5726	-	0.5825
8	-	0.5697	-	0.5683	-	0.5733
16	-	0.5660	246k	-	-	0.5668
32	79k	-	25k	-	61k	-
64	21k	-	9k	-	16k	-
128	10k	-	6k	-	7k	-
256	8k	-	5k	-	6k	-

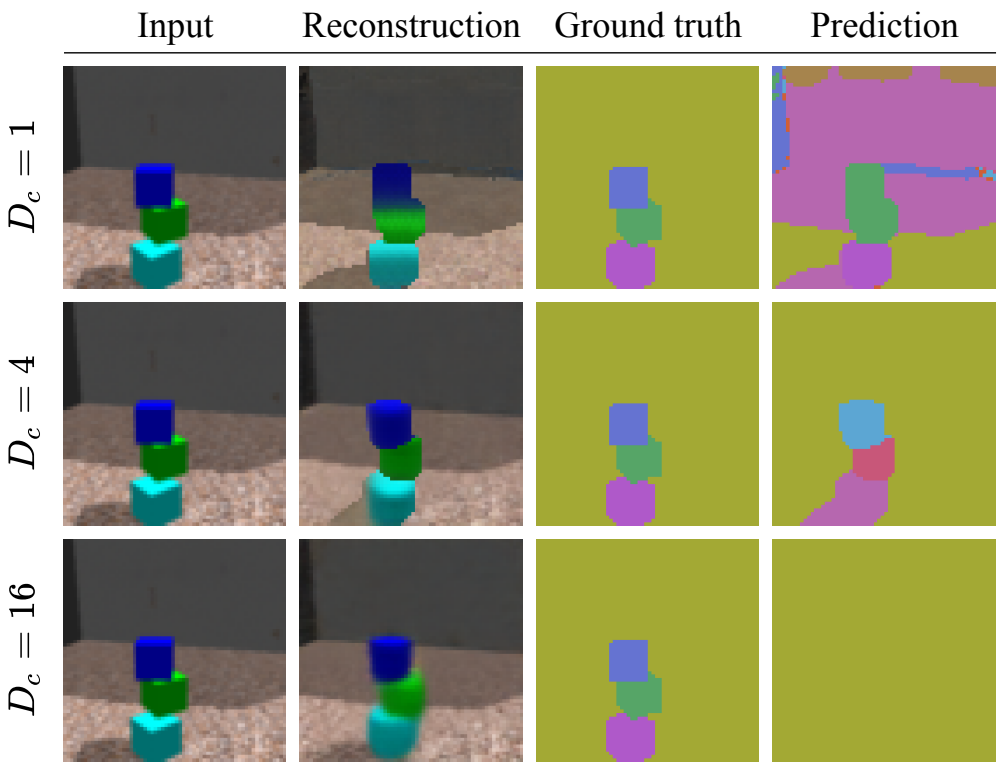


Figure 3.19: Qualitative segmentation results with the more flexible symmetric architecture on ShapeStacks for three different latent dimensions corroborate the observations from Figure 3.18.

entire coherent scenes. The proposed model was evaluated both qualitatively and quantitatively on several well-established datasets. We first showed that GENESIS learns comparable scene segmentations to our custom re-implementation of MONET from Burgess et al. [25]. Crucially and unlike MONET, we show that GENESIS is able to generate scenes with 3D structure in a component-by-component fashion. We also conducted a set of semi-supervised experiments to probe the information contained in the learned representations which indicated that the representations of both GENESIS and MONET are more suitable than two VAE baselines for predicting scene properties relating to the foreground objects in an image. Finally, we conducted an empirical investigation into the role of reconstruction bottlenecks. It was observed that if a bottleneck is too wide, then the models collapse to a single component and do not learn any useful decompositions. If it is too narrow, in contrast, then the model still learns to decompose images but segmentation and reconstruction quality suffer. With regards to the objectives of this thesis, these results show that GENESIS is able to discover objects in simulated images without supervision and also to generate entire coherent scenes, which can be utilised in future work for efficient skill acquisition in object-centric world models. Moreover, inference and generation RNNs run in low-dimensional latent space, so that computationally more expensive convolutional operations can be performed in parallel.

3.4.1 Limitations

GENESIS is a prove of concept to show that performing unsupervised object discovery and object-centric scene generation in a unified and principled fashion is feasible. While we use simulated datasets to demonstrated these capabilities, the visual complexity of these datasets is too limited in order for GENESIS to be suitable for more realistic applications. Specifically, at 64-by-64 the resolution of the images is fairly low, the maximum number of scene components per image is well constrained, images have limited and well-controlled visual variety, and objects are generally fairly distinguishable based on their colour alone. Even though model performance on these datasets is sufficient for demonstrating the desired

capabilities, segmentation and generation quality of GENESIS still leave much room for improvement. Regarding the structure of the model itself, the recurrent inference mechanism where mask variables are inferred with an LSTM from a vector encoding of an input image is unlikely to scale well to larger and more complex images. Overall, getting the model to exhibit the desired behaviour requires significant manual tuning. Although the same hyperparameters are used across the datasets considered in this chapter, this is possibly facilitated by the fact that these datasets are all visually similar to some degree. Training GENESIS on other datasets that are visually quite different—albeit of comparable complexity—would likely require additional tuning in order to achieve interpretable object decompositions.

3.4.2 Future Work

Several improvements of GENESIS as presented in in this chapter would be of interest. For a similar visual complexity of the data, it would be interesting to see how far the model can be pushed in terms of scaling it to higher-resolution images that potentially also contain more objects. This might reveal limitations of the model as well as opportunities for improvement. Such an investigation could be further extended by training the models with smaller GECO goals to obtain more accurate reconstructions, segmentations, and higher fidelity samples. We conjecture that it might be possible to derive further insights from such experiments for scaling OCGMs to real-world datasets, which often contain a larger number of objects as well as finer visual details that are important for practical applications. These improvements would likely require modifying the architecture to make it both more expressive and more efficient in order to keep training times and hardware requirements manageable. However, one open question is whether the inductive biases present in the model would still be sufficient to learn object-like decompositions without supervision in this more difficult setting.

4

Scaling Object-Centric Generative Models to Real-World Robotics Datasets with Differentiable Instance Colouring

In Chapter 3 we discussed unsupervised object discovery and object-centric world modelling with Generative Scene Inference and Sampling (GENESIS). This was evaluated on several simulated datasets as done in prior art (Burgess et al. [25] and Greff et al. [26]). The visual complexity of these datasets, however, is much lower than encountered in real-world robotics applications. This chapter therefore explores means for scaling object-centric generative models to more complex real-world datasets. A few recent works have already investigated this challenge. For example, Greff et al. [26] apply Iterative Object Decomposition Inference Network (IODINE) to images from the ImageNet dataset [10] with limited success: the model tends to segment images into regions of similar colour rather than object-like components. Some promising early results are achieved by tailoring methods to specific datasets with strong inductive biases on scene geometry (Nguyen-Phuoc et al. [37] and Ehrhardt et al. [38]) or object size (Jiang et al. [33]). While the works in Yang et al. [34] and Bear et al. [35] do not rely on such strong constraints, both works only produce very coarse reconstructions and consequently do not capture any fine-grained attributes of scene components. As we saw in Chapter 3, reconstruction

bottlenecks can play a critical role in scene decomposition and it is hence unclear whether these methods are in principle able to produce better reconstructions.

In this chapter, we aim to develop a general method by improving GENESIS and without constraining the model as in Jiang et al. [33], Nguyen-Phuoc et al. [37], and Ehrhardt et al. [38]. To this end, we decide to focus on the development of a more suitable segmentation network given its central role in the quality of scene decompositions. Scene decomposition can be viewed as panoptic segmentation (Kirillov et al. [119]), object detection (Ren et al. [15]), or a combination of the two. Object-Centric Generative Models (OCGMs) that adopt a spatial mixture model such as GENESIS or the works in Burgess et al. [25] and Greff et al. [26] effectively perform panoptic segmentation and models that utilise Spatial Transformer Networks (STNs) (Jaderberg et al. [89]) perform object detection (e.g. Eslami et al. [23] and Kosiorek et al. [24]). Panoptic segmentation itself is a special case of instance segmentation with the additional constraints that each pixel has to be assigned to an instance and instances are strictly non-overlapping. Instance segmentation methods broadly fall intro three categories: (1) bounding-box based (e.g. He et al. [105] and Liu et al. [120]), Recurrent Neural Network (RNN) based (e.g. Romera-Paredes and Torr [121] and Ren and Zemel [122]), and instance colouring methods (e.g. De Brabandere et al. [123] and Bai and Urtasun [124]). As in Chapter 3, we want to avoid the use of bounding boxes as they typically require explicit handling of different object scales and rely on a sampling grid for obtaining gradients (Jaderberg et al. [89]) which makes optimisation more difficult. We begin by observing that GENESIS and the works in Burgess et al. [25] and Greff et al. [26] rely on different recurrent mechanisms to predict segmentation masks. As mentioned in Novotny et al. [41], however, recurrent methods are computationally expensive and do not scale well to large images with many objects, rendering them unattractive for real-world robotics applications where computational efficiency is of the essence. Adopting an instance colouring framework is therefore an attractive alternative. This involves the prediction of pixel-wise embeddings which are subsequently clustered into a set of instances, therefore

avoiding both the use of bounding boxes and RNNs. Moreover, we conjecture that the recurrent inference mechanism in GENESIS is not only problematic from a computational point of view, but that it can also inhibit the learning of object-centric representations by imposing an ordering on scene components. In particular, we posit that the asymmetric, sequential treatment of objects affects the flow of information during the forward pass and the flow of gradients during the backward pass, rendering the model more likely to collapse to a single component as observed in Chapter 3, Section 3.3.8. For example, if the first component receives gradients of larger magnitude than the components, then its associated weights change more quickly and the first component might end up reconstructing the entire image at once as a trivial and non-interesting solution. In this chapter, we therefore instead propose to infer objects in a "symmetric" fashion without imposing an ordering as in Greff et al. [26] and Locatello et al. [36], for example.

While adopting an instance colouring framework for segmentation seems beneficial both from a computational and a conceptual point of view, existing methods either use a non-differentiable clustering algorithm as a post-processing step to obtain instance masks (e.g. De Brabandere et al. [123] and Bai and Urtasun [124]) or rely on supervised signals to identify "seeds" which are used for initialising clusters (e.g. Novotny et al. [41] and Fathi et al. [125]). Neither of these approaches can be employed for unsupervised, end-to-end learning of instance segmentation masks. As one of the innovations in this chapter, we therefore propose a kernel-based, fully-differentiable Instance Colouring Stick-Breaking Process (IC-SBP) which samples cluster seeds stochastically to perform a soft clustering of pixel embeddings without imposing a fixed ordering on scene components. This is utilised in combination with "semi-convolutional embeddings" as developed in Novotny et al. [41], who show that a fully translation-invariant Convolutional Neural Network (CNN) is incapable of distinguishing multiple objects of identical appearance. In addition, we argue that semi-convolutional embeddings contain an implicit weak bias that induces segmentation masks to be spatially localised as further elaborated on in Section 4.2.2. Given these semi-convolutional embeddings, we derive principled initialisations for

different distance kernels in the IC-SBP based on prior knowledge about the maximum number of objects in an image. These mechanisms should help to avoid uninteresting trivial solutions where the model segments regions of similar colour as such segmentations are often not spatially localised. However, using symmetric object representations without an inherent ordering is potentially problematic when trying to use a sequential, autoregressive prior for object-centric image generation as introduced in Chapter 3. The autoregressive prior predicts the next object latents one at a time, which is more difficult to perform when the posterior latents are in a random order. We therefore investigate a greedy sorting of posterior object latents to learn a better autoregressive prior for scene generation. We refer to the resulting model that incorporates these innovations as GENESIS++.

This chapter begins by reviewing literature related to unsupervised learning of real-world object-centric representations and the symmetric prediction of unordered sets in Section 4.1. GENESIS++ and its individual components are subsequently described in Section 4.2; including the IC-SBP, principled kernel initialisations for semi-convolutional embeddings, and the autoregressive prior with greedy sorting. In Section 4.3, GENESIS++ is first evaluated on the simulated ShapeStacks (Groth et al. [88]) and ObjectsRoom (Burgess et al. [25] and Kabra et al. [103]) datasets where ground truth segmentations masks are available. We show that GENESIS++ outperforms GENESIS and our re-implementation of Multi-Object Network (MONET) from Chapter 3 both in terms of scene decomposition and object-centric scene generation. Furthermore, we present qualitative and quantitative results on three challenging real-world datasets where we show that GENESIS++ achieves promising results. Finally, the conclusions of this chapter are summarised in Section 4.4.

4.1 Related Work

Previous models for unsupervised, object-centric representation learning have been applied to real-world data with some success under certain conditions. Kosiorek et al. [24] track pedestrians in CCTV footage, but videos are preprocessed with a

background subtraction algorithm which simplifies the problem to tracking white silhouettes of people on a black background. Similarly, Jiang et al. [33] also track pedestrians in overhead, greyscale CCTV footage. While this setup is more general than in Kosiorek et al. [24], the task is still simplified by the fact that people largely correspond to dark, connected regions on a light background and they are all of approximately the same size. Unlike Kosiorek et al. [24] and Jiang et al. [33], GENESIS++ does not use temporal information as an additional decomposition signal, but an extension to incorporate temporal information is in principle feasible. Moreover, our formulation allows the identification of scene components of arbitrary size. Yang et al. [34] achieve promising results on the Flying Animals dataset which has complex visual appearances as found in natural images. Nevertheless, animals are artificially superimposed on background pictures without background occlusion. This also leads to lighting discontinuities which make objects more distinguishable. Moreover, reconstructions are very blurry and fine visual details are not captured. Similarly, Bear et al. [35] show promising results on both simulated and and real-world indoor scenes. Yet, many visual details are again lost in the reconstructions. In particular, Bear et al. [35] use "quadratic texture rendering" which effectively fits a quadratic function to the RGB values in an image region. In contrast to these works, we strive to also capture more fine-grained image details and GENESIS++ employs a learnable decoder which can in principle generate arbitrarily complex textures. While these previous works are trained with a reconstruction objective, a few works learn object-centric generative models in an adversarial fashion (van Steenkiste et al. [30], Nguyen-Phuoc et al. [37], and Ehrhardt et al. [38]). Both Nguyen-Phuoc et al. [37] and Ehrhardt et al. [38] present results on real-world datasets, showing a single car in front of a background and several cars crossing a traffic junction, respectively. As in Chapter 3, we again adopt a reconstruction objective which allows the joint learning of inference and generative models. We also refrain from using explicit geometric constraints as in Nguyen-Phuoc et al. [37] and Ehrhardt et al. [38] to make the model as general as possible. As in Chapter 3, GENESIS++ is also equipped with an autoregressive prior that enables object-centric scene generation.

As motivated earlier, we argue that object representations should be inferred in a symmetric fashion without imposing an unnatural ordering. This is effectively a form of "set prediction" which has also been explored in the context of object detection and point cloud generation (Rezatofighi et al. [126], Zhang et al. [127], Kosiorek et al. [128], and Carion et al. [129]). Of particular relevance are the works in Greff et al. [26] and Locatello et al. [36] who also learn symmetric object representations in an unsupervised fashion. While Greff et al. [26] employ a costly iterative inference procedure, Locatello et al. [36] use a computationally cheaper iterative mechanism based on self-attention (Vaswani et al. [130]) to extract symmetric object representations from a feature map. Neither Greff et al. [26] nor Locatello et al. [36], however, address the task of learning a prior for scene generation from a symmetric set of object representations. Moreover, both models require the number of scene components to be specified *a priori*. The IC-SBP proposed in this chapter generates object masks using an iterative—but non-parametric—clustering procedure that leads to a random ordering of scene components. In principle, the IC-SBP can generate masks until the whole image is "explained" without having to specify a maximum number of scene components beforehand. Nevertheless, in this work we still run the IC-SBP for a fixed number of steps for simplicity and the empirical validation of using a flexible number of steps is left for future work.

Performing scene decomposition via a spatial mixture model as done in this chapter, Chapter 3 and many other works (e.g. Burgess et al. [25] and Greff et al. [26]) effectively corresponds to panoptic segmentation Kirillov et al. [119], which is specific form of instance segmentation. Works on instance segmentation typically consider a supervised setting (He et al. [105], Kirillov et al. [119], and Liu et al. [120]), which is a possible additional application of the IC-SBP developed here. As discussed previously, instance segmentation methods broadly fall in one or several of three categories: (1) instances from bounding boxes (e.g He et al. [105] and Liu et al. [120]), (2) RNN based methods (e.g. (Romera-Paredes and Torr [121], Ren and Zemel [122], and Salvador et al. [131])), and (3) instance colouring methods (e.g. De Brabandere et al. [123], Bai and Urtasun [124], Fathi et al. [125], Kirillov

et al. [132], Kulikov et al. [133], and Kong and Fowlkes [134]). This work adopts an instance colouring approach. One form of instance colouring proposed in Kulikov et al. [133] is to use a K-channel softmax to predict pixel-wise output probabilities with each softmax channel corresponding to an instance. This, however, upper bounds the number of instances and also imposes an ordering as is to be avoided for symmetric object inference. To the best of our knowledge, we found no existing instance colouring methods to be suitable for unsupervised end-to-end learning of instance segmentations as typical methods either utilise a hard clustering or thresholding operation to obtain individual instance masks (De Brabandere et al. [123], Bai and Urtasun [124], and Kong and Fowlkes [134]) or rely on a supervised training signal to obtain a seed to perform a soft clustering operation (Novotny et al. [41] and Fathi et al. [125]). In contrast to these non-differentiable clustering mechanisms, the proposed IC-SBP is a form of Dirichlet process. In particular, a kernel-based distance is used to assign cluster probabilities which is akin to "kernel stick-breaking processes" (Dunson and Park [135]). Dirichlet processes have previously also been used before in the context of image segmentation (An et al. [136], Orbanz and Buhmann [137], and Ren et al. [138]), but none of these methods leverage the recent advances in visual feature learning with CNNs (LeCun et al. [2]).

4.2 GENESIS++

In this section, we describe the details of GENESIS++. The overall goal is the development of a model that is better in terms of scene segmentation and object-centric generation than GENESIS as developed in Chapter 3 on both simulated and real-word datasets. To this end, we modify the structure of the model but we still formulate GENESIS++ as a probabilistic latent variable model which is trained as a Variational Autoencoder (VAE) (Kingma and Welling [27] and Rezende et al. [28]) given the successful adoption of this framework in Chapter 3. Consequently, we adopt the same notation in this chapter as introduced in Chapter 3.

The central new component of GENESIS++ is the IC-SBP which is utilised to cluster pixel embeddings $\zeta \in \mathbb{R}^{H \times W \times D_\zeta}$ into a set of normalised attention

masks $\mathbf{m}_{1:K}$ with $\mathbf{m}_k \in [0,1]^{H \times W}$ and $\sum_k \mathbf{m}_{i,j,k} = 1$ for all pixel coordinate tuples (i, j) in an image. As in Section 3.2, H and W are the spatial dimensions of the input image $\mathbf{x}$. D_ζ is the dimensionality of the pixel embeddings. Crucially, the attention masks $\mathbf{m}_{1:K}$ differentiate scene components without imposing an ordering on objects. How this is achieved by the IC-SBP is described in detail in Section 4.2.1. Overall, the model first extracts a shared image encoding using a deterministic, fully-convolutional backbone. This encoding is used to predict a map of semi-convolutional pixel embeddings (Novotny et al. [41]) which serve as inputs to the IC-SBP. Semi-convolutional embeddings are further discussed in Section 4.2.2. Now, there are several options in terms of how these attention masks can be utilised. The first option is to follow the approach in Burgess et al. [25] and use these deterministic masks for scene reconstruction. As established in Chapter 3, though, this leads to a training objective that does not constitute a variational lower bound and therefore lacks theoretical grounding. Instead, we take inspiration from Locatello et al. [36] and use the attention masks to pool a feature vector for each scene component from a shared feature map that is computed from the deterministic image encoding. A set of object latents $\mathbf{z}_{1:K}$ is then inferred from these pooled feature vectors which are reparameterised as in Kingma and Welling [27] and Rezende et al. [28]. The individual latents $\mathbf{z}_k$ are then decoded in parallel to parameterise the statistics of a Spatial Gaussian Mixture Model (SGMM) similar as in Chapter 3. The resulting VAE architecture is illustrated diagrammatically in Figure 4.1. Further implementation details of the CNN are described in Section 4.3.2. This section proceeds by first describing the IC-SBP in detail. Afterwards, we describe a principled initialisation of the distance kernels in the IC-SBP based on semi-convolutional embeddings and given the expected maximum number of scene components in a dataset to implement a weak inductive bias on object size. Finally, we discuss considerations for training an autoregressive prior for scene generation given an unordered set of object latents.

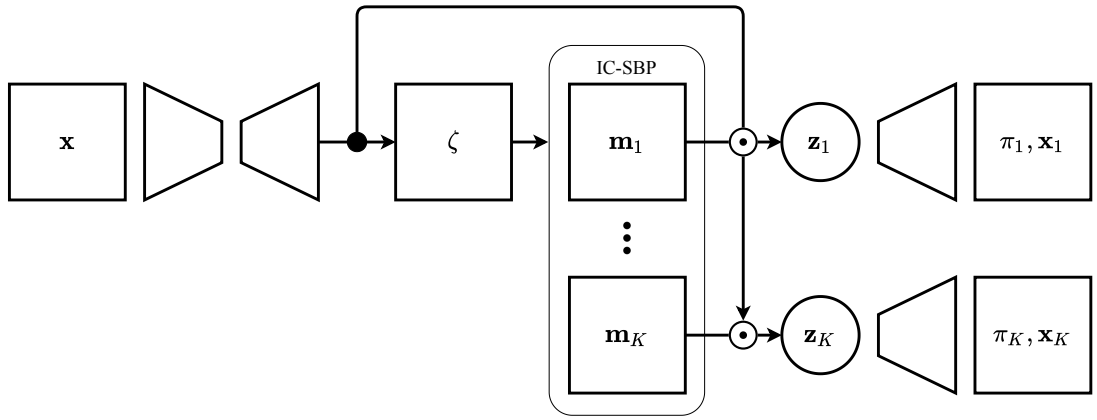


Figure 4.1: GENESIS++ autoencoder architecture. Illustration does not include the autoregressive prior. The image $\mathbf{x}$ is passed into a deterministic backbone. The resulting encoding is used to compute the semi-convolutional pixel embeddings ζ which are clustered into the attention masks $\mathbf{m}_{1:K}$ by the IC-SBP. Features are pooled according to these attention masks to infer the object latents $\mathbf{z}_{1:K}$. These are decoded into the object masks $\pi_{1:K}$ and reconstruction components $\mathbf{x}_{1:K}$.

4.2.1 Instance Colouring Stick-Breaking Process

Given a set of pixel-wise instance embeddings, we require a fully-differentiable mechanism for clustering these embeddings into instances masks, ideally without requiring prior knowledge of the maximum number instances. To this end, we propose a stochastic, fully-differentiable algorithm that performs a soft clustering of instance embeddings, which we are refer to as the Instance Colouring Stick-Breaking Process (IC-SBP). Consider a map of instance embeddings $\zeta \in \mathbb{R}^{H \times W \times D_\zeta}$, where H and W are the spatial dimensions of the embedding map and D_ζ is the dimensionality of the embeddings. Assume H and W are equal to the spatial dimensions of an input image. While H and W could also be smaller than the spatial dimensions of the image to reduce computational cost, setting them equal to the image dimensions in principle allows each pixel to be assigned to a different instances, making the algorithm as general as possible.

Similar to the Stick-Breaking Process (SBP) in Chapter 3 and in Burgess et al. [25], the IC-SBP proceeds by initialising a "scope" $\mathbf{s}$ to a matrix of ones $\mathbb{1}^{H \times W}$. The scope $\mathbf{s} \in [0, 1]^{H \times W}$ keeps track of the degree to which pixels still need to be explained. In addition, a map of "cluster seed scores" is initialised by sampling from a uniform distribution $\mathbf{c} \sim \mathcal{U}(0, 1) \in \mathbb{R}^{H \times W}$. These scores determine which input

embeddings are selected as cluster seeds. Now, the IC-SBP first selects a cluster seed from the input embeddings at the spatial location corresponding to the argmax of the seed scores. Next, an unnormalised alpha mask $\alpha \in [0,1]^{H \times W}$ is computed as the distance between the seed embedding and each individual pixel embedding according to a distance kernel ψ whose output is unity if two embeddings are identical and it decreases to zero as the distance between the embeddings increases. The element-wise multiplication of the alpha masks by the current scope then returns the normalised mask and the scope is updated by an element-wise multiplication with one minus the alpha masks similar as in Section 3.2.1. Lastly and crucially, the seed scores are also multiplied by one minus the alpha masks to ensure that cluster seeds are sampled from pixel embeddings that have not yet been assigned to an instance. Once some form of stopping condition is satisfied, the final scope is added as an additional instance mask to explain any remaining pixels. Rather than adding the remaining pixels to the set of masks, the previous masks could alternatively be re-normalised to ensure that they sum to one as an alternative, or the fraction of remaining pixels could also be treated as "unassigned". The IC-SBP is summarised in Algorithm 1.

Algorithm 1: Pseudocode for the IC-SBP

Input: embeddings $\zeta \in \mathbb{R}^{H \times W \times D_\zeta}$
Output: masks $\mathbf{m}_{1:K}$ with $\mathbf{m}_k \in [0,1]^{H \times W}$
Initialise:
masks $\mathbf{m}_{1:K} = \{\,\}$,
scope $\mathbf{s} = \mathbb{1}^{H \times W}$,
cluster_seed_scores $\mathbf{c} \sim \mathcal{U}(0,1) \in \mathbb{R}^{H \times W}$

while *not StopCondition(masks)* **do**
    i, j = argmax(cluster_seed_scores, (i, j));
    seed = embeddings[i, j];
    alpha_mask = DistanceKernel(embeddings, seed);
    instance_mask = scope $\odot$ alpha_mask;
    masks.add(instance_mask);
    scope = scope $\odot$ (1 - alpha_mask);
    cluster_seed_scores = cluster_seed_scores $\odot$ (1 - alpha_mask);
end
masks.add(scope)

The IC-SBP has several interesting properties. To begin with, the masks have a random ordering due to the stochastic sampling of cluster seeds. It is consequently not possible for a model to learn a fixed, sequential decomposition strategy as in Chapter 3. The masks and the resulting object representations are therefore symmetric. This does not strictly apply to the last mask when it is set equal to the remaining scope. However, we observe empirically that models typically learn a strategy where later components remain unused or the final component corresponds to a generic "background cluster" with foreground objects remaining unordered as desired. Furthermore, it is possible to backpropagate gradients directly through the soft instance masks to the instance embeddings. This is facilitated by the random selection of cluster seeds from the input embeddings. While any input embedding can be selected as a cluster seed, the algorithm avoids multiple seeds to be sampled from the same object as the stochastic seed scores are reduced for pixels that have already been partially of fully explained. Due to this stochastic sampling procedure, several executions of the algorithm lead to slightly different outputs when the mask values are between zero and one. However, in the limit as mask values become discrete and exactly equal to zero or one, different executions of algorithm always produces the same binary masks. Hence, the model is implicitly encouraged to drive segmentations towards binary instance assignments in order to minimise the effect of the stochasticity introduced by the sampling procedure. The stopping condition of the IC-SBP can in principle be dependent on the current set of instance masks. For example, iterations can be performed until a certain fraction of the input has been assigned to clusters. Nevertheless, in this work we run IC-SBP for a fixed number of K iterations for simplicity, where K is set heuristically for each dataset as in Chapter 3.

Finally, we note that the component masks of the SGMM rather than the attention masks from the IC-SBP are used as the instance segmentation masks for computing the quantitative metrics. While the IC-SBP attention masks also correlate with objects in the scenes, we found empirically that unlike the SGMM masks, they do not necessarily follow the object boundaries as closely and therefore

achieve lower quantitative scores. This is not surprising given that the receptive field of each element in the pixel-wise instance embedding map can be very large and therefore contain information that is not necessarily related to the associated pixel. From an application point of view, in some circumstances it might be desirable in terms of computational efficiency to only run the attention network to identify objects. In this case, attention masks could be encouraged to be more similar to the SGMM masks with an additional loss term in the training objective. We conjecture that using a backbone CNN with a small receptive field as in Locatello et al. [36] might also lead the attention masks to follow object boundaries more closely.

4.2.2 Semi-Convolutional Embeddings

Now that we have a means for clustering instance embeddings ζ into individual segmentation masks $\mathbf{m}_{1:K}$, we consider how to actually obtain these embeddings. Novotny et al. [41] show that fully translation-invariant CNNs are not able to predict distinct instance embeddings for several objects with identical appearances. They propose to overcome this limitation by utilising "semi-convolutional embeddings". These are constructed by the element-wise addition of relative pixel coordinate values to two channels of each instance embedding. In this work, we use relative pixel coordinates that are linearly spaced in the interval $[-1, 1]$. To obtain the same embedding for a set of pixels, this effectively requires the model to predict a delta vector for each embedding to some shared pixel location, for example the centre of an object. We adopt this paradigm in GENESIS++ given its sound theoretical justification and the positive empirical results in Novotny et al. [41]. Beyond the benefits described in Novotny et al. [41], we argue that semi-convolutional embeddings also have an implicit inductive bias that instance masks should be spatially localised: if the initially predicted delta vectors are zero or small random values, then the resulting instances are localised patches. Given that physical objects are usually spatially localised, this is likely a useful bias. Furthermore, the deltas are also an intuitive quantity to describe object shape. For example, a penalty on the magnitude of the delta vectors could be used as an explicit bias on object size.

Moreover, the IC-SBP requires the definition of a kernel ψ for computing the distances between input embeddings and cluster seeds. Established candidates for this from the literature are the Gaussian ψ_G, Laplacian ψ_L, and Epanechnikov ψ_E kernels (Novotny et al. [41] and Kong and Fowlkes [134]), which can be written as:

$$\psi_G(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{\sigma_G}\right), \tag{4.1}$$

$$\psi_L(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||}{\sigma_L}\right), \tag{4.2}$$

$$\psi_E(\mathbf{u}, \mathbf{v}) = \max\left(1 - \frac{||\mathbf{u} - \mathbf{v}||^2}{\sigma_E}, 0\right), \tag{4.3}$$

where $\mathbf{u}$ and $\mathbf{v}$ are two embeddings of equal dimension. Notably, each these three kernels contains a scaling factor σ which can either be fixed or learned. To give the models more flexibility, we decide to treat σ as a learnable parameter. Moreover, based on the structure of the semi-convolutional embeddings, it is possible to derive principled initialisations for σ in the different kernels so that the initial instance masks are approximately equally-sized circles.

Without loss of generality, assume the relative pixel coordinates in the semi-convolutional embeddings are linearly spaced in the interval $[-1, 1]$ and that we expect a maximum of K scene components to be present in an image. We observe that all three kernels considered above have circular isocontours of equal probability. To initialise masks in a way so that each individual mask covers approximately the same area of an image, let these circular kernels be packed into the image in a square fashion. Thus, dividing an image into K equally sized squares, then each square has a side-length of $2/\sqrt{K}$. Assuming a kernel is centred in a square, we want the mask probability to decrease to 0.5 at the intersection of the square and the circular isocontour, i.e. at a distance of $1/\sqrt{K}$ from the centre of the kernel. This argument is illustrated for an example with $K = 4$ in Figure 4.2. We can therefore write:

$$\psi\left(0, 1/\sqrt{K}\right) = 0.5 \iff \sigma_G = \frac{1}{K \ln 2}, \quad \sigma_L = \frac{1}{\sqrt{K} \ln 2}, \quad \sigma_E = \frac{2}{K}. \tag{4.4}$$

Several examples of the initial masks obtained when running the IC-SBP multiple times with the proposed initialisation scheme and with $K = 7$ are shown

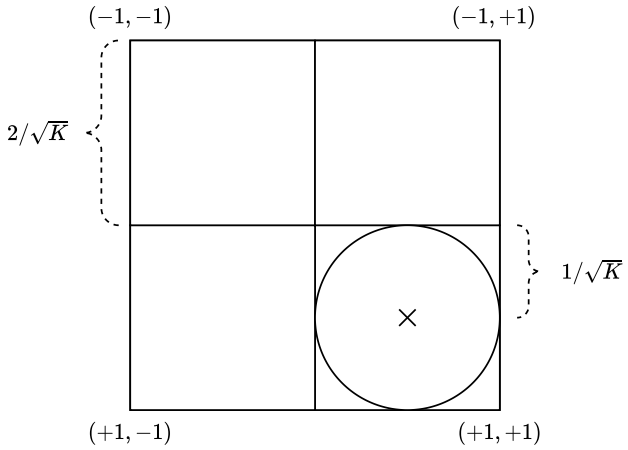
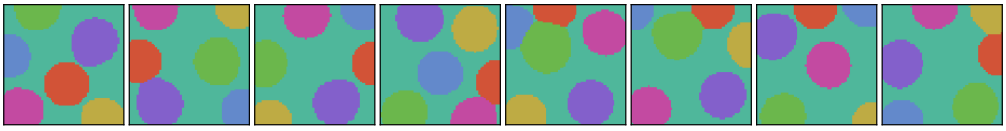
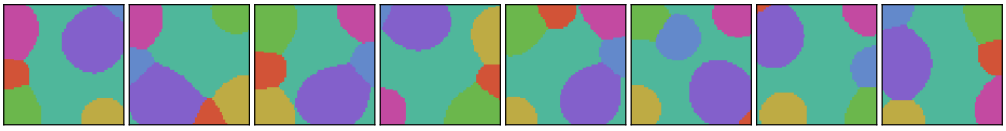


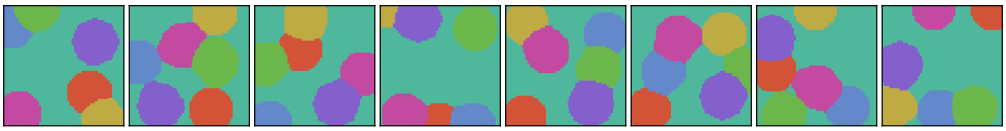
Figure 4.2: Illustration of packing $K = 4$ circles into a square image with relative pixel coordinates being linearly spaced in the interval $[-1, 1]$. In the resulting configuration, the radius of each circle is equal to $1/\sqrt{K}$.



(a) Gaussian



(b) Laplacian



(c) Epanechnikov

Figure 4.3: Initial masks obtained when running the IC-SBP several times with the initialisation in Equation (4.4) with $K = 7$. The final mask is set to the remaining scope.

in Figure 4.3 for these three kernels. In Section 4.3.3, $K = 7$ is also used for experiments on the ObjectsRoom dataset (Burgess et al. [25] and Kabra et al. [103]). It might be argued that in this case the initialisations should be adapted and only consider the $K - 1$ randomly sampled masks. Nevertheless, we conjecture that the resulting empirical difference is likely small and an investigation of this is left for future work. Moreover, the initialisations can likely be further improved by considering the stochastic sampling procedure of the IC-SBP.

4.2.3 Autoregressive Prior with Greedy Sorting

Now that have a mechanism for inferring a set of unordered objects representations $\mathbf{z}_{1:K}$, we turn our attention to how we might learn an autoregressive prior to facilitate the object-centric generation of coherent scenes as in Chapter 3. We could opt to train an autoregressive prior of the same form $p_\theta(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})$ as described in Section 3.2.1 without taking any further measures. The fact that the posterior object latents are not ordered anymore, however, potentially renders this task more difficult. The prior on $\mathbf{z}_k$ is conditioned on $\mathbf{z}_{1:k-1}$, but $\mathbf{z}_{1:k-1}$ can now have any arbitrary order and $\mathbf{z}_k$ could be any element in the set $\mathbf{z}_{k:K}$ where K is the total number of components as before. Consequently, $p_\theta(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})$ needs to capture the distribution over all possible object latents $\mathbf{z}_{k:K}$ rather than just a single component that can be anticipated when a fixed, sequential inference strategy is followed. This would likely require a larger neural network to parameterise $p_\theta(\mathbf{z}_k \mid \mathbf{z}_{1:k-1})$, compared to using a sequential inference procedure that orders the object representations as in Chapter 3. As an alternative, we propose to greedily sort the posterior latents when computing the Kullback-Leibler (KL)-divergence between posterior and prior during training. This is done by greedily assigning the latent posterior with the minimum KL-divergence with respect to the current prior at every step. Simultaneously, the associated sorted posterior latents are used as inputs to condition the prior for the next step. This procedure is summarised in Algorithm 2.

Algorithm 2: Pseudocode for learning a sorted prior

Input: $\mathbf{z}_{1:K}$, $q_\phi(\mathbf{z}_{1:K})$
Output: $\mathrm{KL}_{1:K}$
Initialise: $\mathrm{KL}_{1:K} = [\,]$, $\mathbf{z}^{\text{sorted}}_{1:K} = [\,]$
for $k = 1, \ldots, K$ **do**
  $k' = \text{argmin}(\mathrm{KL}[q_\phi(\mathbf{z}_{k'}) \mid\mid p_\theta\Big(\mathbf{z}_{k'} \mid \mathbf{z}^{\text{sorted}}_{<k}\Big)],\ k')$;
  $\mathrm{KL}_{1:K}.\text{append}(\mathrm{KL}[q_\phi(\mathbf{z}_{k'}) \mid\mid p_\theta\Big(\mathbf{z}_{k'} \mid \mathbf{z}^{\text{sorted}}_{<k}\Big)])$;
  $\mathbf{z}^{\text{sorted}}_{1:K}.\text{append}(\mathbf{z}_{k'})$;
  $\mathbf{z}_{1:K}.\text{pop}(k')$;
  $q_\phi(\mathbf{z}_{1:K}).\text{pop}(k')$;
end

Consequently, even though the inference model learns to infer an unordered set of object representations $\mathbf{z}_{1:K}$, the generative model can learn a fixed, sequential generation strategy with this "sorted autoregressive prior", which should be easier to model when $p_\theta(\mathbf{z}_k \mid \mathbf{z}_{<k})$ is parameterised by a neural network with finite representational capacity. However, introducing an ordering in the generative model is arguably a less general solution and the greedy assignment Algorithm 2 might have a negative impact on the optimisation procedure. Hence, if the generative network is large enough, then training an "unsorted autoregressive prior" possibly leads to both a better and more general solution. This trade-off is investigated empirically in the next section.

4.3 Experiments

We now proceed to evaluate the utility of the proposed methods. In particular, we investigate the suitability of IC-SBP for learning object-centric representations, whether the choice of the kernel function ψ has a significant impact on model behaviour, and how unsorted and sorted autoregressive prior compare to each other. Finally, we want to determine whether the resulting model is capable of learning object-like decompositions of real-world images as might be encountered in typical robotics applications. To this end, we split the experimental sections into two parts. Firstly, we validate the scene decomposition and generation performance of GENESIS++ on two simulated datasets—ShapeStacks (Groth et al. [88]) and ObjectsRoom (Burgess et al. [25] and Kabra et al. [103])—where ground truth segmentation masks are available. In particular, we compare variants of GENESIS++ trained with different segmentation kernels ψ and with either an unsorted or a sorted autoregressive prior. These results are also compared against GENESIS and our re-implementation of MONET as developed in Chapter 3. Secondly, we evaluate GENESIS++, GENESIS, and MONET on three real-world datasets collected in the context of robotics applications: Sketchy (Cabi et al. [139]), BAIR Robot Pushing (Ebert et al. [140]), and the MIT-Princeton Amazon Picking

Challenge 2016 Object Segmentation Training Dataset (Zeng et al. [141]). For brevity, we refer to this third dataset as the "APC dataset".

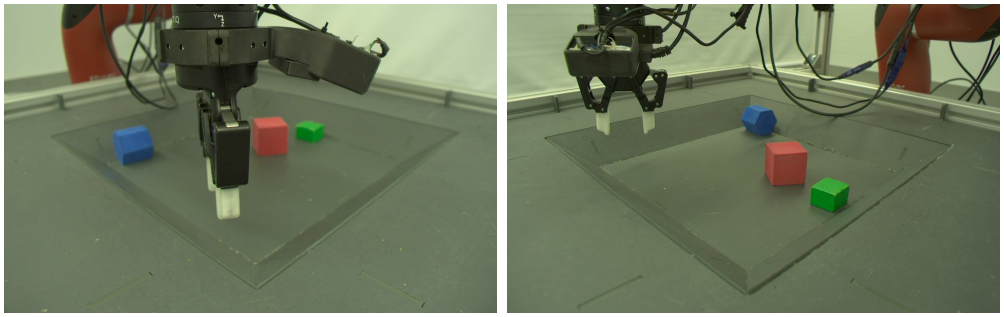
4.3.1 Datasets

The ShapeStacks (Groth et al. [88]) and ObjectsRoom (Burgess et al. [25] and Kabra et al. [103]) datasets were already introduced in Chapter 3, Section 3.3.1. We therefore do not describe them here again to avoid repetition. After the publication of GENESIS in Engelcke et al. [43], both datasets have also been used in more recent works such as Yang et al. [34] and Ehrhardt et al. [38]. The Sketchy, BAIR, and APC are not yet as well established in the context of OCGMs and we proceed by providing detailed descriptions of these datasets as well as our pre-processing procedures. A summary of the details of these three datasets after following the associated pre-processing procedures is provided in Table 4.1. We proceed by describing these datasets in more detail.

Table 4.1: Summary of the details of the three real-world datasets considered in this chapter after pre-processing.

	Train images	Valid. images	Test images	Resolution
Sketchy	337,498	41,426	41,426	128-by-128
BAIR	1,147,920	150,000	7,680	32-by-32
APC	109,281	13,644	13,650	128-by-128

Sketchy (Cabi et al. [139]) The Sketchy dataset is designed for off-policy Reinforcement Learning (RL), providing episodes showing a robotic arm performing different tasks that involve three differently coloured shapes (blue, red, green) or a cloth. The dataset includes camera images from several view points, depth images, joint angles, rewards, and other meta-data. The dataset is quite considerable in size and takes about 5TB of storage in total. As demonstrated in Chapter 3, it is not necessary to use all of this data to train well-performing OCGMs. We therefore ease the computational and storage demands by only using a subset of this dataset. Specifically, we use the high-quality demonstrations from the "lift-green" and "stack-green-on-red" tasks, corresponding to a total of 395 episodes, 10% of

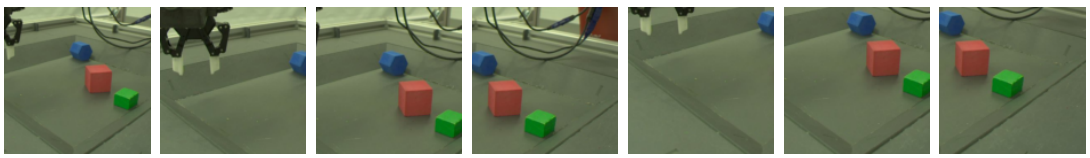


(a) Front-left camera **(b)** Front-right camera

Figure 4.4: Two raw views of a single scene from the Sketchy dataset (Cabi et al. [139]).



(a) Front-left camera



(b) Front-left camera

Figure 4.5: Seven 128-by-128 crops as used for training from the front-left and front-right cameras of a single scene from the Sketchy dataset. Showing from left to right: centre, top-left, top-centre, top-right, bottom-left, bottom-centre, and bottom-right crops.

which are set aside as validation and test sets each. Sketchy also contains episodes from a task that involves lifting a cloth and an even larger number of lower-quality demonstrations that offer a wider coverage of the state space. We restrict ourselves to the high-quality episodes that involve the manipulation of solid objects. The number of high-quality episodes alone is already considerable and we want to evaluate whether the models can separate multiple foreground objects. From the these episodes, we use the images from the front-left and front-right cameras, which show the arm and the foreground objects without obstruction. A pair of example images is shown in Figure 4.4.[1] The raw images have a resolution of 600-by-960 pixels. To remove uninteresting pixels belonging to the walls in the background and the surface around the pit where the objects are placed, 144 pixels on the left and

[1]The Sketchy dataset is available at `https://github.com/deepmind/deepmind-research/tree/master/sketchy`.

right are cropped away for both camera views, the top 71 and bottom 81 pixels are cropped away for the front-left view, and the top 91 and bottom 61 are cropped away for the front-right view, resulting in a 448-by-672 crop. From this 448-by-672 crop, seven square crops are extracted to obtain a variety of views for the models to learn from. The first crop corresponds to the centre 448-by-448 pixels. For the other six crops, the top and bottom left, centre, and right squares of size 352 are extracted. Finally, we resize these crops to a resolution of 128-by-128 to reduce the computational demands of training the models. This leads to a total of 337,498 training; 41,426 validation; and 41,426 test images. Example images obtained with this pre-processing procedure are shown in Figure 4.5.

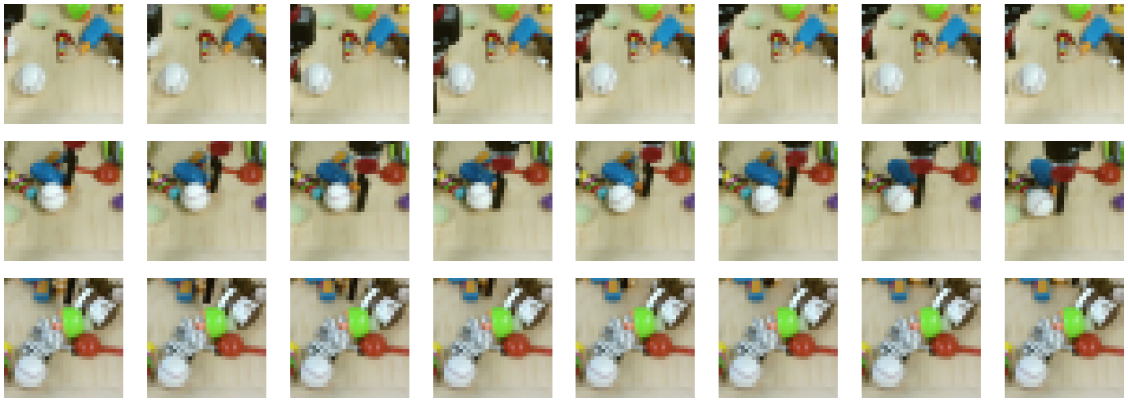
BAIR (Ebert et al. [140]) The BAIR Robot Pushing dataset shows top-down 64-by-64 images of a robot arm pushing toys in a box. This dataset is visually a lot more difficult compared to Sketchy due to the low image resolution and scenes being very cluttered with a large number of diverse objects that are often grouped together towards the edges of the box. To remove uninteresting background pixels belonging to the edges of the box and to reduce the number of objects per input image, we crop away [16, 16, 20, 12] pixels from the top, bottom, left, and right, respectively, resulting in 32-by-32 crops. Several example raw images and the associated cropped images are shown in Figure 4.6 for eight frames from three episodes.[2] The official training and test set consist of 43,264 and 256 episodes, respectively, with each episode containing 30 images. From this official training set, we set aside 5,000 episodes for validation and use the remaining 38,264 episodes for actual training.

APC (Zeng et al. [141]) For their entry to the 2016 Amazon Picking Challenge (APC), the MIT-Princeton team created and released an object segmentation training set, showing a single challenge object either on a shelf or in a red tray. Four example images from the raw dataset are shown in Figure 4.7.[3] The raw images are first resized so that the shorter image side has a length of 128 pixels and the

[2]The BAIR dataset is available at `https://sites.google.com/view/sna-visual-mpc/`.
[3]The APC dataset is available at `https://vision.princeton.edu/projects/2016/apc/`.



(a) Raw image sequences



(b) Corresponding cropped images

Figure 4.6: Three example sequences from the BAIR dataset (Ebert et al. [140]).

centre 128-by-128 pixels are extracted to remove uninteresting pixels belonging to the shelf or tray. The example images from Figure 4.7 after processing are shown in Figure 4.8. For each object, there exists a set of scenes showing the object in different poses on both the shelf and in the red tray. For each scene, there are images taken from different camera viewpoints. We select 10% of the scenes at random to be set aside for validation and testing each, so that scenes between the training, validation, and test sets do not overlap. The resulting training, validation, and test sets consist of 109,281; 13,644; and 13,650 images, respectively.

4.3.2 Implementation Details

We follow the same training details as in Chapter 3, Section 3.3.4: the Generalised ELBO with Constrained Optimisation (GECO) objective (Section 3.2.3) with a pixel reconstruction goal of 0.5655 is minimised using the Adaptive Moment



Figure 4.7: Four raw images from the APC dataset (Zeng et al. [141]).



Figure 4.8: Four images from the APC dataset after cropping and resizing.

Estimation (ADAM) optimiser (Kingma and Ba [117]), a learning rate of 10^{-4}, and a batch size of 32 for 500,000 iterations. Conditional prior and posterior distributions are diagonal Gaussian distributions. For the pixel likelihood, the Gaussian standard deviation in the SGMM is set to 0.7 (Section 3.2.1 and Section 4.2). The scaling factors in the IC-SBP kernels are initialised as described in Section 4.2.2 and jointly optimised. As in Chapter 3, we set the number of components to $K = 9$ for ShapeStacks and $K = 7$ for ObjectsRoom. For the additional real-world datasets that are considered in this chapter, we increase the number of maximum scene components to $K = 10$, $K = 12$, and $K = 10$ for the Sketchy, BAIR, and APC datasets, respectively. This is to accommodate for the increased visual complexity of these datasets. In particular the BAIR dataset contains images that are fairly cluttered. Finally, the GENESIS++ architecture consists of four main components: a convolutional backbone, the attention and feature pooling module, the component decoders, and an optional autoregressive prior. The details of these four components are described in the following.

Backbone GENESIS++ uses a single UNet-like (Ronneberger et al. [116]) encoder similar to the attention network in our re-implementation of MONET (Burgess et al. [25]) with [64, 64, 128, 128, 128] filters in the encoder and the reverse

in the decoder (Section 3.3.3). The only difference is that the Instance Normalisation (IN) layers (Ulyanov et al. [142]) are replaced with Group Normalisation (GN) layers (Wu and He [143]) to preserve contrast information. The number of groups is set to eight in all such layers, which is also referred to as a "GN8" layer. The output of this backbone encoder is a feature map $\mathbf{e}$ with $D_e = 64$ output channels and spatial dimensions that are equal to the height and width of the input image.

Attention and Object Pooling Following feature extraction, an "attention head" computes pixel-wise instance embeddings of dimension eight in a semi-convolutional fashion as in Novotny et al. [41], i.e. $D_\zeta = 8$. It consists of a 3×3 Conv-GN8-Rectified Linear Unit (RELU) block with 64 filters and a 1×1 semi-convolutional layer. The semi-convolutional instance embeddings are clustered into K attention masks $\mathbf{m}_k$ using the IC-SBP. A "feature head" consisting of another 3×3 Conv-GN8-RELU block with 64 filters and a 1×1 convolution with 128 filters refines the encoder output $\mathbf{e}$ to obtain a feature map $\mathbf{f} \in \mathbb{R}^{H \times W \times D_f}$ with $D_f = 128$. Similar to Locatello et al. [36], the attention masks $\mathbf{m}_{1:K}$ are used to pool features from $\mathbf{f}$ by multiplying the feature map with a given attention mask and summing across the spatial dimensions. These pooled features are normalised by dividing by the sum of the attention mask values plus a small epsilon value to avoid numerical instabilities. Next, a "posterior head" consisting of a layer normalisation (Ba et al. [144]), a Fully-Connected (FC)-RELU block with 128 units and a second FC layer takes as input the pooled component features and computes the sufficient statistics of the individual component latents $\mathbf{z}_k \in \mathbb{R}^{64}$.

Component Decoders Following Greff et al. [26] and Locatello et al. [36], the object latents are decoded by separate component decoders with shared weights to parameterise the sufficient statistics of a SGMM. Each decoded component has four channels per pixel. The first three channels are the RGB values describing the component appearances and the fourth channel are the unnormalised segmentation logits which are normalised across scene components using a softmax operator. Again following Locatello et al. [36], the first layer is a spatial broadcasting module

Table 4.2: GENESIS++ component decoder.

Input spatial dim.	Input channels	Output channels	Kernel size	Stride	Layer type	Activation
1	64	64+2	-	-	Broadcast	-
img dim/16	64+2	64	5	2	Deconv	GN8-RELU
img dim/8	64	64	5	2	Deconv	GN8-RELU
img dim/4	64	64	5	2	Deconv	GN8-RELU
img dim/2	64	64	5	2	Deconv	GN8-RELU
img dim	64	4	1	1	Conv	-

as introduced in Watters et al. [115] and previously used in Chapter 3, Section 3.3.3. An additional advantage of spatial broadcasting, aside from encouraging "disentanglement" [115], is that is requires a smaller number of parameters than a fully-connected layer when upsampling a feature vector to a specific spatial resolution. In contrast to GENESIS, however, we do not use spatial broadcasting for upsampling to the full image resolution. Instead, the spatial broadcasting module is followed by four stride-2 deconvolutional layers to retrieve the full image resolution. We found in preliminary experiments that a more expressive decoder can be used in comparison to GENESIS, improving the quality of the results without the model collapsing to a single component as in the investigation on reconstruction bottlenecks in Chapter 3, Section 3.3.8. We conjecture that this is facilitated by the IC-SBP and the semi-convolutional embeddings but this would need to be confirmed with a more comprehensive empirical investigation, a task left for future work. A summary of the architectural details of the component decoder is provided in Table 4.2.

Autoregressive Prior Identical to GENESIS as described in Chapter 3 (Section 3.3.3 and Table 3.6), the autoregressive prior for scene generation is implemented as a Long Short-Term Memory (LSTM) cell (Hochreiter and Schmidhuber [114]) followed by a fully-connected linear layer to infer the sufficient statistics of the prior distribution for each component.

4.3.3 Benchmarking in Simulation

We begin by examining the segmentation performance of GENESIS++ in simulation on the ShapeStacks and ObjectsRoom datasets. The availability of ground truth segmentation labels allows us to evaluate quantitatively whether the IC-SBP as described in Section 4.2.1 is indeed useful for learning object-centric representations before conducting experiments on real-world datasets in Section 4.3.4. We perform experiments with the different segmentation kernels ψ considered in Section 4.2.2 to determine whether the choice of kernel has a significant impact on segmentation performance. Finally, we select one of these kernel function to investigate the impact of an unsorted and a sorted autoregressive prior as described in Section 4.2.3 on both decomposition and generation performance.

4.3.3.1 Scene Decomposition

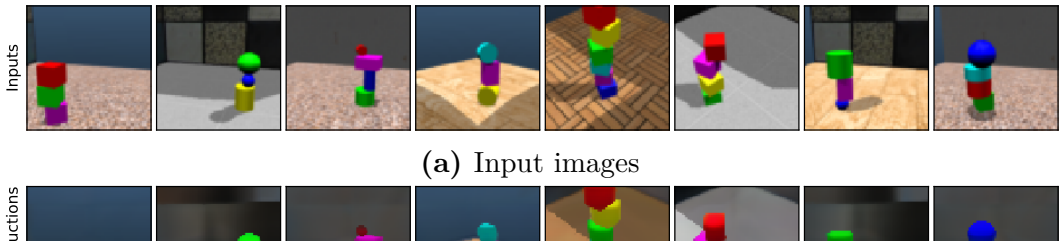
As in Section 3.3.5 of Chapter 3, we evaluate the quality of scene segmentations using the Adjusted Rand Index (ARI) and the Mean Segmentation Covering (MSC) of foreground pixels as described in Section 3.3.2. These averaged over 320 images from the test set of the ShapeStacks and ObjectsRoom datasets. Specifically, we report the means and standard deviations from models trained with three different random seeds for GENESIS++, GENESIS, and our re-implementation of MONET (Section 3.3.3) in Table 4.3. To begin, we consider GENESIS++ trained with an independent prior and Gaussian ψ_G, Laplacian ψ_L, and Epanechnikov ψ_E kernels and an independent component prior. GENESIS++ outperforms GENESIS and MONET in three of the four metrics for all three kernels. GENESIS manages to outperform GENESIS++ on the fourth metric which is the MSC on the ObjectsRoom datasets, Notably, for this metric the standard deviation of GENESIS++ is quite large for the Gaussian and the Laplacian kernels. We investigated the individual runs and found that in both of these two cases, one of the three models undersegments the foreground objects and therefore achieves a much lower MSC. The Gaussian kernel performs on par with the Laplacian kernel on both ShapeStacks metrics. For ObjectsRoom, the Gaussian kernel achieves the

Table 4.3: Means and standard deviations of the segmentation metrics on the ShapeStacks and ObjectsRoom test sets from three random seeds.

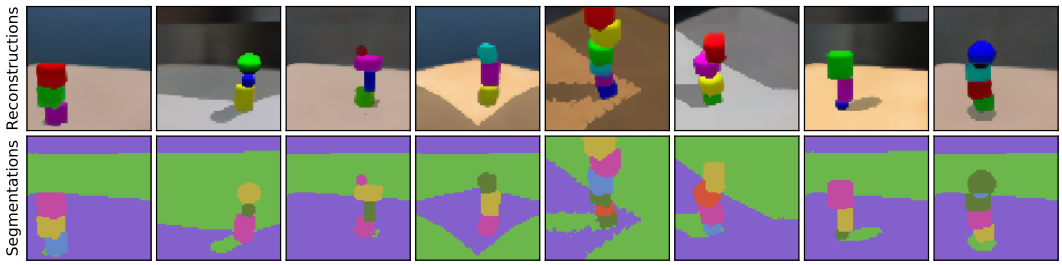
		ShapeStacks		ObjectsRoom	
Model	Variant	ARI	MSC	ARI	MSC
MONET	-	0.70±0.04	0.57±0.12	0.54±0.00	0.33±0.01
GENESIS	-	0.70±0.05	0.67±0.02	0.63±0.03	0.53±0.07
GENESIS++ (ind. prior)	ψ_G	0.79±0.01	0.67±0.00	0.79±0.01	0.47±0.17
	ψ_L	0.79±0.01	0.67±0.01	0.74±0.08	0.48±0.20
	ψ_E	0.78±0.01	0.66±0.01	0.78±0.01	0.34±0.08
GENESIS++ (a.r. prior, ψ_G)	unsorted	**0.81±0.00**	**0.68±0.01**	**0.84±0.01**	**0.58±0.03**
	sorted	0.79±0.01	0.65±0.00	**0.84±0.00**	0.53±0.09

best ARI and the Laplacian kernel the lowest. While the Laplacian kernel has a slightly better MSC on ObjectsRoom than the Gaussian kernel, the latter appears to be the most robust in terms of segmentation performance and is therefore used in all later experiments. When examine the effect of learning an autoregressive prior on scene segmentation, we find that the unsorted prior performs best overall in terms of decomposition performance, while models trained with the sorted prior achieve comparable or slightly better performance than the models trained with independent priors. The results indicate that modelling correlations between scene components in the generative model appears to have a positive effect on decomposition performance, possibly because the generative model is more closely aligned with the true data generation process. The fact that the unsorted prior consistently achieves better decomposition performance than the sorted prior is a possible indication that the greedy sorting procedure might have a negative impact on the optimisation procedure.

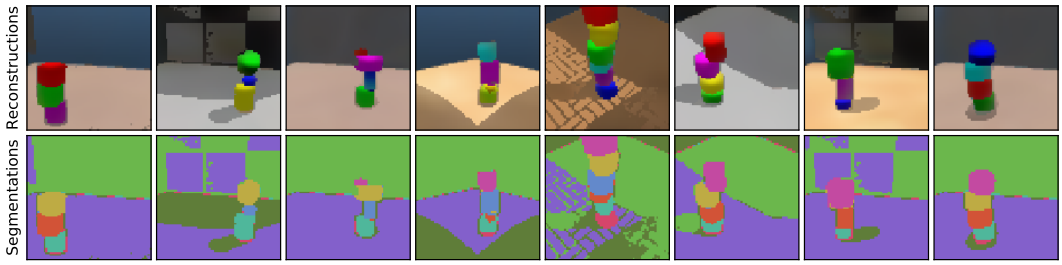
Figures 4.9 to 4.12 show qualitative results for our re-implementation of MONET, GENESIS, and GENESIS++ on the ShapeStacks and the ObjectsRoom datasets. These results are obtained with the random seeds that achieve the highest ARI for each model type. For both datasets, we find that GENESIS++ better separates the individual foreground objects than the baseline models. Furthermore,



(a) Input images



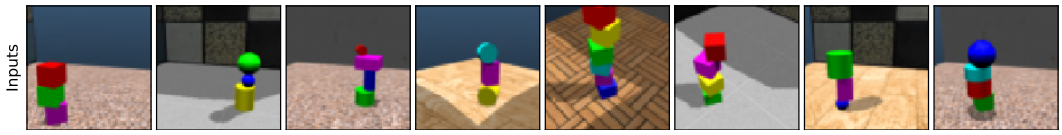
(b) MONET



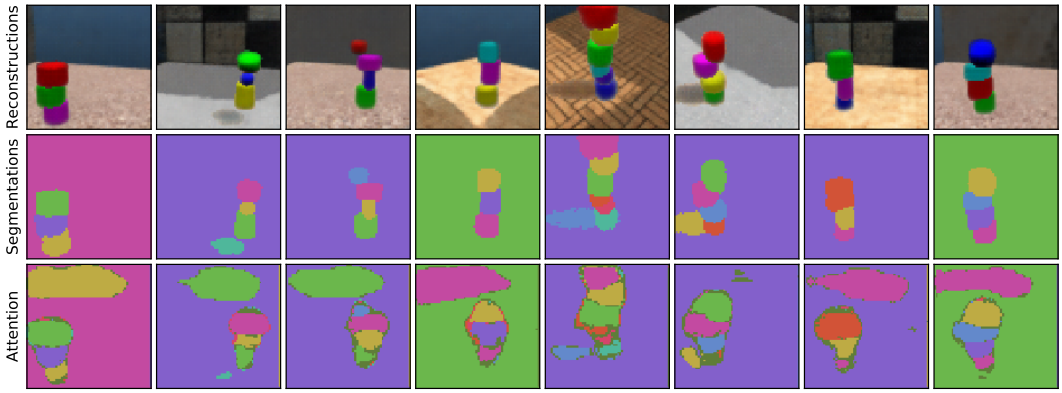
(c) GENESIS

Figure 4.9: ShapeStacks reconstructions and segmentations with GENESIS and MONET. Neither of the two models reconstructs fine details of the background textures.

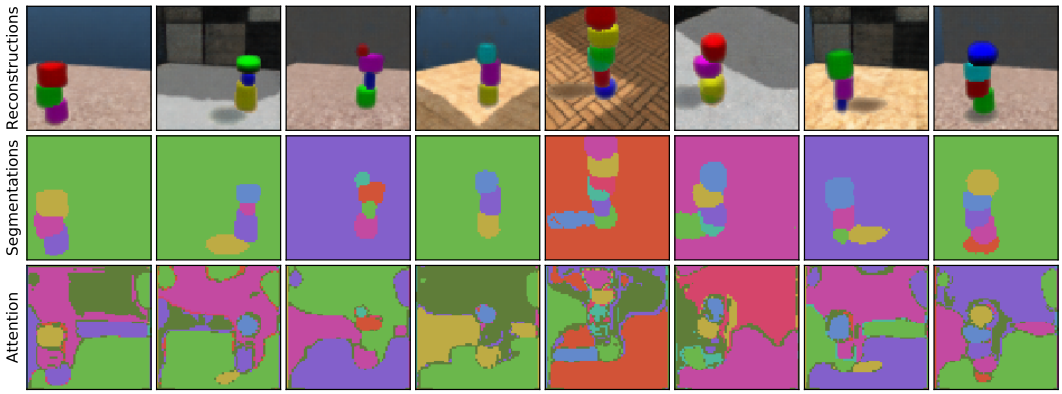
GENESIS++ treats the static scene background as a single component and the reconstruction quality of the background is visually much better; even fine texture details are reconstructed. The figures also include the IC-SBP attention masks for GENESIS++. There is a clear correlation between the attention masks and the scene segmentations as defined by the SGMM, but the attention masks do not follow object boundaries as closely and backgrounds are split into several parts. Given that the receptive field of the encoder backbone covers the whole image due to the fully-connected layers at the lowest resolution of the UNet, it is possible that all background information could be encoded in a small fraction of the final feature map so that it is unnecessary to pool features from all the pixel locations that belong to the background. One aspect that GENESIS++ appears to struggle with is the small, occluded object in the fourth image from the ObjectsRoom dataset in Figure 4.12. Neither of the two models is able to segment this object and it is not



(a) Input images



(b) GENESIS++ with unsorted prior

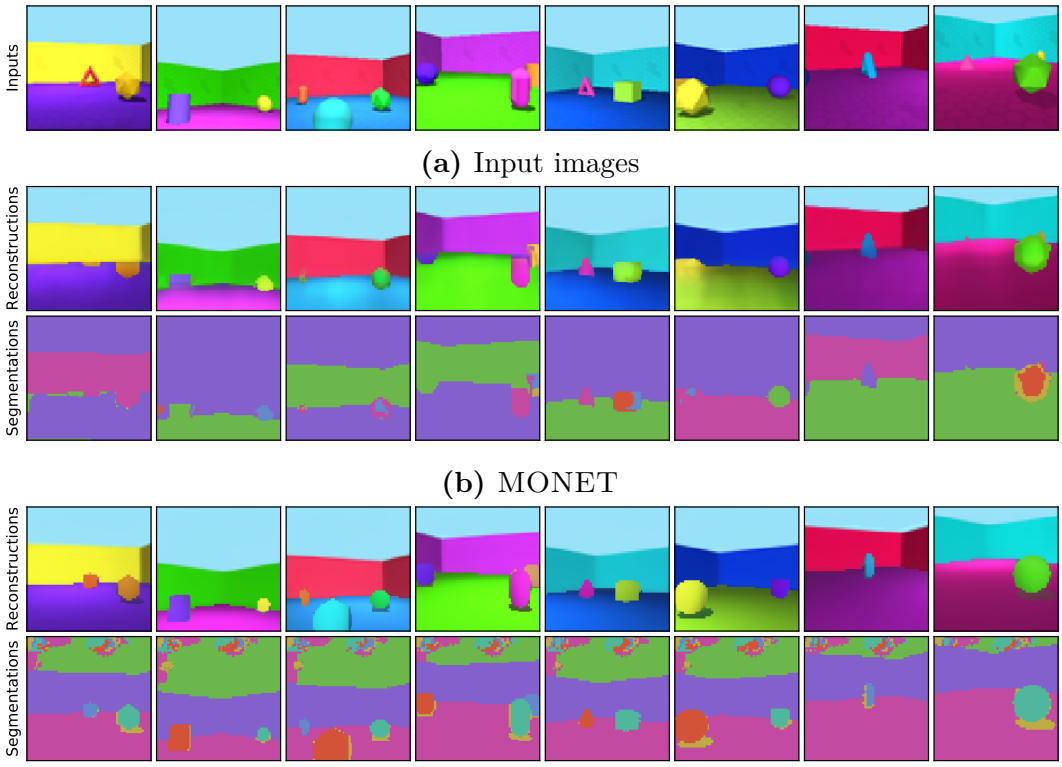


(c) GENESIS++ with sorted prior

Figure 4.10: ShapeStacks reconstructions and segmentations. Both variants of GENESIS++ produce better results than the baselines in Figure 4.9.

present in the reconstructions. It can be observed that the attention masks do not correlate as well with the segmentation masks for the ObjectsRoom dataset and that the small orange object appears to be merged into the background component. It is possible that this might improve if the models are trained for more iterations.

The qualitative results for MONET on the ObjectsRoom dataset in Figure 4.11 are not as good as for the original model in Burgess et al. [25]. It is possible that this is caused by implementation differences, but the most likely explanation is that we train MONET with the same optimisation details as the other models as described in Chapter 3, Section 3.3.4. This is intended to provide an equal basis of comparison,



(a) Input images

(b) MONET

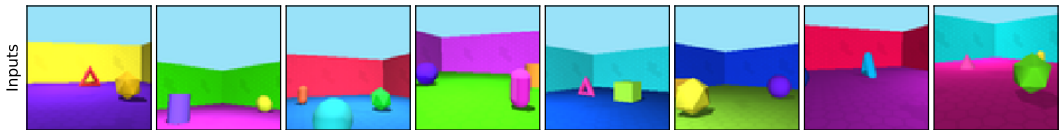
(c) GENESIS

Figure 4.11: ObjectsRoom reconstructions and segmentations with GENESIS and MONET. Both models achieve reasonable results.

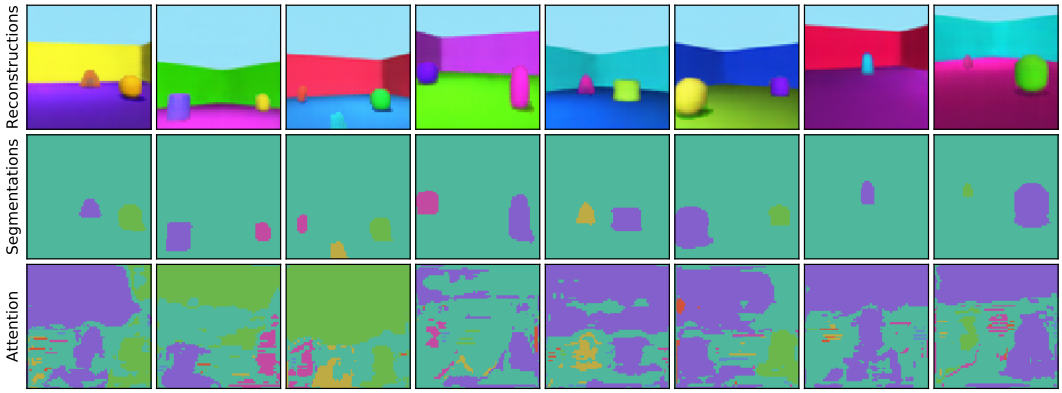
but it is possible that these training details are less favourable for MONET. Nevertheless, this does not significantly impact the conclusions of this chapter, as MONET is ill-suited as an object-centric world model as demonstrated in Chapter 3.

4.3.3.2 Scene Generation

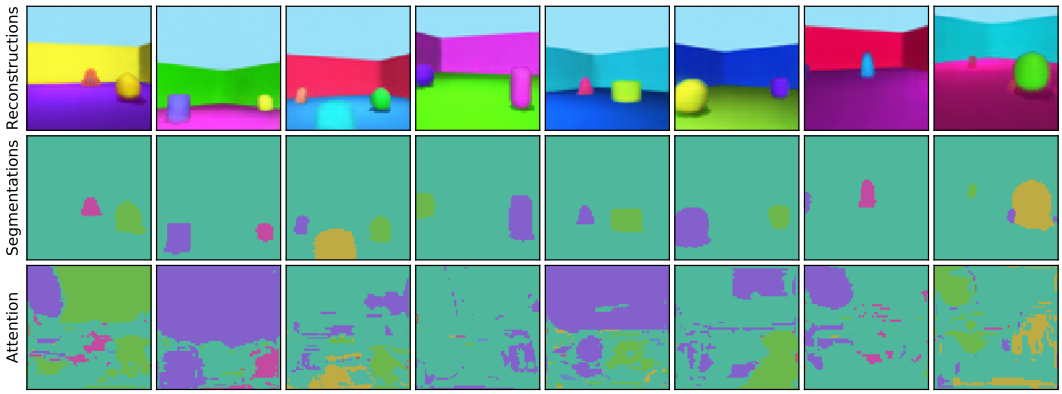
As in Chapter 3, we use the Fréchet Inception Distance (FID) (Heusel et al. [108]) of 10,000 generated and test images each to quantify the quality of generated scenes. Table 4.4 summarises the results for GENESIS++ with unsorted and sorted autoregressive priors, GENESIS, and MONET after training on ShapeStacks and ObjectsRoom. The means and standard deviations are shown for three different random seeds. The two variants of GENESIS++ outperform the baselines on both datasets, with the margins on ShapeStacks being particularly large. The sorted prior performs considerably better than the unsorted prior on ShapeStacks, but the unsorted prior performs slightly better on ObjectsRoom. We conjecture that



(a) Input images



(b) GENESIS++ with unsorted prior



(c) GENESIS++ with sorted prior

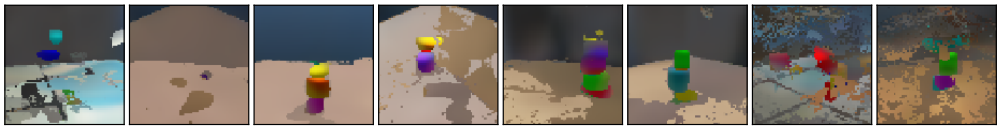
Figure 4.12: ObjectsRoom reconstructions and segmentations. Both variants of GENESIS++ produce qualitatively better results than the baselines in Figure 4.11.

the sorted prior is more useful on the more challenging ShapeStacks dataset. Here, the model struggles to learn a good generative model and it appears that sorting the posterior latents in a greedy fashion makes this task easier. The generated images after training on the ObjectsRoom dataset, though, are comparatively good. The inferior results of the sorted prior in this case possibly indicate that the representational discontinuities introduced by the sorting procedure, as described in Section 4.2.3, might have a negative impact on model optimisation.

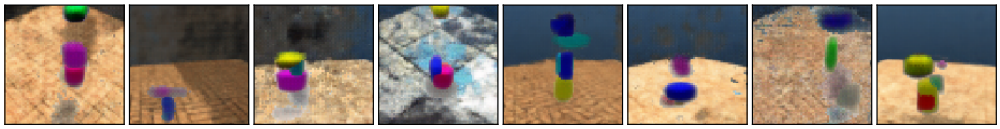
Qualitative results for scene generation are shown in Figures 4.13 and 4.14. Again, these results are obtained with the random seed that achieves the highest ARI for



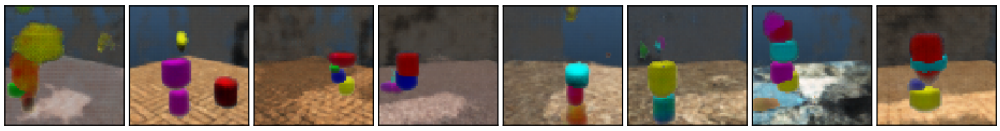
(a) MONET



(b) GENESIS

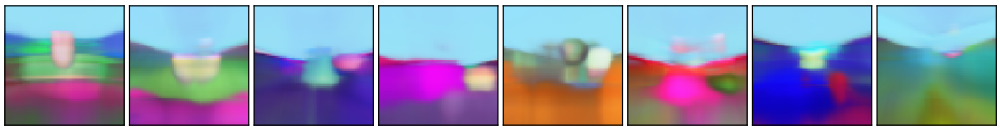


(c) GENESIS++ with unsorted prior

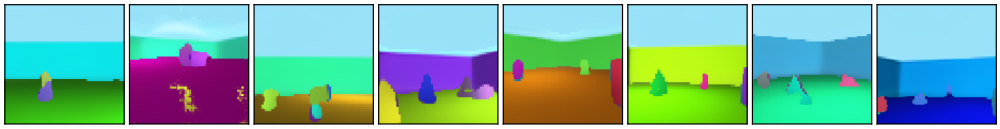


(d) GENESIS++ with sorted prior

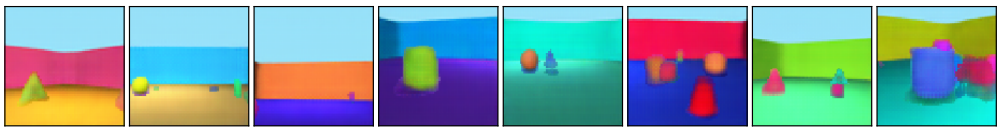
Figure 4.13: Generated images after training on ShapeStacks. Both variants of GENESIS++ produce better results than the baselines.



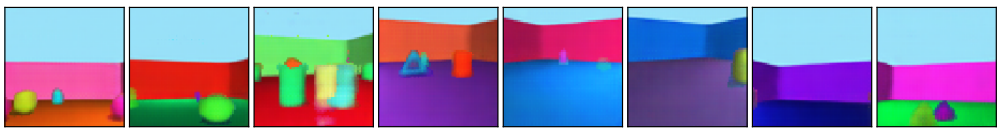
(a) MONET



(b) GENESIS



(c) GENESIS++ with unsorted prior



(d) GENESIS++ with sorted prior

Figure 4.14: Generated images after training on ObjectsRoom. Both variants of GENESIS++ produce comparable results to GENESIS.

Table 4.4: Means and standard deviations of FID scores on the ShapeStacks and ObjectsRoom datasets from three random seeds.

Model	Variant	ShapeStacks	ObjectsRoom
MONET	-	197.8±5.2	205.7±7.6
GENESIS	-	186.8±18.0	62.8±2.5
GENESIS++	unsorted	112.7±3.2	**52.6±2.7**
	sorted	**74.9±5.6**	58.5±9.2

each model. Even though in some cases this selects a model with a lower FID score, we are interested in the quality of the generated images given the best possible decomposition quality. After training on ShapeStacks, it can be observed that GENESIS++ generates higher quality images than MONET and GENESIS as we would expect based on the smaller FID scores in Table 4.4. The quality of generated images after training on ObjectsRoom is not very distinguishable to GENESIS. Nevertheless, we also note that there is still much room left for improvement: individual foreground objects in the generated ShapeStacks images do not form nicely stacked towers and some artefacts can be seen in the generated ObjectsRoom images.

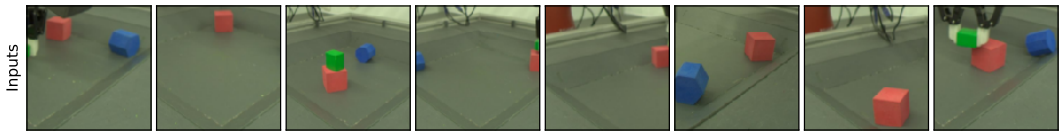
4.3.4 Real-World Decomposition and Generation

Finally, we evaluate GENESIS++ and the two baseline models on three significantly more challenging real-world datasets that were collected in the context of robotic manipulation. The results are presented on the Sketchy (Cabi et al. [139]), BAIR Robot Pushing (Ebert et al. [140]), and APC (Zeng et al. [141]) datasets in Sections 4.3.4.1 to 4.3.4.3, respectively. Similar to before, we evaluate the capabilities of these models both in terms of scene decomposition and scene generation. Both of these capabilities have the potential of increasing the flexibility of robotic systems by facilitating object-based perception and planning as well as the efficient acquisition of new skills in object-centric world models as inspired by Ha and Schmidhuber [29]. As discussed in Section 3.2.3 of Chapter 3, the GECO training objective requires the specification of a numerical goal value for the reconstruction error. Considering the increased level of visual detail in these real-world images, we therefore train the

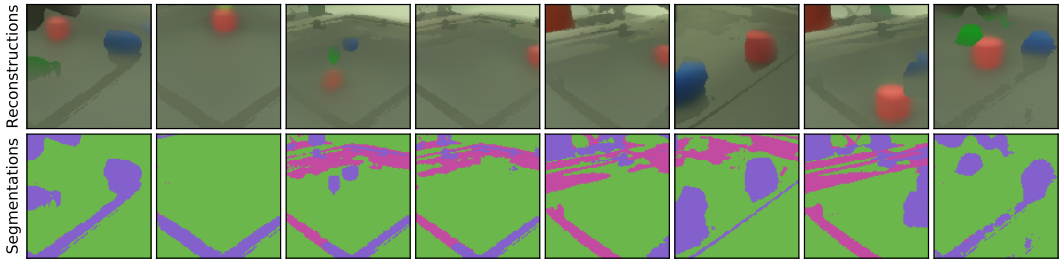
different models both with a goal value of 0.5655 as used previously in Sections 3.3.4 and 4.3.2 as well as two smaller values of 0.5645 and 0.5635. These two values are selected so that the resulting reconstruction qualities are visually different without being overly ambitious. If the goal values are too small so that the models are not able to reach them, then the beta-multiplier in loss function (Equation (3.20)) will remain small, resulting in visually poor samples. The results in Section 4.3.3 indicated that the sorted prior can potentially achieve much better results in terms of scene generation (Table 4.4) at a possibly small degradation of decomposition performance (Table 4.3). For the results in this section, we therefore train GENESIS++ with a sorted autoregressive prior as described in Section 4.2.3.

4.3.4.1 Sketchy

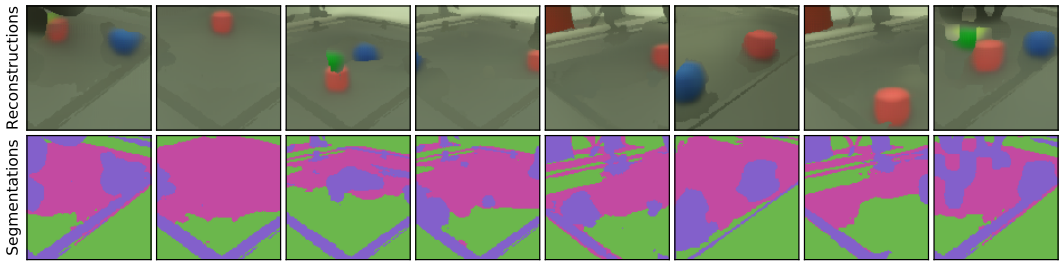
We begin by examining the results on the Sketchy dataset which is arguably the visually simplest dataset of the three. In comparison to Sketchy, the BAIR dataset is very cluttered and lighting conditions in the APC dataset are much more variable as can be seen in the examples in Section 4.3.1. Reconstructions and segmentations for GENESIS, our re-implementation of MONET, and GENESIS++ are shown in Figures 4.15 to 4.17, respectively. Figures 4.18 to 4.20 show several images generated by sampling from the priors of the three models. In Figure 4.15, GENESIS segments the input images into either two or three components for the largest GECO goal. One component explains the floor, and one or two components explain the edge of the pit, the foreground objects, and the gripper. A similar strategy is learned for the medium GECO goal, but the floor is explained as two separate components. For the smallest GECO goal, again the strategy is also similar for the lower half of the images, but the upper half which contains more visual details is segmented into stripes that lack physical interpretability. It is unclear why the model learns such a solution in this more intricate region of the input. Reconstruction quality improves to some degree as the GECO goal is reduced, but reconstruction quality is still mediocre even for the smallest goal value. For MONET and the largest GECO goal in Figure 4.16, one component explains the blue object as well



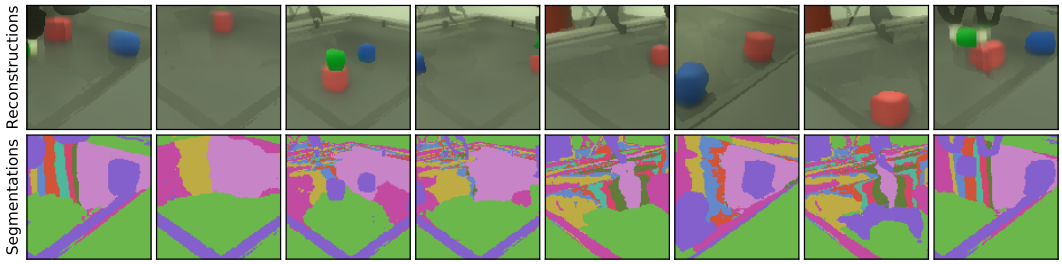
(a) Input images



(b) GECO goal = 0.5655



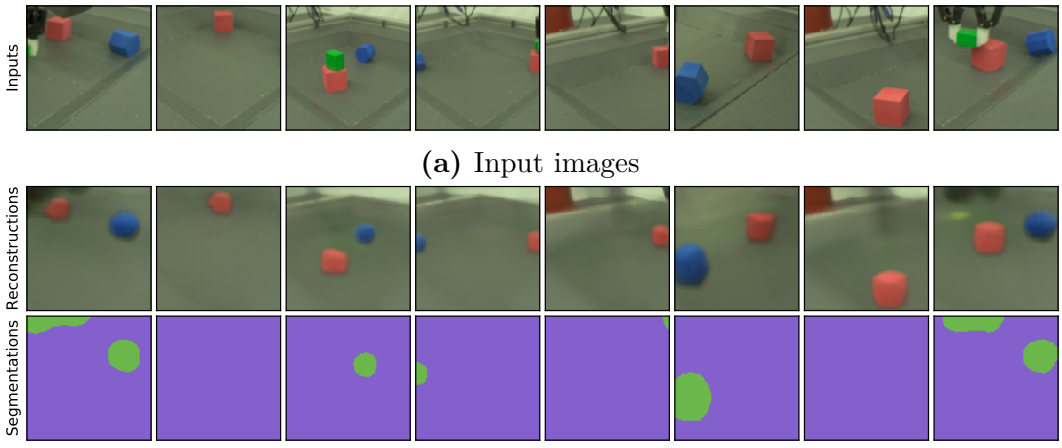
(c) GECO goal = 0.5645



(d) GECO goal = 0.5635

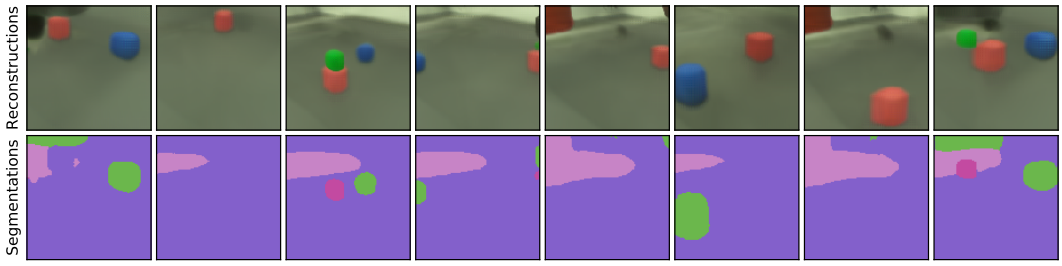
Figure 4.15: GENESIS reconstructions and segmentations after training on Sketchy. All three models struggle to learn object-like decompositions.

the robot gripper. A second component explains the rest of the image. For the medium GECO goal, two components play a similar role as for the largest goal, but a third component explains the green object and a fourth component explains a part of the background in the top half of the image. This would seem like a hopeful indication that object-like segmentations emerge as the GECO goal is further reduced. For the smallest goal value, however, the green object is segmented out and the rest of the image is segmented into two components that explain different parts of the top and bottom halves of the images. Nevertheless, from these results

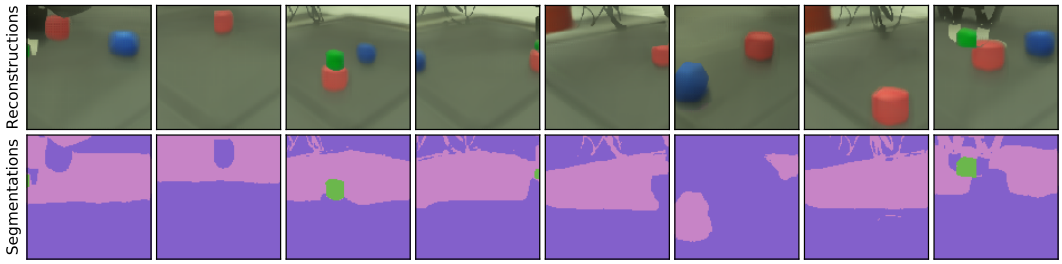


(a) Input images

(b) GECO goal = 0.5655



(c) GECO goal = 0.5645

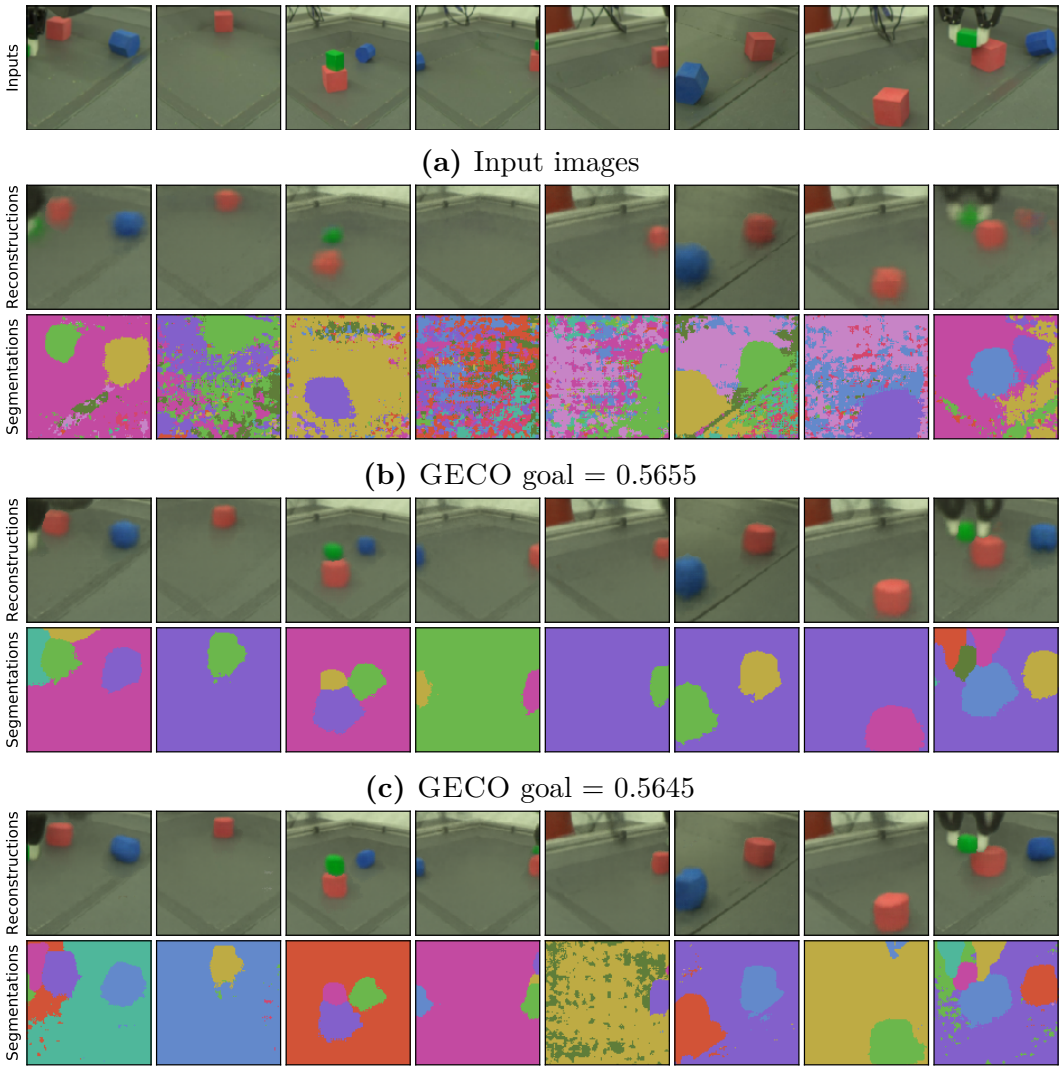


(d) GECO goal = 0.5635

Figure 4.16: MONET reconstructions and segmentations after training on Sketchy. The components are more object-like than for GENESIS, but the model is also not able to segment most objects.

it appears as if MONET might be better suited than GENESIS for learning object-like scene decompositions without supervision from real-world datasets.

For GENESIS++ in Figure 4.17, the reconstruction quality improves remarkably as the GECO goal is decreased. This is not surprising given the qualitative results in Section 4.3.3.1. For the medium goal, both the segmentations and the reconstructions are qualitatively considerably better than for GENESIS and MONET. In particular, GENESIS++ disambiguates the individual foreground



(a) Input images

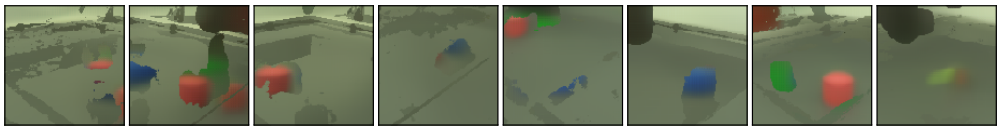
(b) GECO goal = 0.5655

(c) GECO goal = 0.5645

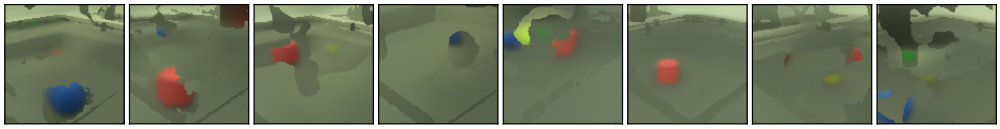
(d) GECO goal = 0.5635

Figure 4.17: GENESIS++ reconstructions and segmentations after training on Sketchy. Reconstruction and decomposition quality are quite promising. Objects are well separated for the moderate GECO goal in particular. There some unusual artefacts in the background segmentation for the smallest and the largest goal.

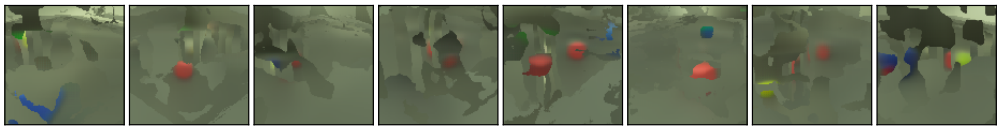
objects and the robot gripper fairly well. For example in the fourth column, GENESIS++ clearly segments out the green, red, and blue shapes and reconstructs them in separate slots. This is a promising signal that GENESIS++ is suitable for learning object-like decompositions from real-world images as encountered in robotics applications without supervision. There are some unusual artefacts in the segmentation of the background for the largest and the smallest GECO goal. It is unclear what causes these artefacts. They might potentially disappear when



(a) GECO goal = 0.5655

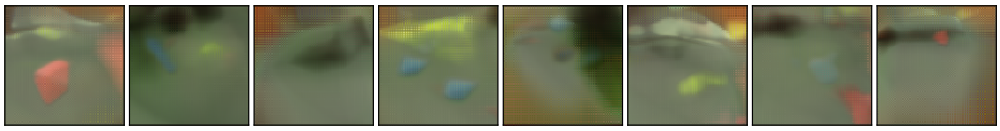


(b) GECO goal = 0.5645

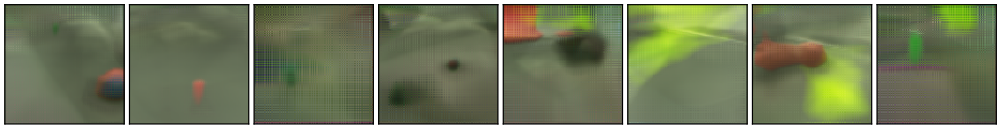


(c) GECO goal = 0.5635

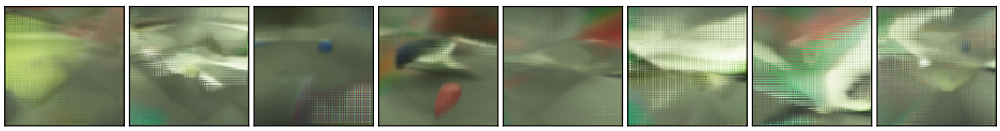
Figure 4.18: GENESIS samples after training on Sketchy. The scene structure of the training dataset is captured to some degree.



(a) GECO goal = 0.5655



(b) GECO goal = 0.5645



(c) GECO goal = 0.5635

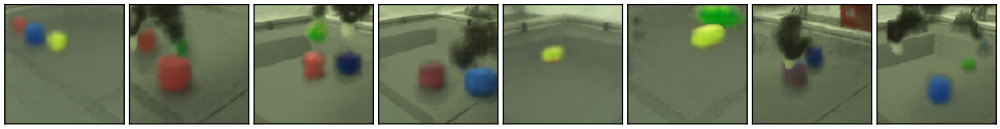
Figure 4.19: MONET samples after training on the Sketchy dataset. As in Chapter 3, MONET does not generate coherent scenes.

training the models for longer, but further investigation is required.

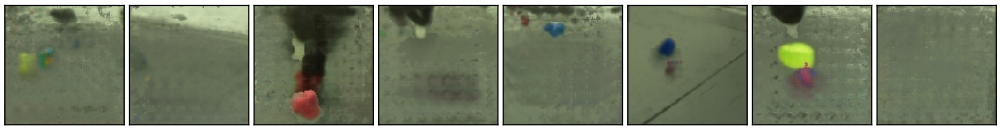
The quality of images generated by GENESIS in Figure 4.18 is mediocre. Nevertheless, they clearly show some scene structure resembling the training data: the edges of the pit are mostly distinguishable, there are between one and three foreground objects of different colours, and in some images a dark region hints at the presence of the robot gripper. For MONET in Figure 4.19, in contrast, the



(a) GECO goal = 0.5655



(b) GECO goal = 0.5645



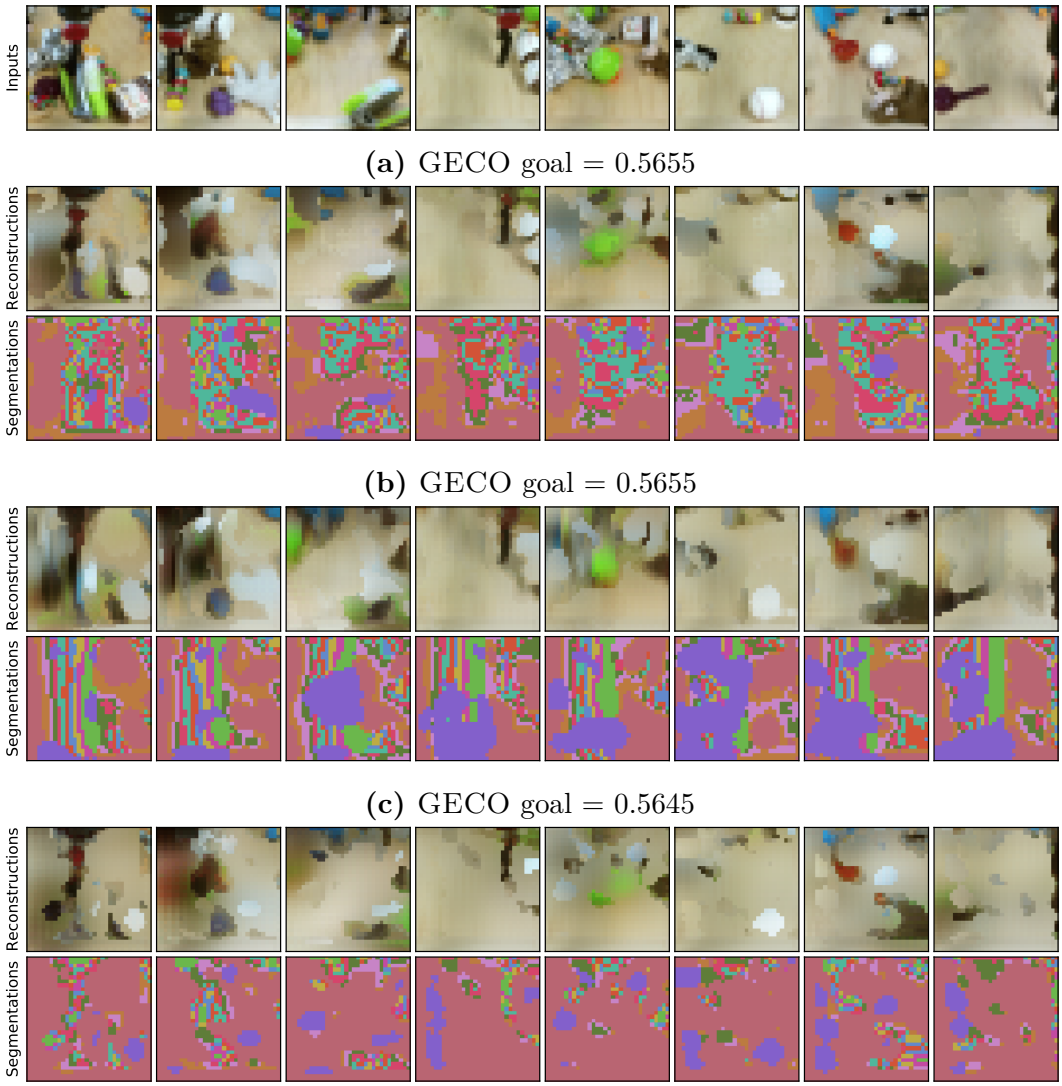
(c) GECO goal = 0.5635

Figure 4.20: GENESIS++ samples after training on the Sketchy dataset. Samples are blurry but representative of the training data for the two larger GECO goals.

generated images show considerably less structure resembling the training data. This is as expected given that the prior of MONET does not capture correlations between scene components as discussed in Chapter 3. Samples from GENESIS++ in Figure 4.20 for the two larger GECO goals clearly capture the structure of the training data. The generated images are somewhat blurry in comparison to the reconstructions in Figure 4.17 and the sample quality for the smallest GECO goal is not as good. Both of these issues can possibly be improved by making the networks larger and by training for longer. Overall, the generated images in Figure 4.20 provide a promising signal that GENESIS++ has the potential of learning object-centric world models from real-world data.

4.3.4.2 BAIR Robot Pushing

We now proceed with considering results on the BAIR Robot Pushing dataset. As described in Section 4.3.1, this dataset is considerably more challenging than Sketchy: the resolution is much lower and many scenes are very cluttered so that it is often difficult to identify objects even by eye. The pre-processing procedure described in Section 4.3.1 is designed to reduce the clutter in individual images, but the dataset is still quite challenging. Reconstructions and segmentations are shown



(a) GECO goal = 0.5655

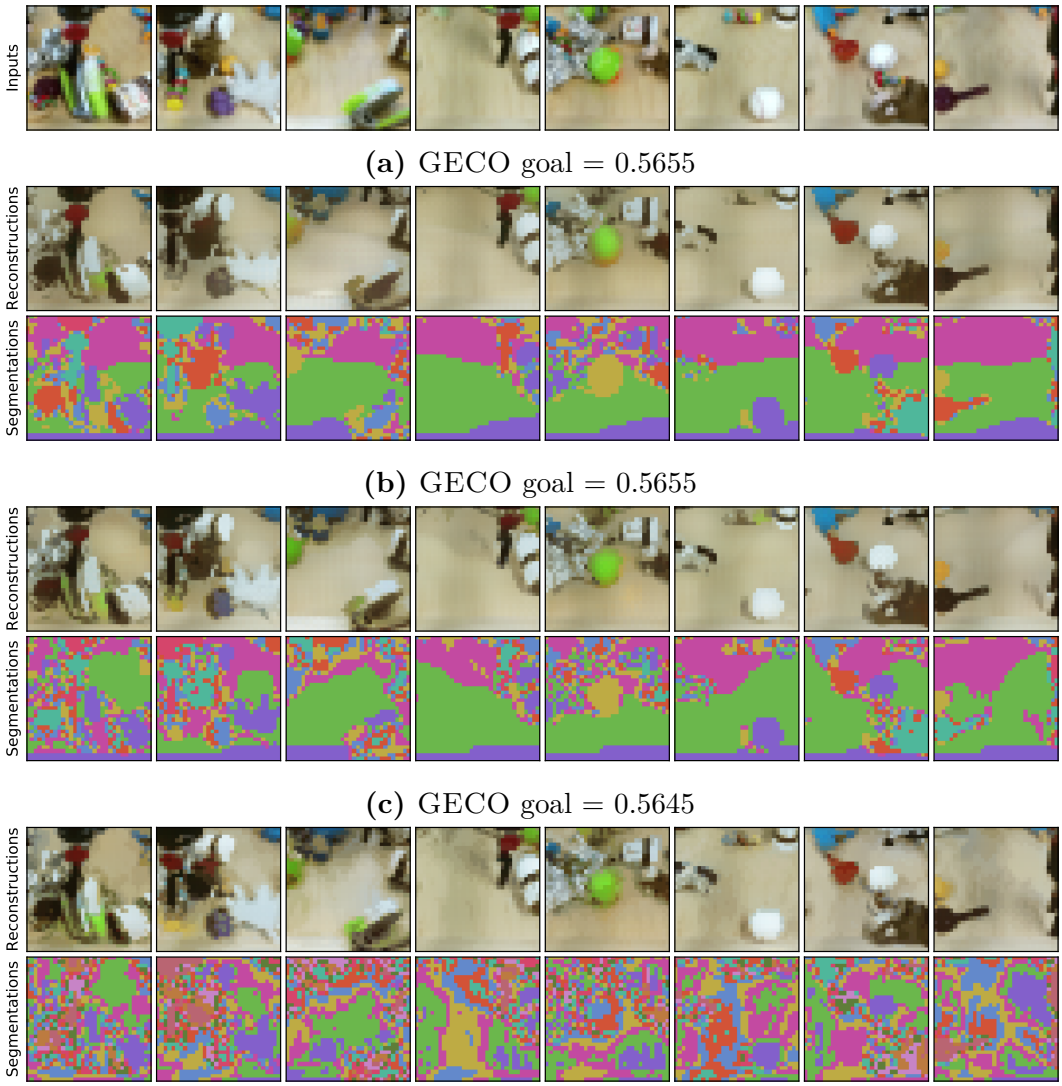
(b) GECO goal = 0.5655

(c) GECO goal = 0.5645

(d) GECO goal = 0.5635

Figure 4.21: GENESIS reconstructions and segmentations after training on BAIR. The quality of both is fairly poor.

in Figures 4.21 to 4.23 and generated images are shown in Figures 4.25 to 4.27. In Figure 4.21 the segmentations are overall not very interpretable for GENESIS. The floor tends to be divided into regions of similar shade and sometimes pixels belonging to blobs of mostly uniform colour are grouped together, for example the white ball in the sixth column. Some components are vertical stripes for the medium GECO goal, as previously also observed on Sketchy in Section 4.3.4.1. The robot end-effector is also not consistently segmented out. From the training curves which are included in Figure 4.24a, it can be seen that none of the models actually



(a) GECO goal = 0.5655

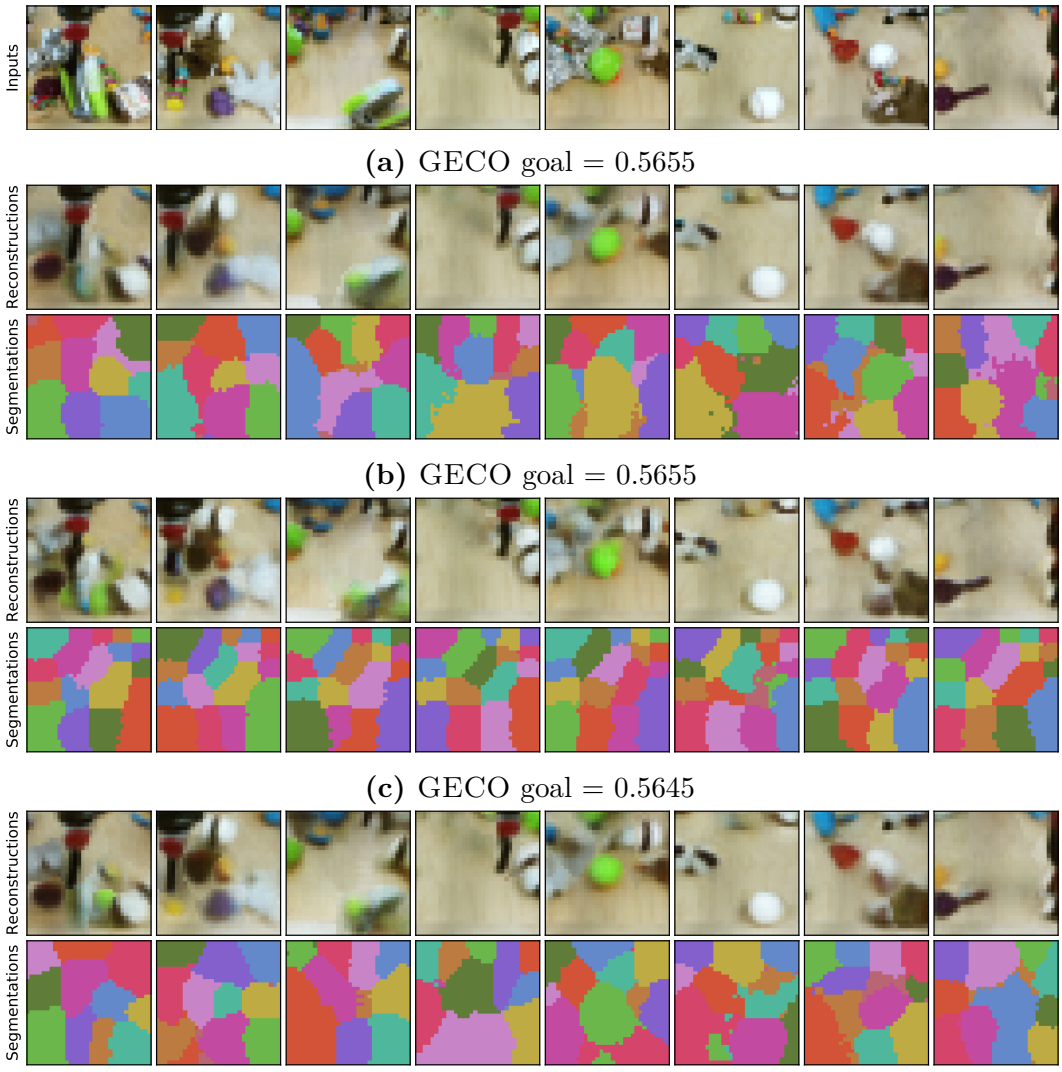
(b) GECO goal = 0.5655

(c) GECO goal = 0.5645

(d) GECO goal = 0.5635

Figure 4.22: MONET reconstructions and segmentations after training on BAIR. Segmentations are somewhat more structured than for GENESIS but they are still not very object-like.

reach even the largest GECO reconstruction error goal. This is corroborated by the observation that reconstruction quality does not improve as the goal is decreased and by the poor sample quality shown in Figure 4.25. This implies that the model is not flexible enough to properly fit the data. In Figure 4.22, the segmentations for MONET and the first two GECO goals are somewhat more interpretable and object-like than for GENESIS. The floor is consistently segmented into a top, middle, and bottom component and individual objects stand



(a) GECO goal = 0.5655

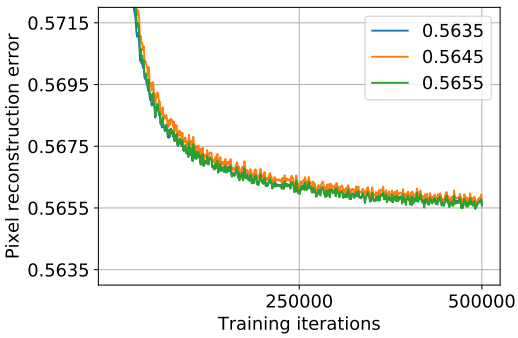
(b) GECO goal = 0.5655

(c) GECO goal = 0.5645

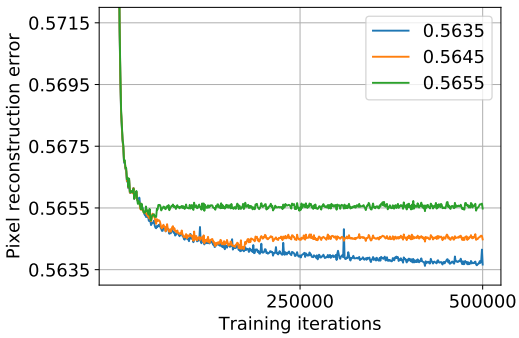
(d) GECO goal = 0.5635

Figure 4.23: GENESIS++ reconstructions and segmentations after training on BAIR. Segmentations are a regular pattern of localised patches similar to random initialisation of the initialised masks in Figure 4.2.

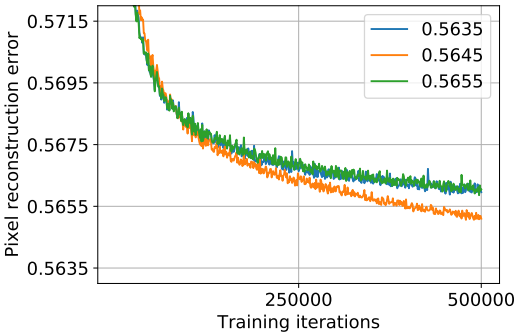
out fairly well, for example the yellow ball and the dark object just below it in the last column of the first row. For the smallest goal, however, segmentations lose interpretability. From the loss curves in Figure 4.24a, it becomes apparent that this model does not reach the GECO goal, so the beta-multiplier in the loss function (Equation (3.20)) remains small. This indicates that the KL-regularisation possibly plays an important role in achieving more interpretable segmentations. For all three models, reconstruction quality is better than for GENESIS, showing again that the MONET architecture is more flexible. GENESIS++ also struggles to learn



(a) GENESIS

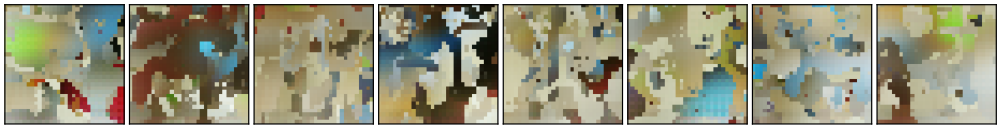


(b) MONET

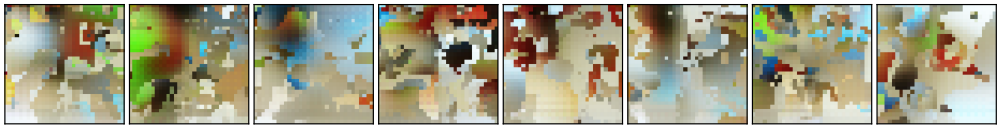


(c) GENESIS++

Figure 4.24: Moving average of the average pixel reconstruction error when training on the BAIR dataset with three different GECO goals (see legend). MONET is the only model that reliably reaches the two larger GECO goals.



(a) GECO goal = 0.5655

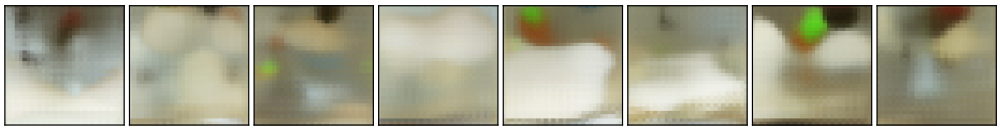


(b) GECO goal = 0.5645



(c) GECO goal = 0.5635

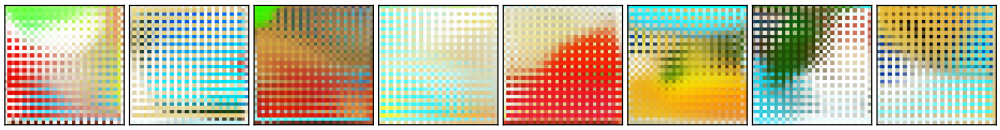
Figure 4.25: GENESIS samples after training on the BAIR dataset.



(a) GECO goal = 0.5655



(b) GECO goal = 0.5645

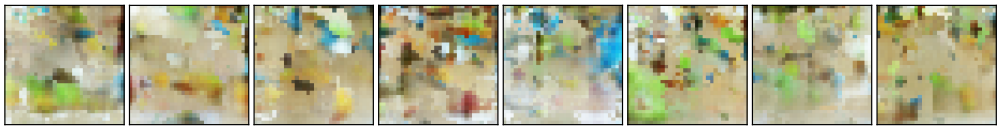


(c) GECO goal = 0.5635

Figure 4.26: MONET samples after training on the BAIR dataset.



(a) GECO goal = 0.5655



(b) GECO goal = 0.5645



(c) GECO goal = 0.5635

Figure 4.27: GENESIS++ samples after training on the BAIR dataset.

object-like segmentations for the BAIR dataset as can be seen in Figure 4.23. The segmentations correspond to a pattern of localised patches that are reminiscent of the mask initialisation from Section 4.2.2. The reconstruction error curves in Figure 4.24c show that GENESIS++ also struggles to meet even the largest GECO goal similar to GENESIS.

For all models, the generated images in Figures 4.25 to 4.27 do not show a lot of structure resembling the training data. This is not surprising given that the

GENESIS and GENESIS++ models are unable to reach the GECO goal and that is MONET generally not suitable for scene generation, as we demonstrated in Chapter 3. The samples from MONET for the smallest GECO goal stand out visually, which is most likely caused by the goal not being reached during training and the KL divergence remaining quite large.

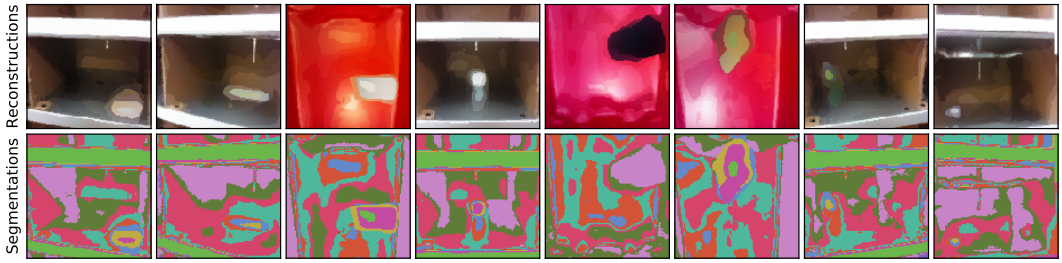
4.3.4.3 MIT-Princeton Amazon Picking Challenge 2016

Lastly, this section presents results on the on the MIT-Princeton Amazon Picking Challenge (APC) 2016 Object Segmentation dataset. Unlike the previous two datasets, images from the APAC dataset only contain a single foreground object, but the backgrounds are more realistic with each object being placed either on a shelf or in a tray as could be encountered in a typical logistics warehouse. Reconstructions and segmentations for the three models are shown in Figures 4.28 to 4.30 and several generated images are shown in Figures 4.31 to 4.33.

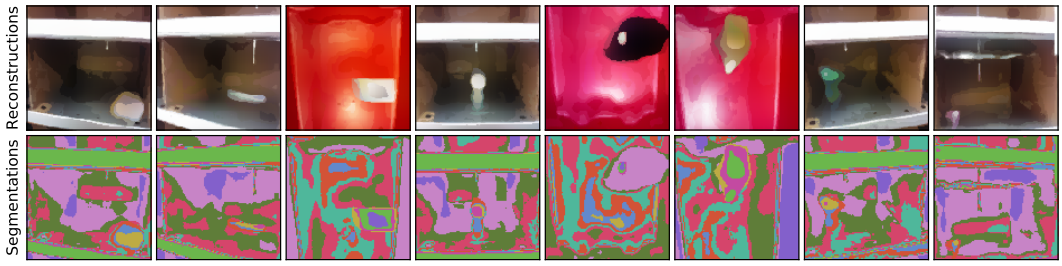
Similar to the BAIR experiments, the segmentations for GENESIS in Figure 4.32 are again quite fragmented and largely correspond to regions of uniform colour. The MONET segmentations in Figure 4.29 are comparatively interpretable for the largest GECO goal. For example, the dark glove in the fifth column and the teddy bear in the sixth column are segmented quite nicely. Segmentation quality decreases, however, as the goal is reduced. In Figure 4.30, it can be seen that GENESIS++ manages to nicely separate each foreground object from the background. The models struggle, however, to capture the fine-grained details of the foreground objects. Thus, rather than segmenting them out as a single component, each foreground object is oversegmented into several parts. The degree of oversegmentation increases as the GECO reconstruction error goal is decreased. This is not surprising given that a smaller GECO goal encourages the model to reconstruct finer visual details. If the model struggles to reconstruct a foreground object as a single component, then it is split into an increasing number of smaller components as increasingly finer details need to be reconstructed. Nevertheless, this is a considerable step forward compared to GENESIS and MONET, representing another promising



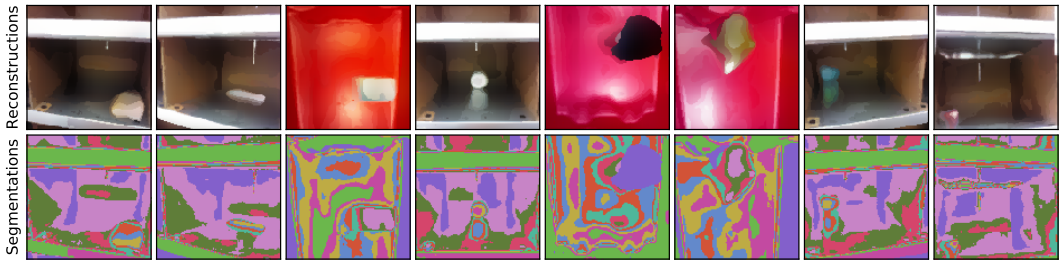
(a) Input images



(b) GECO goal = 0.5655



(c) GECO goal = 0.5645



(d) GECO goal = 0.5635

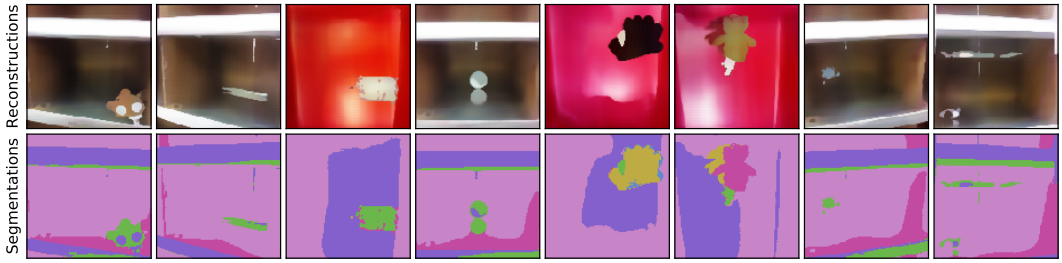
Figure 4.28: GENESIS reconstructions and segmentations after training on APC. Reconstruction quality is low and segmentations correspond to regions of uniform colour with little semantic interpretability.

indication that further incremental improvements of GENESIS++ should render it suitable for learning object-centric representations from real-world images in the context of robotics applications.

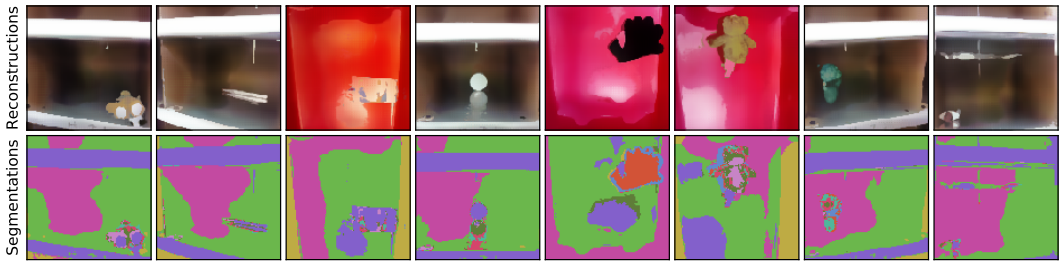
Considering the samples from GENESIS, MONET, and GENESIS++ in Figures 4.31 to 4.33, respectively, observations from the previous two datasets are corroborated. Unlike for MONET, some structure can be seen in the samples from GENESIS and GENESIS++ which facilitated by the autoregressive prior



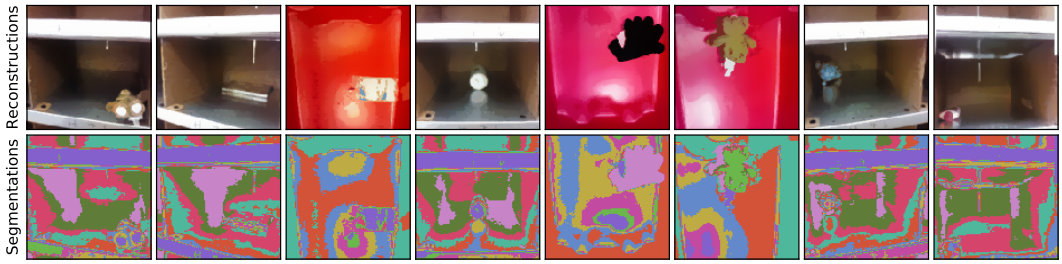
(a) Input images



(b) GECO goal = 0.5655



(c) GECO goal = 0.5645



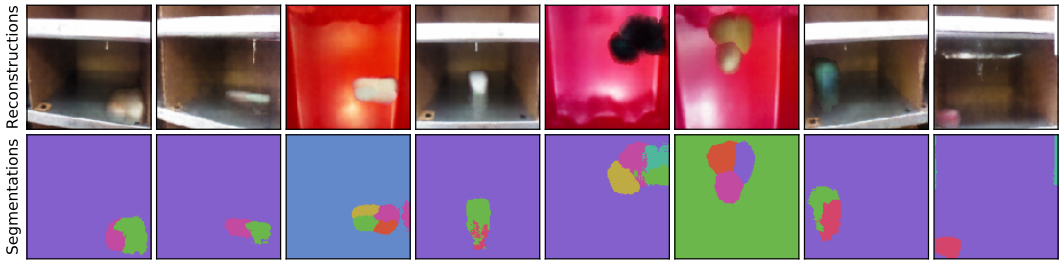
(d) GECO goal = 0.5635

Figure 4.29: MONET reconstructions and segmentations after training on APC. Reconstructions are better than for GENESIS and in some cases the foreground object is successfully segmented.

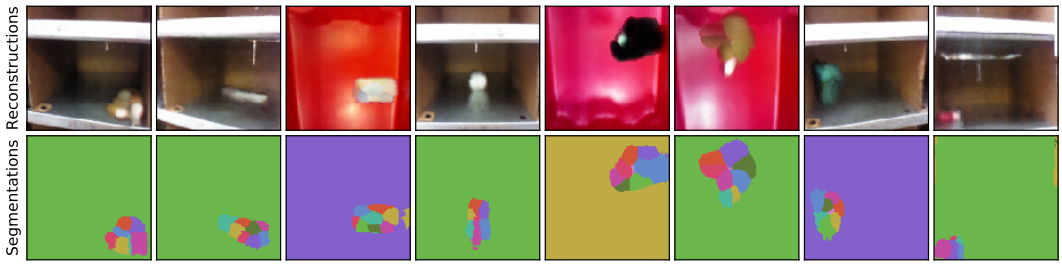
developed in Chapter 3. The visually best images are generated by GENESIS++ for the largest GECO goal of 0.5655: the model generates very crisp backgrounds as encountered in the training data, but the foreground objects are blurry and consist of multiple distinct blobs of similar colour as the reconstructions in Figure 4.30. Generation quality is not as good for the two smaller GECO goals and the model generates chequered patterns which are likely an artefact of the deconvolutional layers in the component decoder (Odena et al. [145]). This is a sign that GENE-



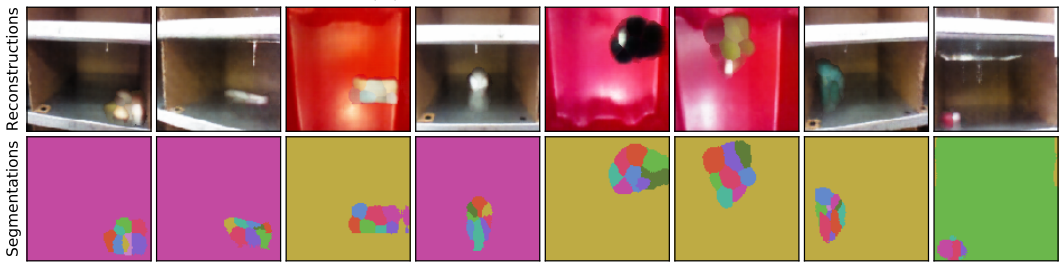
(a) Input images



(b) GECO goal = 0.5655



(c) GECO goal = 0.5645



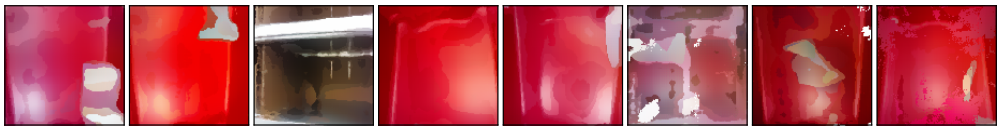
(d) GECO goal = 0.5635

Figure 4.30: GENESIS++ reconstructions and segmentations after training on APC. The foreground object is successfully separated from the background, but it is oversegmented in almost all images.

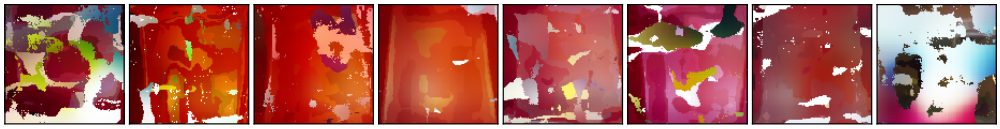
SIS++ struggles to satisfy the GECO reconstruction constraint so that the KL regularisation term in the training objective (Equation (3.20)) remains large.

4.3.4.4 Quantitative Results

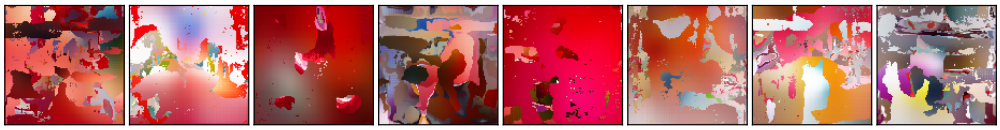
One challenge with evaluating scene decomposition performance is that ground truth labels are not available for most datasets, unless they have been carefully collected manually. For the three real-world datasets that we consider in this chapter, the APC dataset is the only dataset for which ground truth object segmentations are



(a) GECO goal = 0.5655

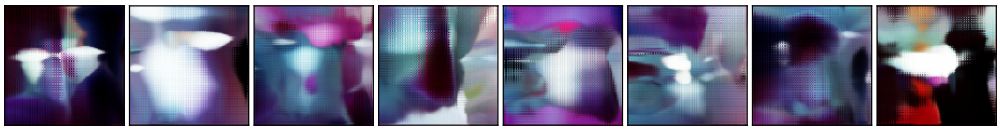


(b) GECO goal = 0.5645

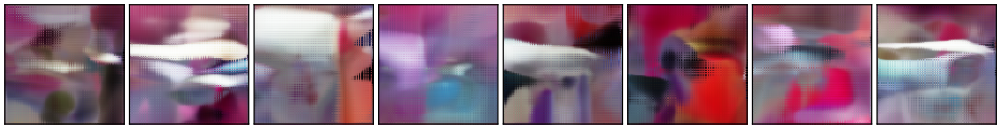


(c) GECO goal = 0.5635

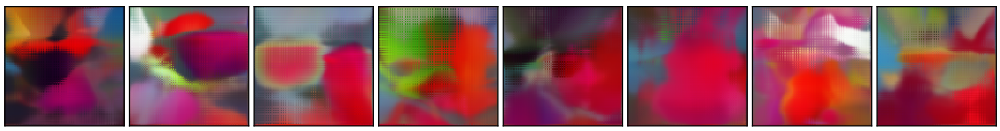
Figure 4.31: GENESIS samples after training on APC. Some scene structure from the training set is visible for the two smaller GECO goals.



(a) GECO goal = 0.5655



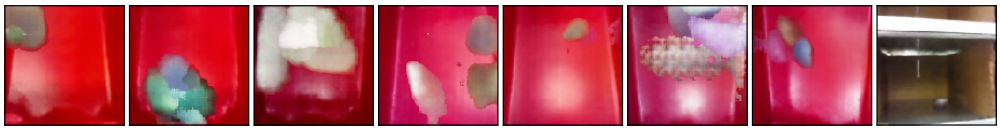
(b) GECO goal = 0.5645



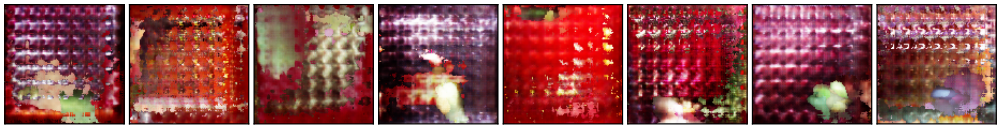
(c) GECO goal = 0.5635

Figure 4.32: MONET samples after training on APC. As before, generated images do not capture the scene structure of the training data.

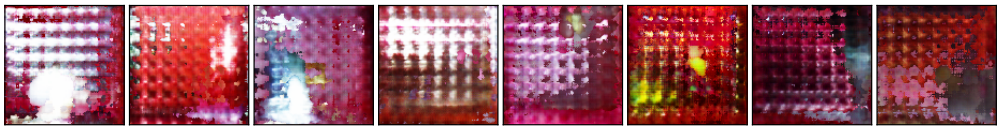
available at the time. One caveat with this is, however, that the scenes from the APC dataset only contain a single foreground object. Consequently, we report results for the ARI and MSC, computed using both the original setting described in Section 3.3.2 where only pixels belonging to the foreground objects are considered and an additional setting where the metrics are computed using the pixels from full image. For the original setting which only considers foreground pixels, the ARI



(a) GECO goal = 0.5655



(b) GECO goal = 0.5645



(c) GECO goal = 0.5635

Figure 4.33: GENESIS++ samples after training on APC. For the smallest GECO goal, the model generates a very realistic background but the foreground objects are blurry as in Figure 4.30. The images contain chequered artefacts for the two smaller GECO goals, which are possibly caused by the deconvolutional decoder (Odena et al. [145]).

Table 4.5: Scene decomposition metrics on the APC dataset.

	GECO goal	ARI	MSC	ARI FG	MSC FG
MONET	0.5655	0.11	0.48	**0.19**	**0.34**
	0.5645	0.06	0.37	0.15	0.25
	0.5635	0.04	0.27	0.10	0.26
GENESIS	0.5655	0.04	0.29	0.11	0.29
	0.5645	0.04	0.28	0.11	0.28
	0.5635	0.04	0.28	0.08	0.29
GENESIS++	0.5655	**0.54**	**0.66**	**0.19**	0.33
	0.5645	**0.54**	0.61	0.14	0.21
	0.5635	0.45	0.60	0.18	0.22

could be trivially maximised by assigning all image pixels to the same component and the MSC will be equal to the largest Intersection Over Union (IOU) of the predicted masks with the foreground objects. For the additional setting which considers all pixels, the optimal solution for both metrics would correspond to exactly one set of pixels being assigned to the foreground object and another set of pixels being assigned to the background.

Table 4.6: FID scores on the APC dataset.

	GECO goal	Sketchy	BAIR	APC
MONET	0.5655	294.1	223.6	269.3
	0.5645	294.3	245.6	299.8
	0.5635	303.1	325.4	313.7
GENESIS	0.5655	237.1	256.6	183.2
	0.5645	241.9	240.0	203.8
	0.5635	268.6	259.8	269.3
GENESIS++	0.5655	251.0	**205.8**	**158.8**
	0.5645	**174.1**	239.9	272.9
	0.5635	179.7	206.5	273.2

The results for these metrics are summarised in Table 4.5 for GENESIS++ and the two baselines for a single random seed. GENESIS++ achieves much better ARI and MSC scores than the baselines when all image pixels are considered, but it performs roughly on par with MONET when only the pixels belonging to the foreground object are taken into consideration. Generally, performance decreases as the target for the reconstruction error in GECO becomes smaller. This is corroborated by the qualitative results. The MONET segmentations in Figure 4.29 are quite interpretable in comparison for the highest GECO goal. For example, the dark glove in the fifth column and the teddy bear in the sixth column are clearly segmented out. As before in Section 4.3.3, however, the background is split into several smaller components which explains why the ARI and MSC for the full image are quite low. GENESIS++ in Figure 4.30, in contrast, nicely segments the entire background as a single component and therefore achieves much better scores on these two metrics. As the GECO goal is reduced, however, GENESIS++ increasingly oversegments the individual foreground objects. For GENESIS, the segmentations in Figure 4.28 are not very interpretable and quite fragmented, resulting in comparatively low scores on all four metrics.

While we need ground truth segmentation labels for the quantitative evaluation of scene decomposition quality, the quality of generated scenes can be compared with the FID score without any annotations as done before with in Section 4.3.3.2

and in Chapter 3, Section 3.3.6. The FID scores for the three models and the three datasets are summarised in Table 4.6. Corroborating the qualitative results in Sections 4.3.4.1 to 4.3.4.3, GENESIS++ achieves the smallest and therefore best FID score as facilitated by the autoregressive prior. Even though the reconstruction and decomposition quality of GENESIS is generally worse than for the other two models, it still manages to achieve better FID scores than MONET on the Sketchy and the APC datasets. Interestingly, despite not modelling correlations between scene components, MONET manages to achieve a better FID score than GENESIS on the BAIR dataset. This is possibly caused by the fact that the images generated by GENESIS as in Figure 4.25 contain many artefacts that are not present in the training data, whereas images generated by MONET corresponding to the best FID score in Figure 4.26 consist of blurry surfaces that have similar colours as occur in the dataset.

4.3.4.5 Discussion

We made a range of observations when running GENESIS, MONET, and GENESIS++ on the Sketchy, BAIR, and APC datasets. For all three datasets, particularly GENESIS struggles with reconstructing and decomposing the images. GENESIS to some degree segments images into regions of uniform colour so that even uniformly coloured surfaces are oversegmented due to lighting effects. MONET performs somewhat better in terms of reconstruction and decomposition than GENESIS, at least for larger GECO reconstruction error goals. Yet, on the BAIR and the APC dataset which are visually more challenging than Sketchy, the segmentations are very fragmented for the smallest GECO goal. The reconstructions and decompositions learned by GENESIS++ are much better than the two baselines on the Sketchy and APC datasets. GENESIS++ also struggles with the BAIR dataset so that segmentations collapse to localised patches that resemble the mask initialisation developed in Section 4.2.2. GENESIS++ does a particularly good job at segmenting out the multiple foreground objects for the Sketchy dataset. While reconstruction and sample quality are still mediocre, this is a hopeful sign that

better performance can be achieved using a larger architecture. The decomposition quality of GENESIS++ on the APC dataset is also promising: even if foreground objects are oversegmented into multiple parts, they are clearly separated from the background. Overall, GENESIS++ and its sub-components, in particular the IC-SBP, seem to be a promising step towards unsupervised object discovery and object-centric world modelling of real-world data from robotics applications, as highlighted by the qualitative results on the Sketchy and APC datasets in Sections 4.3.4.1 and 4.3.4.3, respectively.

Nevertheless, there is still much room for improvement in terms of both scene decomposition and generation. It would be interesting to further investigate how much the results of GENESIS++ can be improved by further tuning the model and optimisation hyperparameters. The results indicate that even the smallest GECO reconstruction error goal considered here of 0.5635 is still too large for the models to capture most of the more fine-grained visual details. Yet, even at this goal, there are also indications that GENESIS++ struggles to fit the data well, for example as illustrated by the poor generated images after training on the APC dataset in Figure 4.33. Consequently, we believe that performance can likely be considerably improved by increasing the representational capacity of the architecture and by training the models for longer, possibly with better tuned optimiser hyperparameters such as the learning rate, batch size, or learning rate warmup as discussed in Ma and Yarats [146]. Moreover, given that the Sketchy and the BAIR datasets contain sequences of images, leveraging object motion and temporal consistency as additional cues for scene decomposition should help with making the architecture larger without decompositions collapsing to a single component as explored previously in Chapter 3, Section 3.3.8.

4.4 Conclusions

This chapter explores the challenge of scaling unsupervised OCGMs to real-world datasets collected in the context of robotic manipulation tasks. To this end, we propose several innovations to overcome the limitations of GENESIS as developed

in Chapter 3: an Instance Colouring Stick-Breaking Process (IC-SBP) to facilitate differentiable clustering of instance embeddings without imposing an unnatural ordering on scene components, a principled initialisation for the scaling factor in the distance kernels of the IC-SBP based on semi-convolutional embeddings (Novotny et al. [41]), and a greedy sorting procedure to ease the optimisation of an autoregressive prior given unordered scene components. These innovations are combined in a novel model, GENESIS++, which outperforms GENESIS and our re-implementation of MONET in terms of scene decomposition and generation on the ShapeStacks [88] and ObjectsRoom [25, 103] datasets by considerable margins while also achieving better results on two more challenging real-world datasets. In the process, we find that the greedy sorting procedure for the autoregressive prior can improve the quality of generated scenes in some circumstances, but can also have a deteriorating effect on decomposition performance. We investigate the Sketchy, BAIR, and APC datasets as challenging testing grounds for learning unsupervised object-centric representations from real-world data. We find that all models struggle particularly on the BAIR dataset, which contains very cluttered images and has a comparatively low resolution. While there is still considerable room for improvement, GENESIS++ achieves better results than GENESIS and our re-implementation of MONET. Importantly, GENESIS++ learns scene decompositions that separate the individual foreground objects reasonably well from the background. Better results can very likely be achieved by further tuning various architectural details and hyperparameter choices.

However, there is also the question of "how good is good enough" in terms of both scene decomposition performance and scene generation performance. Ultimately, this depends on the downstream application that the models are used for. Most applications likely do not require pixel perfect segmentations for reliable task execution. Similarly, part of the appeal of leaning to solve tasks in generative "world models" (Ha and Schmidhuber [29]) is that learning takes place in a low-dimensional latent space rather than in pixel space. It is consequently not necessary to have a generator that is capable of generating pixel-perfect images, as long is the structure

of the real-world tasks is captured in the low-dimensional encoding. While ARI, MSC, and FID are useful metrics for developing a sense of the model properties, they are merely a substitute and it is necessary to close the loop of perception and planning to fully assess the suitability of these models for real-world applications.

4.4.1 Limitations

GENESIS++ overcomes some of the limitations of GENESIS and takes a step towards learning object representations without labels from real-world datasets. To demonstrate this, we consider three datasets collected in the context of robotics applications. Nevertheless, GENESIS++ still leaves much room for improvement both in terms of the results achieved on these datasets as well as the complexity of the datasets themselves. For example, while the reconstructions of the input images are reasonable, they are still fairly blurry and finer visual details are lost. Moreover, the datasets only contain images in well-constrained settings as opposed to images collected more open domains such as autonomous driving, for example.

4.4.2 Future Work

Interestingly, GENESIS++ as presented in Chapter 4 exhibits somewhat different qualitative behaviour compared to its predecessors MONET and GENESIS. In particular, GENESIS++ is able to produce reconstructions and samples with better visual detail on the ShapeStacks and ObjectsRoom datasets. In contrast to GENESIS and despite its more expressive latent encoder and decoder, GENESIS++ has not yet been observed to collapse to a single component during training. We thus conjecture that the symmetric nature of the IC-SBP and the implicit induction of mask locality by the semi-convolutional embeddings are very effective means for inducing object-like scene decompositions. It would be interesting to observe how far reconstruction, segmentation, and sample quality could be improved by making the architecture increasingly more expressive. In it's current from, GENESIS++ still uses vector-valued latents for encoding scene components. This makes capturing fine-grained textures quite difficult, which could possibly addressed by using a hierarchical latent-variable formulation (e.g. Maaløe

et al. [147] and Vahdat and Kautz [148]). There are also other opportunities, though, to improve the model. For example, Kong and Fowlkes [134] use mean shift clustering to group instance embeddings, arguing that this emphasises the learning signal from pixels at object boundaries. In Figure 4.10, it can be seen that the model leaves a gap between attention masks of adjacent components. This might be remedied with the mean shift operation from Kong and Fowlkes [134], which potentially also helps with inducing better object decomposition. Moreover, the greedy sorting of posterior latents might not be the right long-term strategy for addressing the challenge of learning a prior over a set of objects. Instead, it might be better to increase the flexibility of the prior network or to use more flexible distributions rather than diagonal Gaussians to better model the distribution over scene components.

It could potentially be worthwhile to collect instance segmentation labels for the foreground objects and the robot arm in the Sketchy dataset to perform a more elaborate quantitative evaluation and comparison of different approaches. Recently, Nematollahi et al. [149] also released a robot interaction dataset that shows scenes that are qualitatively similar to Sketchy, but with a larger variety of more complex foreground objects. This could be useful additional dataset for evaluating GENESIS++ and other OCGMs. Finally, a full characterisation of the suitability of different OCGMs for robotics applications requires closing the loop between perception and planning. This could either be examined using object-centric world models (Ha and Schmidhuber [29]) to learn skills in simulation or by utilising OCGMs for model-based RL, for example as in Watters et al. [150]. Moreover, while unsupervised object understanding is a useful capability, it is also quite a challenging and somewhat under-constrained problem. Learning object representations jointly with a planning model would allow information from the task to be used for identifying useful abstractions. Similarly, it might be interesting to investigate how a small number labelled examples or some form of weak supervision might be best utilised to guide object discovery.

5
Conclusions

In this thesis, we explore the subject of computation and supervision efficient object detection and discovery for real-world robotics applications. Object perception is a critical component in robotic systems, whether this is in the context of autonomous driving (Chapter 2) or in robotic manipulation (Chapters 3 and 4). In particular, we focus on the unique challenges and opportunities in robotics. This includes the availability of diverse sensing modalities such a lidar devices which directly provide 3D spatial information (Chapter 2). Care needs to be taken, however, with regards to computational efficiency when handling such data streams, considering the physical hardware restrictions of robotic platforms. Moreover, flexible and versatile robots need to be able to learn from data in a supervision-efficient manner in order to handle novel situations and to facilitate flexible, fast skill acquisition (Chapters 3 and 4).

Chapter 2 explores means for computationally efficient object detection in 3D lidar point clouds. In particular, we observe that prior works that utilise dense 3D convolutions are far too slow for processing full lidar point clouds in practical applications (Maturana and Scherer [18]) and that projecting point clouds into a 2D representations as in Li et al. [19] leads to a less general solution that discards the inherent three-dimensional structure of the data. Consequently, we introduce Vote3Deep to perform effective object detection in lidar point clouds natively in 3D. We show that we can overcome the computational burden introduced by the third

spatial dimension by exploiting the natural spatial sparsity inherent in 3D point clouds and by leveraging a computationally efficient sparse convolution operation from Wang and Posner [20]. Crucially, the spatial sparsity of the input data is maintained throughout the entire Convolutional Neural Network (CNN) feature hierarchy through the use of Rectified Linear Unit (RELU) non-linearities and by constraining filter biases to be non-positive. Moreover, in order to explicitly tune the trade off between detection accuracy and speed, we show that an $\mathcal{L}_1$ sparsity regulariser can be used to increase the sparsity in intermediate representations to further improve detection speed at minimal drop in prediction accuracy. Moreover, we investigate the sparsity of the hierarchical feature maps both qualitatively and quantitatively, where we observe how the $\mathcal{L}_1$ sparsity loss increases spatial sparsity while still maintaining the spatial structure of the input point cloud in the intermediate representations. We show that these innovations enable the training of comparatively small CNNs that achieve high prediction accuracy even when the number of available training examples is relatively small. This leads to significant advancements in comparison to prior art in terms of detection performance on the KITTI Vision Benchmark Suite (Geiger et al. [12]), while remaining competitive in terms of detection speed. The main limitations of this work are the use hand-crafted input features and fixed bounding box dimensions. We discuss different avenues for overcoming these limitations by making use of recently developed software libraries for GPU-accelerated sparse convolutions (Graham et al. [46], Ren et al. [75], Choy et al. [76], Hackel et al. [77], and Tang et al. [78]).

The paradigm of supervised object detection as explored in Chapter 2 is powerful in constrained applications where a large number of annotated training examples are available. Nevertheless, we observe that it is fundamentally infeasible to manually collect data for all possible object categories, motiving the need for models that are able to identify object-like entities in an unsupervised fashion. In Chapter 3, we frame this as learning a probabilistic, generative model of the data distribution, whereby the generative model is structured to capture the compositional, object-based nature of visual scenes. Moreover, we discuss the possibility of utilising

Object-Centric Generative Models (OCGMs) for efficient skill acquisition in "world models" (Ha and Schmidhuber [29]). Based on these motivations, we develop a novel OCGM called Generative Scene Inference and Sampling (GENESIS), which learns to decompose rendered images of simulated 3D environments and to generate entire, structurally coherent scenes in an object-centric fashion. This is achieved without using any form of labelled supervision. In particular, we propose an autoregressive prior to model correlations between scene components in the generative model. To better characterise the properties of GENESIS, we also perform a set of semi-supervised experiments and we conduct an empirical investigation of "reconstruction bottlenecks". The former corroborates that the representations learned by GENESIS are well suited for tasks that require reasoning about the objects in a scene. In the latter, we observe an intricate interplay between reconstruction and segmentation: if the reconstruction bottleneck is too wide, then segmentations collapse to a single component; if it is too narrow, then reconstruction and segmentation fidelity deteriorate. Even though we demonstrate that GENESIS extends prior art from Burgess et al. [25] and Greff et al. [26] by offering the ability to generate scenes in an object-centric fashion, its main limitation is that it is only able to learn object-centric representations from simulated datasets with limited visual complexity.

Ultimately, we want to perform unsupervised object discovery and learn object-centric world models for real-world robotics applications. To this end, we propose several innovations in Chapter 4 to overcome the limitations of GENESIS which culminate in an improved model called GENESIS++. Inspired by the works in Greff et al. [26] and Locatello et al. [36] and in contrast to GENESIS, GENESIS++ does not impose an unnatural ordering on scene components and instead infers object representations in a symmetric fashion. We motivate this approach in terms of computational benefits as it avoids the utilisation of Recurrent Neural Networks (RNNs) (Novotny et al. [41]) and we also posit that symmetric representations are less likely to collapse to a single component during training as explored in Chapter 3. Inspired by recent literature on instance colouring methods, this is

achieved with a novel fully-differentiable, soft clustering mechanism for instance embeddings, which we refer to as the Instance Colouring Stick-Breaking Process (IC-SBP). Furthermore, we discuss that it might be potentially challenging to learn an autoregressive prior to model correlations between scene components as in Chapter 3 when object representations are unordered. As an alternative, we explore a greedy sorting procedure of the posterior latents so that the prior can learn a fixed, sequential generation strategy. We evaluate GENESIS++ empirically on two simulated datasets and three challenging real-world datasets that have not been studied extensively before in the context of OCGMs: Sketchy (Cabi et al. [139]), BAIR Robot Pushing (Ebert et al. [140]), and the MIT-Princeton APC dataset 2016 [141]. On the simulated datasets, GENESIS++ outperforms GENESIS and our re-implementation of Multi-Object Network (MONET) (Burgess et al. [25]) by considerable margins in terms of both scene decomposition and generation. Similarly, we also observe that GENESIS++ learns more interpretable and object-like decompositions on two of the three real-world datasets. The BAIR dataset proves to be an exception to this, but all three models struggle with this dataset given its very low resolution and cluttered scenes. It might therefore not be a suitable benchmark dataset and future research might rather focus on the Sketchy and APC datasets. While the results of GENESIS++ are promising, there is still much work to be done in terms of learning object-centric representations from real-world datasets. Yet, the results indicate that further improvements can likely be achieved with GENESIS++ with better architecture details and training hyperparameters. Moreover, it is unclear how good scene decomposition and generation performance actually need to be for the purpose of practical robotics applications. Answering this question will likely require testing the direct utilisation of OCGMs such as GENESIS++ in actual applications, a task left for future work.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton. "Deep Learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten ZIP Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551.

[3] R. Collobert, K. Kavukcuoglu, and C. Farabet. "Torch7: A Matlab-like Environment for Machine Learning". In: *NeurIPS BigLearn Workshop* (2011).

[4] A. Vedaldi and K. Lenc. "MatConvNet: Convolutional Neural Networks for MATLAB". In: *ACM International Conference on Multimedia* (2015).

[5] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, et al. "Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv preprints arXiv:1605.02688* (2016).

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. "Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic Differentiation in PyTorch". In: (2017).

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision (IJCV)* 88.2 (2010), pp. 303–338.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision (ECCV)* (2014).

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[12] A. Geiger, P. Lenz, and R. Urtasun. "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[13] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. "1 Year, 1000 km: The Oxford RobotCar Dataset". In: *International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15.

[14] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner. "The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2020).

[15] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2015).

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. "SSD: Single Shot MultiBox Detector". In: *arXiv preprint arXiv:1512.02325* (2015).

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[18] D. Maturana and S. Scherer. "3D Convolutional Neural Networks for Landing Zone Detection from LIDAR". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2015).

[19] B. Li, T. Zhang, and T. Xia. "Vehicle Detection from 3D Lidar Using Fully Convolutional Network". In: *arXiv preprint arXiv:1608.07916* (2016).

[20] D. Z. Wang and I. Posner. "Voting for Voting in Online Point Cloud Object Detection". In: *Robotics: Science and Systems (RSS)* (2015).

[21] C. Cortes and V. Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

[22] J. Huang and K. Murphy. "Efficient Inference in Occlusion-Aware Generative models of Images". In: *arXiv preprint arXiv:1511.06362* (2015).

[23] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2016).

[24] A. R. Kosiorek, H. Kim, Y. W. Teh, and I. Posner. "Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[25] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. "MONet: Unsupervised Scene Decomposition and Representation". In: *arXiv preprint arXiv:1901.11390* (2019).

[26] K. Greff, R. L. Kaufmann, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. "Multi-Object Representation Learning with Iterative Variational Inference". In: *International Conference on Machine Learning (ICML)* (2019).

[27] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *International Conference On Learning Representations (ICLR)* (2014).

[28] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *International Conference on Machine Learning (ICML)* (2014).

[29] D. Ha and J. Schmidhuber. "World Models". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[30] S. van Steenkiste, K. Kurach, and S. Gelly. "A Case for Object Compositionality in Deep Generative Models of Images". In: *NeurIPS Workshop on Modeling the Physical World: Learning, Perception, and Control* (2018).

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2014).

[32] Z. Lin, Y.-F. Wu, S. V. Peri, W. Sun, G. Singh, F. Deng, J. Jiang, and S. Ahn. "SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition". In: *International Conference on Learning Representations (ICLR)* (2020).

[33] J. Jiang, S. Janghorbani, G. De Melo, and S. Ahn. "SCALOR: Generative World Models with Scalable Object Representations". In: *International Conference on Learning Representations (ICLR)* (2020).

[34] Y. Yang, Y. Chen, and S. Soatto. "Learning to Manipulate Individual Objects in an Image". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[35] D. M. Bear, C. Fan, D. Mrowca, Y. Li, S. Alter, A. Nayebi, J. Schwartz, L. Fei-Fei, J. Wu, J. B. Tenenbaum, et al. "Learning Physical Graph Representations from Visual Scenes". In: *arXiv preprint arXiv:2006.12373* (2020).

[36] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. "Object-Centric Learning with Slot Attention". In: *arXiv preprint arXiv:2006.15055* (2020).

[37] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra. "BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images". In: *arXiv preprint arXiv:2002.08988* (2020).

[38] S. Ehrhardt, O. Groth, A. Monszpart, M. Engelcke, I. Posner, N. Mitra, and A. Vedaldi. "RELATE: Physically Plausible Multi-Object Scene Synthesis Using Structured Latent Spaces". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

[39] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[40] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2011).

[41] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi. "Semi-Convolutional Operators for Instance Segmentation". In: *European Conference on Computer Vision (ECCV)* (2018).

[42] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. "Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2017).

[43] M. Engelcke, A. R. Kosiorek, O. Parker Jones, and I. Posner. "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations". In: *International Conference on Learning Representations (ICLR)* (2020).

[44] M. Engelcke, A. R. Kosiorek, O. Parker Jones, and I. Posner. "Reconstruction Bottlenecks in Object-Centric Generative Models". In: *ICML Workshop on Object-Oriented Learning* (2020).

[45] L. Yi, L. Shao, M. Savva, H. Huang, Y. Zhou, Q. Wang, B. Graham, M. Engelcke, R. Klokov, V. Lempitsky, et al. "Large-Scale 3D Shape Reconstruction and Segmentation from ShapeNet Core55". In: *arXiv preprint arXiv:1710.06104* (2017).

[46] B. Graham, M. Engelcke, and L. Van Der Maaten. "3D Semantic Segmentation with Submanifold Sparse Convolutional Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[47] E. Wagstaff, F. B. Fuchs, M. Engelcke, I. Posner, and M. Osborne. "On the Limitations of Representing Functions on Sets". In: *International Conference on Machine Learning (ICML)* (2019).

[48] A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, I. Havoutis, and I. Posner. "First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020).

[49] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. *A Neural Network and Method of Using a Neural Network to Detect Objects in an Environment*. EP Patent App. 20170777642. 2019.

[50] J. Behley, V. Steinhage, and A. B. Cremers. "Laser-Based Segment Classification using a Mixture of Bag-of-Words". In: *IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)* (2013).

[51] D. Prokhorov. "A Convolutional Learning System for Object Classification in 3-D Lidar Data". In: *IEEE Transactions on Neural Networks* (2010).

[52] D. Maturana and S. Scherer. "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015).

[53] Z. Wu, S. Song, A. Khosla, F. Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. "3D ShapeNets: A Deep Representation for Volumetric Shapes". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[54] B. Graham. "Spatially-Sparse Convolutional Neural Networks". In: *arXiv Preprint arXiv:1409.6070* (2014).

[55] B. Graham. "Sparse 3D convolutional neural networks". In: *arXiv preprint arXiv:1505.02890* (2015).

[56] V. Jampani, M. Kiefel, and P. V. Gehler. "Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[57] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy. "Sparse Convolutional Neural Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[58] J. Park, S. Li, W. Wen, H. Li, Y. Chen, and P. Dubey. "Holistic SparseCNN: Forging the Trident of Accuracy, Speed, and Size". In: *arXiv preprint arXiv:1608.01409* (2016).

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances In Neural Information Processing Systems (NeurIPS)* (2012).

[60] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference On Learning Representations (ICLR)* (2015).

[61] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with Convolutions". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[62] J. Long, E. Shelhamer, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[63] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *International Conference On Learning Representations (ICLR)* (2016).

[64] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum Learning". In: *International Conference on Machine Learning (ICML)* (2009).

[65] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *arXiv preprint arXiv:1502.01852* (2015).

[66] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[67] L. Plotkin. “PyDriver: Entwicklung eines Frameworks für Räumliche Detektion und Klassifikation von Objekten in Fahrzeugumgebung”. Bachelor’s Thesis (Studienarbeit). Germany: Karlsruhe Institute of Technology, 2015.

[68] A. Gonzalez, G. Villalonga, J. Xu, D. Vazquez, J. Amores, and A. M. Lopez. “Multiview Random Forest of Local Experts Combining RGB and LIDAR Data for Pedestrian Detection”. In: *IEEE Intelligent Vehicles Symposium (IV)* (2015).

[69] C. Premebida, J. Carreira, J. Batista, and U. Nunes. “Pedestrian Detection Combining RGB and Dense LIDAR Data”. In: *IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)* (2014).

[70] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. “Deep Sets”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[71] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *arXiv preprint arXiv:1612.00593* (2017).

[72] R. Klokov and V. Lempitsky. “Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)* (2017).

[73] G. Riegler, A. Osman Ulusoy, and A. Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[74] Y. Yan, Y. Mao, and B. Li. “SECOND: Sparsely Embedded Convolutional Detection”. In: *Sensors* 18.10 (2018), p. 3337.

[75] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun. “SBNet: Sparse Blocks Network for Fast Inference”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[76] C. Choy, J. Gwak, and S. Savarese. “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[77] T. Hackel, M. Usvyatsov, S. Galliani, J. D. Wegner, and K. Schindler. “Inference, Learning and Attention Mechanisms that Exploit and Preserve Sparsity in CNNs”. In: *International Journal of Computer Vision* (2020).

[78] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. “Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution”. In: *European Conference on Computer Vision (ECCV)* (2020).

[79] L. Han, T. Zheng, L. Xu, and L. Fang. “OccuSeg: Occupancy-aware 3D Instance Segmentation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[80] C. R. Qi, O. Litany, K. He, and L. J. Guibas. “Deep Hough Voting for 3D Object Detection in Point Clouds”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)* (2019).

[81] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. "One-Shot Imitation From Observing Humans via Domain-Adaptive Meta-Learning". In: *arXiv preprint arXiv:1802.01557* (2018).

[82] K. Greff, A. Rasmus, M. Berglund, T. Hao, H. Valpola, and J. Schmidhuber. "Tagger: Deep Unsupervised Perceptual Grouping". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2016).

[83] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2 (1999), pp. 183–233.

[84] K. Greff, S. van Steenkiste, and J. Schmidhuber. "Neural Expectation Maximization". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[85] S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber. "Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions". In: *arXiv preprint arXiv:1802.10353* (2018).

[86] Y. LeCun, C. Cortes, and C. Burges. "MNIST Handwritten Digit Database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[87] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. "Neural Scene Representation and Rendering". In: *Science* 360.6394 (2018), pp. 1204–1210.

[88] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. "ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking". In: *European Conference on Computer Vision (ECCV)* (2018).

[89] M. Jaderberg, K. Simonyan, A. Zisserman, et al. "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2015).

[90] K. Xu, C. Li, J. Zhu, and B. Zhang. "Multi-Objects Generation with Amortized Structural Regularization". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2018).

[91] K. Stelzner, R. Peharz, and K. Kersting. "Faster Attend-Infer-Repeat with Tractable Probabilistic Models". In: *International Conference on Machine Learning (ICML)* (2019).

[92] E. Crawford and J. Pineau. "Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks". In: *AAAI Conference on Artificial Intelligence* (2019).

[93] A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton. "Stacked Capsule Autoencoders". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[94] S. Sabour, N. Frosst, and G. E. Hinton. "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[95] M. Chen, T. Artières, and L. Denoyer. "Unsupervised Object Segmentation by Redrawing". In: *arXiv preprint arXiv:1905.13539* (2019).

[96] A. Bielski and P. Favaro. "Emergence of Object Segmentation in Perturbed Generative Models". In: *arXiv preprint arXiv:1905.12663* (2019).

[97] R. Arandjelović and A. Zisserman. "Object Discovery with a Copy-Pasting GAN". In: *arXiv preprint arXiv:1905.11369* (2019).

[98] S. Azadi, D. Pathak, S. Ebrahimi, and T. Darrell. "Compositional GAN: Learning Image-Conditional Binary Composition". In: *arXiv preprint arXiv:1807.07560* (2019).

[99] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[100] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations (ICLR)* (2017).

[101] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. "How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks". In: *International Conference on Machine Learning (ICML)* (2016).

[102] D. J. Rezende and F. Viola. "Taming VAEs". In: *arXiv preprint arXiv:1810.00597* (2018).

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

[104] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. *dSprites: Disentanglement Testing Sprites Dataset*. 2017. URL: `https://github.com/deepmind/dsprites-dataset/`.

[105] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *IEEE/CVF International Conference on Computer Vision (ICCV)* (2017).

[106] L. Hubert and P. Arabie. "Comparing Partitions". In: *Journal of Classification* 2.1 (1985), pp. 193–218.

[107] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour Detection and Hierarchical Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010).

[108] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[109] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[110] R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. "Sylvester Normalizing Flows for Variational Inference". In: *Conference on Uncertainty in Artificial Intelligence (UAI)* (2018).

[111] D. J. Rezende and S. Mohamed. "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning (ICML)* (2015).

[112] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. "Language Modeling with Gated Convolutional Networks". In: *International Conference on Machine Learning (ICML)* (2017).

[113] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *International Conference on Machine Learning (ICML)* (2015).

[114] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[115] N. Watters, L. Matthey, C. P. Burgess, and A. Lerchner. "Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs". In: *arXiv preprint arXiv:1901.07017* (2019).

[116] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015).

[117] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference On Learning Representations (ICLR)* (2015).

[118] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *AAAI Conference on Artificial Intelligence* (2017).

[119] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. "Panoptic Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[120] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. "Path Aggregation Network for Instance Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[121] B. Romera-Paredes and P. H. S. Torr. "Recurrent Instance Segmentation". In: *European Conference on Computer Vision (ECCV)* (2016).

[122] M. Ren and R. S. Zemel. "End-to-End Instance Segmentation with Recurrent Attention". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[123] B. De Brabandere, D. Neven, and L. Van Gool. "Semantic Instance Segmentation with a Discriminative Loss Function". In: *arXiv preprint arXiv:1708.02551* (2017).

[124] M. Bai and R. Urtasun. "Deep Watershed Transform for Instance Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[125] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. "Semantic Instance Segmentation via Deep Metric Learning". In: *arXiv preprint arXiv:1703.10277* (2017).

[126] S. H. Rezatofighi, R. Kaskman, F. T. Motlagh, Q. Shi, D. Cremers, L. Leal-Taixé, and I. Reid. "Deep Perm-Set Net: Learn to Predict Sets with Unknown Permutation and Cardinality using Deep Neural Networks". In: *arXiv preprint arXiv:1805.00613* (2018).

[127] Y. Zhang, J. Hare, and A. Prugel-Bennett. "Deep Set Prediction Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019), pp. 3212–3222.

[128] A. R. Kosiorek, H. Kim, and D. J. Rezende. "Conditional Set Generation with Transformers". In: *arXiv preprint arXiv:2006.16841* (2020).

[129] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-End Object Detection with Transformers". In: *arXiv preprint arXiv:2005.12872* (2020).

[130] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Sysems (NeurIPS)* (2017).

[131] A. Salvador, M. Bellver, V. Campos, M. Baradad, F. Marques, J. Torres, and X. Giro-i-Nieto. "Recurrent Neural Networks for Semantic Instance Segmentation". In: *arXiv preprint arXiv:1712.00617* (2017).

[132] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. "InstanceCut: From Edges to Instances with MultiCut". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[133] V. Kulikov, V. Yurchenko, and V. Lempitsky. "Instance Segmentation by Deep Coloring". In: *arXiv preprint arXiv:1807.10007* (2018).

[134] S. Kong and C. C. Fowlkes. "Recurrent Pixel Embedding for Instance Grouping". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[135] D. B. Dunson and J.-H. Park. "Kernel Stick-Breaking Processes". In: *Biometrika* 95.2 (2008), pp. 307–323.

[136] Q. An, C. Wang, I. Shterev, E. Wang, L. Carin, and D. B. Dunson. "Hierarchical Kernel Stick-Breaking Process for Multi-Task Image Analysis". In: *International Conference on Machine Learning (ICML)* (2008).

[137] P. Orbanz and J. M. Buhmann. "Nonparametric Bayesian Image Segmentation". In: *International Journal of Computer Vision (IJCV)* 77.1-3 (2008), pp. 25–45.

[138] L. Ren, L. Du, L. Carin, and D. B. Dunson. "Logistic Stick-Breaking Process". In: *Journal of Machine Learning Research (JMLR)* 12.1 (2011).

[139] S. Cabi, S. Gómez Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, et al. "Scaling Data-Driven Robotics with Reward Sketching and Batch Reinforcement Learning". In: *arXiv preprint arXiv:1909.12200* (2019).

[140] F. Ebert, C. Finn, A. X. Lee, and S. Levine. "Self-Supervised Visual Planning with Temporal Skip Connections". In: *Conference on Robot Learning (CoRL)* (2017).

[141] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao. "Multi-view Self-supervised Deep Learning for 6D Pose Estimation in the Amazon Picking Challenge". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2017).

[142] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[143] Y. Wu and K. He. "Group Normalization". In: *European Conference on Computer Vision (ECCV)* (2018).

[144] J. L. Ba, J. R. Kiros, and G. E. Hinton. "Layer Normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[145] A. Odena, V. Dumoulin, and C. Olah. "Deconvolution and Checkerboard Artifacts". In: *Distill* (2016). URL: `http://distill.pub/2016/deconv-checkerboard`.

[146] J. Ma and D. Yarats. "On the Adequacy of Untuned Warmup for Adaptive Otimization". In: *arXiv preprint arXiv:1910.04209* (2019).

[147] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. "Biva: A Very Deep Hierarchy of Latent Variables for Generative Modeling". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[148] A. Vahdat and J. Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". In: *arXiv preprint arXiv:2007.03898* (2020).

[149] I. Nematollahi, O. Mees, L. Hermann, and W. Burgard. "Hindsight for Foresight: Unsupervised Structured Dynamics Models from Physical Interaction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020).

[150] N. Watters, L. Matthey, M. Bosnjak, C. P. Burgess, and A. Lerchner. "COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration". In: *arXiv preprint arXiv:1905.09275* (2019).