Universidad Zaragoza
1542

Trabajo Fin de Máster

Aprendizaje de representaciones desenredadas de escenas a partir de imágenes

Learning disentangled representations of scenes from images

Author

Tomás Berriel Martins

Supervisor

Javier Civera Sancho

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2022

Abstract

Artificial intelligence is at the forefront of a technological revolution, in particular as a key component to build autonomous agents. However, not only training such agents come at a great computational cost, but they also end up lacking human basic abilities like generalization, information extrapolation, knowledge transfer between contexts, or improvisation. To overcome current limitations, agents need a deeper understanding of their environment, and more efficiently learning it from data.

There are very recent works that propose novel approaches to learn representations of the world: instead of learning invariant object encodings, they learn to isolate, or disentangle, the different variable properties which form an object. This would not only enable agents to understand object changes as modifications of one of their properties, but also to transfer such knowledge on the properties between different categories.

This Master Thesis aims to develop a new machine learning model for disentangling object properties on monocular images of scenes. Our model is based on a state-of-the-art architecture for disentangled representations learning, and our goal is to reduce the computational complexity of the base model while also improving its performance. To achieve this, we will replace a recursive unsupervised segmentation network by an encoder-decoder segmentation network. Furthermore, before training such overparametrized neural model without supervision, we will profit from transfer learning of pre-trained weights from a supervised segmentation task. After developing a first vanilla model, we have tuned it to improve its performance and generalization capability. Then, an experimental validation has been performed on two commonly used synthetic datasets, evaluating both its disentanglement performance and computational efficiency, and on a more realistic dataset to analyze the model capability on real data. The results show that our model outperforms the state of the art, while reducing its computational footprint. Nevertheless, further research is needed to bridge the gap with real world applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

On this first chapter, the elements that have motivated this Master Thesis are presented, offering a context to frame them within the Master in Robotics Graphics and Computer Vision. After that, the main goals are presented, and likewise the span of the work to reach those goals. Then, the methodology and tools applied are briefly developed, including a schedule with the temporal distribution of the different tasks. Finally, the structure of the rest of the thesis is described.

1.1 Motivation: The need for generalizable and efficient learning

Through the last 50 years, human technology has experienced an outstanding growth. Among others, robotics has gained a significant weight on the industrial and service sectors. Its surge has been mainly due to robots surpassing human speed and power in simple, short and repetitive tasks, increasing productivity levels. However, the evolution of the field progresses at a fast pace towards more complex and general tasks. Robots span already a wide range of sectors (see in Figure 1.1 several examples): from test and assembling in manufacturing plants, to hazardous materials manipulation in chemical industry, shipping, handling and quality control in logistic tasks, including transportation and surveillance in warehouses, autonomous tractors and collectors in agriculture, and even cleaning assistants like the autonomous vacuum cleaners, or, recently, cooks and waiters on restaurants. Such a wide spread of robots stands on the great developments during the last century on several areas like electronics and computer science, which together brought the so-called "Information era" and formed the perfect ground for the machine learning bloom. In the last decades we have witnessed how machine learning has become an ubiquitous technology, supported by huge processing capabilities and big amounts of data. Vast and information-rich data, like the one from smartphones and social networks, allowed robots to learn complex patterns that should result in them executing more and more complex tasks.

Figure 1.1: Robotic applications examples. From left to right, up to down: automotive manufacturing; autonomous tractor; logistic in a warehouse; waiter in a restaurant.

Even if its expected that robotics usage will expand in the next years to new applications, like fully autonomous vehicles, personal assistants, maintenance task in dangerous environments (like radioactive or submarine) or rescue aid in catastrophic scenarios, there still are many tasks that remain very challenging, such as counseling, teaching, necessities detection, short-term future prediction (predict how the environment will react after a given event), or natural human-robot interaction. At the moment, the algorithms driving robots cannot adapt to small changes on their environment (like navigating in the same place in two different seasons) and need a huge engineering effort to adapt to changes on the assigned tasks.

Regarding the price we pay for these recent technological advances, it is important to highlight that the outstanding development has been powered by the use of fossil fuels as energy sources. The gasses released to the atmosphere by these fuels are producing an excessive increase on Earth's average temperature that is impacting the weather in a way that can be critical for most of the biosphere [Masson-Delmotte et al., 2018, Shukla et al., 2019, Ripple et al., 2019, World Health Organization et al., 2014, Cahill et al., 2013]. The same factors that fueled the incredible development society has lived in for the last 50 years have brought an unexpected, and undesirable, player to the game: climate crisis [Hardy, 2003, Cahill et al., 2013, Stern and Kaufmann, 2014, Ring et al., 2012]. Furthermore, current trends on Machine Learning development tends to search for improvement in performance through bigger models with an increase in the number of parameter, data, and hardware, consuming every time more resources [Thompson et al., 2020, Dhar, 2020, Toews, 2020, ThomasGriffin, 2020]. Currently, training an Natural Language Processing pipeline with tuning and experimentation takes 36 thousand kg of CO2 emissions, which is equivalent to a year of emissions of two US citizens [Strubell et al., 2019]. Training the state of the art model EfficientNet-L2 on the ImageNet dataset with an error rate of 11.5% has an estimated burden of 450 thousand kg of CO2, equivalent to one year worth of electricity use for 80 USA homes. If the trend keeps going, it is estimated that achieving a 1% error rate would cost between 10^{18} and 10^{26} kg of CO2 [Thompson et al., 2020]. This goes directly against the Sustainable Development Goals (SDGs) of the United Nations [United Nations, 2015], designed to avoid the worst cases scenarios, and it brings out the importance of working towards a more sustainable direction [Cowls et al., 2021].

1.2 Disentangled representations

In general, even consuming the huge amount of resources detailed in the previous section, artificial agents are severely limited by the lack of basic human abilities like common-sense, the ability to improvise, generalizing and extrapolating from small data, and over all, abstract reasoning. The proper interaction of autonomous agents with a natural environment depends on its capability to deeply comprehend it, and therefore, to be able to differentiate between meaningful and trivial information, static and dynamic elements and intrinsic and extrinsic properties among others. To break the current ceiling and keep up with the progression, it is essential the improvement of autonomous navigation and autonomous agents along all this lines. Impacting directly in all these aspects is the concept of representation learning, that studies how to learn appropriate (according to certain criteria) low-dimensional models from high-dimensional big data.

More specifically, and already related to the topic of this thesis, the ability to *disentangle* the different properties of an object is a desirable aspect of representation learning. More formally, the concept of **disentanglement** is based on the idea of isolating the different factors or properties which combined form an element.

Figure 1.2 shows a simple example applied to a an object with several different properties like colour, position, scale, size and shape. Disentangling the different

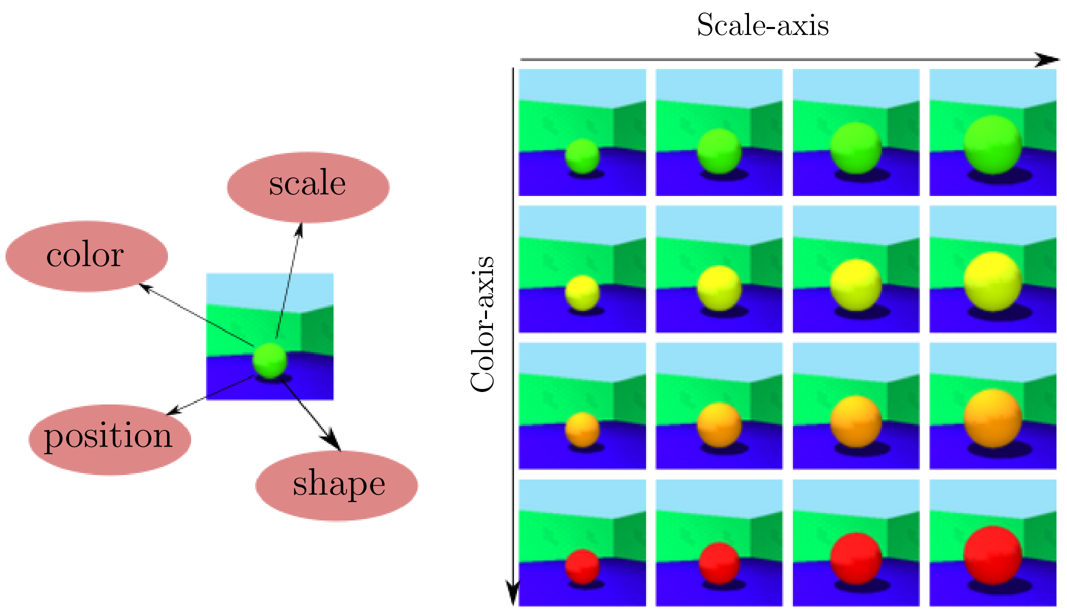


Figure 1.2: Disentanglement decomposition of a green sphere of the 3D Shapes dataset [Burgess and Kim, 2018]. The left side shows some examples it properties. The right side shows how the scene would change by altering two of those factors.

factors would allow the agent to understand changes in the environment as composed of changes in some of those individual properties, increasing its robustness. Such representations would be relevant to the building of lifelong maps and the localization of the agents within it, known as SLAM (Simultaneous Localization and Mapping [Cadena et al., 2016]).

Recently, the concept of spatial artificial intelligence has appeared, which emphasizes the SLAM role as a 3-dimensional anchor for abstract concepts related with reasoning and intelligence, more than just as a geometric estimation. This is also essential to allow autonomous agents to generate abstract models of reality, perceiving space and giving models and their properties to concrete localization in the world. Currently, most SLAM literature is centered around geometric maps, with the general goal of generating trajectories without collisions. Scenes are commonly represented as cloud of points [Mur-Artal and Tardós, 2017] and occupancy maps [Newcombe et al., 2011], and visual sensors are used. These representations have two main limitations: 1) purely geometric maps do not have relevant information for reasoning and interaction (i.e. physical properties, dynamics, semantics, hierarchies or interaction ability). This narrows the possible task than an agent can perform. And 2), the computational footprint is immense, while biologic evidence suggest that such representations, with so much excessive detail, are not essential to navigate. For example, mammals keep compact representations which abstract most of the details [Bermudez Contreras et al., 2020].

In this thesis, we aim to improve the performance and computational footprint of disentangled representation learning, advancing towards its use in more practical and realistic setups for scene understanding, SLAM or spatial AI. As we will detail in the related work in Chapter 3 and the following technical chapters, disentangled representation learning is an emerging research field still with very preliminary results, mostly in synthetic simple data. Although we will defer the technical details for later chapters, we advance that our work targets specifically the modification of a state-of-the-art architecture in a manner that supervised pre-training can be transferred to the typical unsupervised training in the disentanglement field. **Our experiments demonstrate that our proposals improve notably the state of the art both performance- and computation-wise.**

1.3 Thesis Goals

The general goal of this work is the design and implementation of efficient and accurate models for the unsupervised learning of disentangled variables from images. To reach this goal, the following intermediate goals are proposed:

- **Study of the state of the art on unsupervised learning of disentangled representations**. This includes comprehending the necessary background regarding disentanglement definitions and evaluation metrics; and state-of-the-art models for learning disentangled variables from monocular images.

- **Implementation from scratch of the baseline MONet [Burgess et al., 2019]**, a state of the art neural network for unsupervised disentanglement learning. This includes not only the model, but also training and evaluation scripts.

- **Implementation of our novel proposal MONet-RN50**. In this stage an exhaustive analysis of the MONet architecture is needed, in order to design and implement the modifications that improve both its performance and computational efficiency. Specifically, we replaced its iterative attention module by an end-to-end segmentation network to reduce its dependency on the maximum number of objects detected.

- **Thorough experimental evaluation of our proposed MONet-RN50 model against the state of the art**. We trained our MONet-RN50 models and evaluated their segmentation, reconstruction and disentanglement capabilities on multiple datasets, including an analysis of the weight's initialization impact on the systems convergence. The evaluation was planned on synthetically generated datasets with both qualitative and quantitative criterias. The models were compared against the state of the art regarding performance and computational footprint (model size and training time).

1.4 Methodology and tools

The main experiments have been carried on two synthetic datasets of the field of unsupervised disentangled representation learning: *Objects Room* [Burgess et al., 2019] and *Clevr* [Johnson et al., 2017]. Furthermore, an extra experiment has been performed on *SceneNet* [McCormac et al., 2017], although only with a qualitative analysis.

All the models and experiments have been implemented on Python 3.9.7. The library Pytorch 1.9.1 has been used for training and evaluation of the models; TensorFlow 2.4.1 for reading the datasets stored as TFrecord files; and Tensorboard 2.6.0 for visualization. All of this working with conda 4.10.3 for environment management; Git for version control, and GitHub for cloud storage of the code. The computers used run on Ubuntu 20.04. The experiments were run on two different GPUs: a Nvidia 1660Ti with CUDA 11.2 and 6 GB of RAM, used to debug the code; and a Nvidia RTX 3090 with CUDA 11.1.74 with 24 GB of RAM, to run all the training, profiling and test of the models.

Figure 1.3 presents, using a Gantt diagram, the schedule that has been followed to perform the tasks associated with the Master Thesis.

1.5 Thesis structure

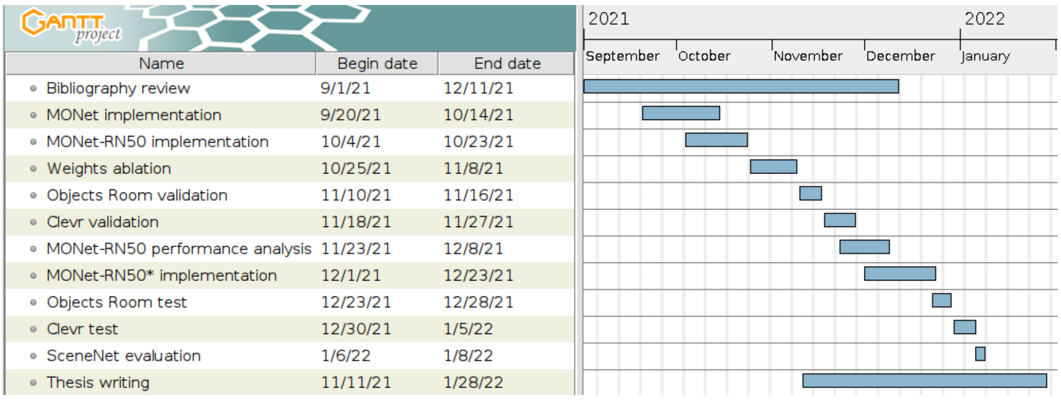
The developed work is structured on the current document on the following way:

- Chapter 1 presents the problem and the importance to both improve the ability to extract data from images and to reduce the computational burden of given algorithms.

- Chapter 2 makes a review of key fundamental concepts needed to better understand the developed work.

- Chapter 3 performs a survey of the literature and state of the art of disentangled representation learning algorithms.

- Chapter 4 explains the chosen model, and the different modifications performed to improve both its performance and efficiency.
- Chapter 5 comments and evaluates the performed experiments.
- Chapter 6 concludes with the most important results, summarizes the work, and propose lines for future work.



Name	Begin date	End date
Bibliography review	9/1/21	12/11/21
MONet implementation	9/20/21	10/14/21
MONet-RN50 implementation	10/4/21	10/23/21
Weights ablation	10/25/21	11/8/21
Objects Room validation	11/10/21	11/16/21
Clevr validation	11/18/21	11/27/21
MONet-RN50 performance analysis	11/23/21	12/8/21
MONet-RN50* implementation	12/1/21	12/23/21
Objects Room test	12/23/21	12/28/21
Clevr test	12/30/21	1/5/22
SceneNet evaluation	1/6/22	1/8/22
Thesis writing	11/11/21	1/28/22

Figure 1.3: Schedule of the tasks performed through the Master Thesis.

Chapter 2

Background

The objective of this chapter is to introduce the necessary background to properly understand the developed work. First, a brief review of machine learning and deep learning will be done, and after that four neural architectures that are relevant in this thesis will be introduced.

2.1 Deep Learning foundations

Machine learning is the field that focuses on creating algorithms that autonomously adjust its parameters through an optimization, to improve its performance in inference [Murphy, 2012]. Although there is a great variety of machine learning algorithms, in the last years neural networks have taken a leading role in many fields. Neural networks have loose connections to the human brain by concatenating layers of interconnected functions or processors. Each layer processes input data to generate an output that is passed to the next layer. The interconnections between the layers, and within each layer, depend on the particular architecture of each network. Two popular layers, especially in computer vision, are linear and convolutional layers. Deep learning is the sub-field of machine learning specialized in creating models with a high number of concatenated layers [Goodfellow et al., 2016, LeCun et al., 2015].

Training There are different types of optimization or training for neural networks. Supervised learning is possibly the most used one, but unsupervised learning certainly has a bigger potential. Supervised training is based on, given some data, computing a prediction and an error by comparing it against ground-truth labels; and using such supervisory signal to adjust the parameters. Instead, unsupervised learning training is characterized by the lack of ground-truth labels. The training is also based on generating a prediction given some data, but instead of comparing with the ground-truth, a data-related score is computed and used as error to optimize the model parameters. Even if there are different optimization algorithms, this is usually done using backpropagation and stochastic gradient descent [Goodfellow et al., 2016].

2.2 Relevant architectures

Different tasks and data types have given place to different neural networks architectures. The ones that will be mentioned, due to their use in this work are Autoencoders [Goodfellow et al., 2016]; Variational Autoencoders [Kingma and Welling, 2019]; Residual Networks [He et al., 2016]; and U-net architectures [Ronneberger et al., 2015].

2.2.1 Autoencoders

An autoencoder is a type of neural networks used to learn efficient encodings of unlabeled data. An autoencoder consists of two parts, the encoder and the decoder. First the encoder function maps the input $\boldsymbol{x} \in \mathcal{X}$ from the input space $\mathcal{X}$, to a latent representation $\boldsymbol{h} \in \mathcal{F}$ in the feature space $\mathcal{F}$. Then, the decoder, given the latent representation $\boldsymbol{h}$, generates a reconstruction $\boldsymbol{x'} \in \mathcal{X}$.

These two parts can be described as mapping functions between the input space $\mathcal{X}$ and the feature space $\mathcal{F}$, such that:

$$f_\phi(\boldsymbol{x}) : \mathcal{X} \to \mathcal{F}, \tag{2.1}$$

$$g_\psi(\boldsymbol{h}) : \mathcal{F} \to \mathcal{X}, \tag{2.2}$$

with the encoder function f being parametrized by ϕ, and the decoder function g parametrized by ψ.

The autoencoders parameters are optimized to minimize a reconstruction error, such as the squared error,

$$\mathcal{L}_{\phi,\psi}(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x'}\|_2, \tag{2.3}$$

where $\|\|_2$ is the L2 norm.

When the encoder and decoder have too much capacity or become too big, the autoencoder can converge to copying the input instead of extracting useful information about its distribution. Theoretically, a sufficiently big autoencoder could learn to code the input data to a single integer, and recover the input data from that integer without learning anything useful about the dataset [Goodfellow et al., 2016]. As a consequence, the training and the architectures are restricted in manners that allow them to generate only approximate copies. This forces the model to prioritize different aspects of the input, and often leads to learning useful properties of the data.

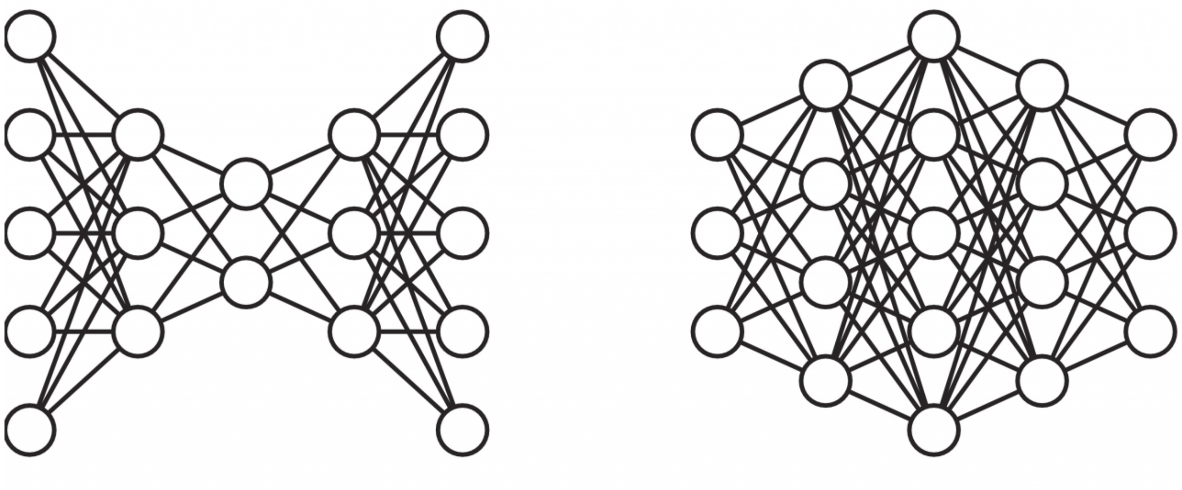


Figure 2.1: Undercomplete (left) and overcomplete (right) autoencoders.

Depending on the relation between the dimensionality of the input $\boldsymbol{x}$ and the hidden representation $\boldsymbol{h}$, autoencoders can be classified as undercomplete ($dim(\boldsymbol{x}) > dim(\boldsymbol{h})$); overcomplete ($dim(\boldsymbol{x}) < dim(\boldsymbol{h})$), or with equal dimensions ($dim(\boldsymbol{x}) = dim(\boldsymbol{h})$), see Figure 2.1. Usually, undercomplete autoencoders are used because they restrict the capacity of the module forcing it to capture the most salient features of the training data. Nevertheless, to train successfully an architecture one would like to choose the dimension and capacity of the autoencoder based on the complexity of the distribution to be modeled.

Regularized autoencoders provide the ability to do so using a loss function that encourages other properties besides copying the input data. This includes sparsity of the representation (Sparse Autoencoders), smallness of the derivative of the representations (Contractive Autoencoders), and robustness to noise or to missing inputs (Denoising Autoencoders).

Nearly any generative model with latent variables and a procedure to compute latent representations given an input, can be considered a particular form of autoencoder. This includes the Variational Autoencoder and the Generative Stochastic Networks. These are models that naturally learn useful representations because they are trained to approximately maximize the probability of the training data, rather than just copying it.

Autoencoders have been part of the landscape of neural networks for decades [Lecun, 1987, Bourlard and Kamp, 1988, Hinton and Zemel, 1993]. They were used for dimensionality reduction or feature learning. Recently, they have also been used for generative modeling due to the theoretical connections with latent variables.

2.2.2 Variational Autoencoders

Even if the Variational Autoencoder [Kingma and Welling, 2014] can be viewed as a particular form of autoencoder, it belongs to the families of probabilistic graphical models, and variational Bayesian methods. Its has a different goal and mathematical formulation.

The objective of the VAE is to model the input data $\boldsymbol{x}$, following an unknown probability function $P(\boldsymbol{x})$ and generated by a multivariate latent encoding vector $\boldsymbol{z}$, as a distribution $p_\theta(\boldsymbol{x})$ parametrized by the set of network parameters θ. The distribution can be formalized as

$$p_\theta(\boldsymbol{x}) = \int_{\boldsymbol{z}} p_\theta(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}, \tag{2.4}$$

with $p_\theta(\boldsymbol{x})$ the evidence of the model's data with marginalization performed over unobserved variables. Therefore $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ represents the joint distribution between input data and the latent representations according to the model's parameters θ. Applying the chain rule, this can be extended to

$$p_\theta(\boldsymbol{x}) = \int_{\boldsymbol{z}} p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\theta(\boldsymbol{z}) d\boldsymbol{z}. \tag{2.5}$$

The standard version of VAE assumes that $\boldsymbol{z}$ has a finite dimension, that $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is a Gaussian distribution, and then that $p_\theta(\boldsymbol{x})$ is a mixture of Gaussian distributions. Then $p_\theta(\boldsymbol{z})$ is the prior distribution; $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the conditional likelihood, and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is the posterior distribution. Given that the computation of $p_\theta(\boldsymbol{x})$ is very expensive, and in some cases intractable, the posterior is approximated for a new function to make it tractable,

$$q_\Phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_\theta(\boldsymbol{z}|\boldsymbol{x}), \tag{2.6}$$

where Φ is the set of real values that parametrizes q. This allows to translate the problem to the autoencoder domain, where $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is carried by the probabilistic decoder, while $q_\Phi(\boldsymbol{z}|\boldsymbol{x})$ is computed by the probabilistic decoder.

ELBO VAE are trained minimizing the Evidence Lower Bound (ELBO). For any approximate distribution $q_\Phi(\boldsymbol{z}|\boldsymbol{x})$, the ELBO can be deduced from the log likelihood, see Eq. (2.7), (2.8), (2.9), (2.10).

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x})\right] && (2.7)\\
&= \mathbb{E}_{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right] && (2.8)\\
&= \mathbb{E}_{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\frac{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right] && (2.9)\\
&= \underbrace{\mathbb{E}_{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\right]\right]}_{\mathcal{L}_{\theta,\Phi}(\boldsymbol{x})\ \text{(ELBO)}} + \underbrace{\mathbb{E}_{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left[\frac{q_\Phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right]\right]}_{=D_{KL}(q_\Phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x}))} && (2.10)
\end{aligned}$$

By definition, the KL divergence is non-negative, and zero if, and only if, the two distributions are equals. Due to this, the first term of eq. (2.10), the evidence lower bound, is a lower bound on the log-likelihood of the data.

$$\begin{aligned}
\mathcal{L}_{\theta,\Phi}(\boldsymbol{x}) &= \log p_\theta(\boldsymbol{x}) - D_{KL}(q_\Phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})) && (2.11)\\
&\leq \log p_\theta(\boldsymbol{x}) && (2.12)
\end{aligned}$$

As a consequence, the maximization of the ELBO $\mathcal{L}_{\theta,\Phi}(\boldsymbol{x})$ with respect to the parameters θ and Φ will concurrently optimize two aspects of interest: 1) it will approximately maximize the marginal log-likelihood $p_\theta(\boldsymbol{x})$, improving the generative model; and 2) it will improve the approximate posterior $q_\Phi(\boldsymbol{z}|\boldsymbol{x})$ by minimizing its KL divergence with the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$

Finally, to make the VAE differentiable the reparametrization trick is performed. It basically removes the stochastic component from the network parameters, by sampling it from a random node that will not affect the gradient back-propagation through the rest of the net.

2.2.3 Residual Networks

The last decade has seen how deep learning has reduced error rates by increasing the number of layers in neural network architectures. At the beginning, this approach faced a great problem: the vanishing or exploding gradient. It was observed that increasing the number of layers could cause the gradient to either become zero, vanish, or become too large and explode. He et al. introduced a new type of architecture with skip connections as a solution to this problem: Residual Networks (ResNet) [He et al., 2016].

Instead of using a stack of layers to directly fit a desired underlying mapping $\mathcal{H}(\boldsymbol{x})$, they let the stacked non-linear layers fit another mapping $\mathcal{F}(\boldsymbol{x}) = \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}$, and recast the original mapping into $\mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$. They used the addition of the input to the

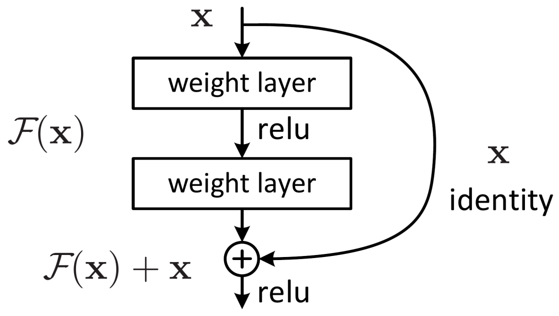


Figure 2.2: Residual Learning building block. Schematic from [He et al., 2016]

output, as a preconditioning to help solve the problem. This follows the hypothesis that its easier to optimize a residual mapping than to optimize an original unreferenced mapping. If the objective function is closer to an identity mapping than to a zero mapping, it should be easier for a solver to find the solution as perturbations with reference to the identity mapping, than to learn the function as a new one.

In practice, the idea is to create shortcuts to skip a few layers and directly connect to the output. They implemented this creating ResNet.

The architecture is based on four main blocks of Convolutional Neural Networks with internal skip-connections, each with a different number of channels and layers. They proposed five different configurations of ResNet changing the number of layers of the different blocks giving place to five models of different depth (18, 34, 50, 101, and 152 layers). ResNet showed to the world how to create deeper models with lower error rate.

2.2.4 U-net

U-net [Ronneberger et al., 2015] was proposed to generate segmentation maps of cells images on an end-to-end architecture, and has been used as blue-print for several other segmentation tasks. The objective was to not only detected the elements presents on the scene, but to classify each pixel of the input image to a class.

It is a neural network that combines an autoencoder architecture with skip-connections between the encoder and the decoder. These skip-connections are different than ResNet's in that they do not try to perform an identity mapping, but rather they are concatenated.

The encoder consists of five blocks, each made of two convolutional layers with respective activation functions, followed by a maxpooling operation for downsampling. Each block doubles the number of features, and halves the size of each side of the image, except for the last one that does not perform downsampling. The output of the encoder is upsampled trough a so called "up-convolution", and fed to the decoder. This

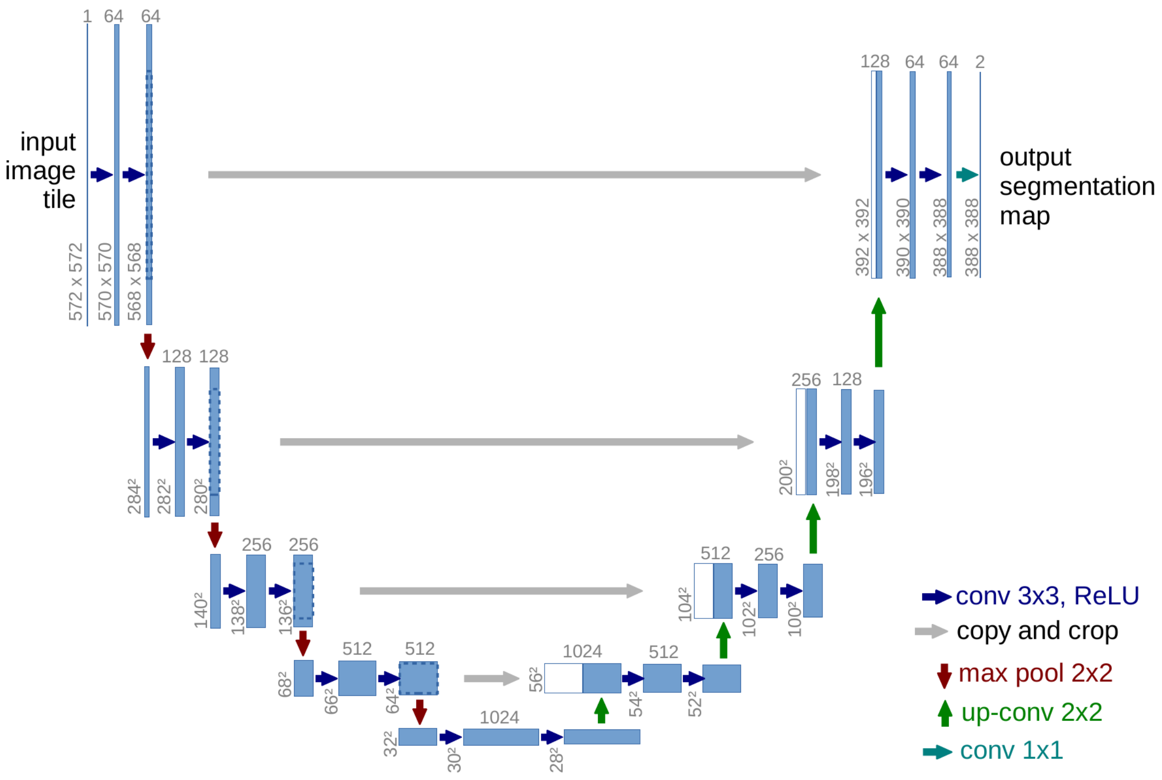


Figure 2.3: U-net architecture. Schematic from [Ronneberger et al., 2015].

is made of four blocks each of which halves the number of channels, and doubles each side of the image. In each block, first the input is concatenated with the output of the corresponding block of the encoder. Then it is passed thorugh a pair of convolutional layers, and finally is upsampled and passed to the next block.

Chapter 3

Related Work

Representation Learning is a growing field which has started to attract interest in the last five years. This task can be performed by both supervised an unsupervised methods. Due to the high complexity of generating a classical supervised dataset, this review focuses on unsupervised approaches. There are multiple approaches, although not all learn to disentangle the properties of the detected elements. Two main tendencies can be observed in the field: 1) models based on *AIR* (Attend Infer Repeat) [Ali Eslami et al., 2016]; and 2) models that build on the work from *β-VAE* [Higgins et al., 2017].

3.1 AIR based approaches

The first disentangled model presented was ***AIR***, an unsupervised recurrent neural network designed to attend to different objects in a scene. Each object is sequentially processed, and **three object-oriented latent representations** are prescribed: **'What', 'Where' and 'Presence'**. 'What' and 'Where' respectively encode the appearance information and the location of the object. Instead, 'Presence' encodes how many steps the recurrent neural network runs, and acts as an interruption variable when the model decides that all elements have been processed. Its main **drawback** stands on its complexity: **it scales with the number of objects**, therefore it is not suited for scenes with a great number of elements.

SQAIR (Sequential Attend Infer Repeat) [Kosiorek et al., 2018], is an augmentation of *AIR* with a state-space probabilistic model. It is composed of two parts: Discovery, responsible for detecting new objects at each time-step; and Propagation, responsible for updating latent variables from the previous time-step given the new observations.

SPAIR (Spatially Invariant Attend Infer Repeat) [Crawford and Pineau, 2019] attempts to address *AIR* scalability issue by replacing the recurrent network with a convolutional network. Similar to YOLO [Redmon et al., 2015], the locations of objects are specified relative to local grid cells rather than the entire image, allowing for spatially invariant computations. In the encoder network, a convolutional neural network is first used to map the image to a feature volume with dimensions equal to a pre-specified grid size. Then, each cell of the grid is processed sequentially to produce objects. This is done sequentially because the processing of each cell takes as input feature vectors and sampled objects of nearby cells that have already been processed. Therefore, *SPAIR* scales with the pre-defined grid size which also represents the maximum number of objects that can be detected. Objects are encoded with 4 latent variables: the three latent variables of *AIR* (‘What’, ‘Where’ and ‘Presence’), plus a fourth latent that encodes the depth.

SPACE (Spatially Parallel Attention and Component Extraction) [Lin et al., 2020] propose a unified probabilistic generative model that tries to combine the benefits of the spatial-attention and scene-mixture models. It consists of a foreground module and a background module used to compute the same 4 latent variables than *SPAIR*. In the foreground module, the input image is divided into grid cells, and an image encoder is used to compute the ‘Where’, depth, and ‘Presence’ for each cell in parallel. ‘Where’ is used to identify proposal bounding boxes, and a spatial transformer is used to attend to each bounding box in parallel, computing a ‘What’ encoding for each cell. The model selects patches using the bounding boxes and reconstructs them using a VAE from all the foreground latent. The background module segments the scene into K components using a pixel-wise mixture model. Each components consists of a pair of latents where one models the mixing probability of the component and the other models the RGB distribution of the component. The components are combined to reconstruct the background using a VAE. The reconstructed background and foreground are then combined using a pixel-wise mixture model to generate the full reconstructed image.

SCALOR [Jiang et al., 2020] is a model based on *SQAIR*, which represents each image through one latent variable for the foreground, one for the background, and a set for each element on the scene. The set for the individual elements is disentangled between the three basic variables of *AIR*. In contrast to previous approaches, the ‘Where’ is further disentangled between the scale, the position, and the relative depth (for occlusion). The model uses several Neural Networks to encode and decode the different variables, a proposal-rejection procedure to add new elements to track, and a Recurrent Neural Network to track the different entities.

3.2 β-VAE based approaches

The second main tendency is based on ***β-VAE*** [Higgins et al., 2017]. In it the original **VAE** framework has been **augmented including an hyperparameter**, β, in the **loss function**. It is used to modulate the learning constrain applied to the model, **controlling** the emphasis on **learning** statistically **independent latent factors**.

Annealed-VAE [Burgess et al., 2018] proposed a modification to the training regime of *β-VAE* that progressively increases the information capacity of the latent code during training, facilitating the robust learning of disentangled representations in *β-VAE* without the trade-off in reconstruction accuracy. *Factor-VAE* [Kim and Mnih, 2018] included a method that disentangles by encouraging the distribution of the representation to be factorial and, as a consequence, independent across the dimensions.

β-TCVAE (Total Correlation Variatioanl Autoencoder) [Chen et al., 2018] introduced a refinement of *β-VAE* that penalizes the total correlation between the latent variables in the ELBO, with a tractable, but biased, Monte-Carlo estimator. *DIP-VAE* (Disentangled Inferred Prior) [Kumar et al., 2018] introduced a regularizer in the generic loss function to minimize the distance between the inferred prior and the disentangled generative prior distributions, with the objective of achieving better control on the disentanglement.

MONet (Multi Object Network) [Burgess et al., 2019] is a neural network model trained end-to-end. It is comprised of two modules: an attention module based on a recurrent neural network which learns to generate individual segmentation masks for the different elements on the scene; and a component module based on a VAE, which learns to extract the latent variables for the object relative to each mask.

Genesis [Engelcke et al., 2019] is an extension of *MONet* that tries to capture the dependencies between different latent factors with an autoregressive prior that is learned alongside a sequential inference network, which enables it to also sample new scenes.

IODINE [Greff et al., 2019] models images with a spatial Gaussian mixture model where each mixing component corresponds to a single object to represent the relationship between the different elements. The architecture is a spatial broadcast decoder trained to minimize the ELBO. Each object latent is decoded separately into pixels-wise means and mask-logits, which parametrize the spatial mixture, through an iterative inference process. *Slot Attention* [Locatello et al., 2020] is an architectural component that takes the feature vectors of perceptual modules, like CNN, as a set of feature vectors that are abstract representations of objects or entities, called slots. Slots compete for explaining parts of the input through iterative attention mechanism and update their representations using recurrent update function. It uses the same decoder and reconstruction loss than *IODINE*.

Constellation [Whittington et al., 2021] uses *MONet* to extract entities and a Graph Neural Network to relate them. It extracts the patterns of relationship between the different entities and is able to generate novel scenes changing them. It also adapt *SCAN* [Higgins et al., 2018] to relate words to the learned latent representations.

OCIG (Object-Centric Image Generation) [Anciukevicius et al., 2020] is a generative model that uses a Bayesian hierarchical model to represent the image in three different levels: first level of random variable modelling dependencies between objects; second level modelling dependencies between pixels of the same object; and third level modelling pixel-level noise. A compositor function combines the decoded objects and background according to the position, depths, and masks. A variational posterior distribution is parametrized by an encoder network to predict parameters of the posterior distribution over latent variables 'explaining' a given image. The ELBO is maximized applying [Higgins et al., 2017] modification, and a substitution of the KL term for the objects's position with the L1 divergence between the aggregated posterior and the uniform prior, to encourage use of the factored position.

Genesis v2 [Engelcke et al., 2021] is an embedding-based approach in which embeddings of pixels are clustered in a differentiable fashion using a stochastic, non-parametric stick-breaking process. Similar to iterative refinement, this clustering procedure also leads to randomly ordered object representations, but without the need of initializing a fixed number of clusters a priori.

SIMONe [Kabra et al., 2021] is a VAE consisting of an inference network based on a Transformer, which infers latent variables from a given input sequence; and a generative process (decoder) which decodes these latent variables back into pixels.

3.3 Other approaches

Despite the prevalence of the two main currents, there are also other approaches.

SCAE [Kosiorek et al., 2019] tried a change of perspective. Images are decomposed into 1) parts, represented by part capsules that encode pose (6 degree of freedom), presence, and special features; 2) objects, represented by object capsules; and 3) object-viewer and object-part relationship. The part discovery problem is framed as auto-encoding: the encoder learns to infer the poses and presences of different part capsules, while the decoder learns an image template for each part.

OP3 [Veerapaneni et al., 2019] defines the state of the world as the combination of the state of the objects in the world. It infers a set of entity variables from an observation and predicts their future states given a set of sequence of actions. For each entity, first the effect of the action is estimated using a generic dynamic function, and then the effect of each entity on the others is computed. To bid pixels to entities, an iterative inference algorithm is used to iteratively refine an initial guess for the posterior parameters. To disambiguate objects, actions are incorporated into an interactive inference algorithm, based on [Marino et al., 2018]. Three neural networks are optimized backpropagating the ELBO to estimate the different distributions parameters (observation model G, dynamic model D, and recognition distribution Q). The model is evaluated to predict the interaction between objects when picking and dropping to build a structure of blocks; to perform multitask planning by generating the actions that it would perform to build a goal structure given an initial state; and in a real world scenario. The model is seemingly able to predict the interactions properly, although the used object are monotonous, simplistic, and with a stable geometry (cubes), and when the number of objects increases the reconstruction gets blurrier.

3.4 Common limitations

All the proposed approaches share some **common problems: they are evaluated on synthetically generated datasets, that lack the visual complexity or noise of real life scenes; they have significantly high computational demands; and, overall, lack of both a common definition of disentanglement and proper evaluation metrics**.

The first procedure used to evaluate disentanglement was visual inspection. Due to the subjective nature of this procedure, with the growth of the field new criteria have been proposed. The lack of a common definition of disentanglement gives place to an ambiguity regarding what has to be evaluated.

Usually, each new criteria tries to evaluate an aspect that was not properly evaluated by previous metrics. Despite that, it is not clear what each metric tries to quantify and under which condition. Both *β-VAE* and *Factor-VAE* proposed an homonymous metric. The *DCI* (Disentanglement Completeness and Informativeness) [Eastwood and Williams, 2018] has been proposed as a metric composed of three parts. Each of the parts separately quantified a property between explicitness, compactness, and modularity. Another approach used group and representation theory to propose a formalized definition of disentanglement [Higgins et al., 2018]. It was focused on the transformation properties of the world, assuming that those are the ones that give exploitable structure to any kind of data.

A large-scale study performed on model selection and disentanglement evaluation

[Locatello et al., 2019], found that even if most of the metrics were correlated on simplistic datasets, they did not keep the correlation on more realistic datasets. Furthermore, they also observed how for current architectures, random seeds and hyper-parameters selection have a greater importance on the performance than the architecture selection.

A review of the different metrics (BetaVAE, FactorVAE, DCI, MIG [Chen et al., 2018] and SAP [Kumar et al., 2018]) to choose the most appropriate was done defining two intuitive criteria that all metric should fulfill [Sepliarskaia et al., 2019]. This were: 1) give high scores to representations that satisfy the characteristic that the metric reflects; and 2) give low score to representations that do not satisfy the characteristic that the metric reflects. Surprisingly, they found that only MIG fulfilled both characteristics, while the rest did not fulfill either. Another approach proposed a new definition of disentangled representations based on information theory [Do and Tran, 2020]. They also designed several metrics (WSEPIN, WINDIN, RMIG and JEMMIG) for evaluation along three new dimensions: informativeness, separability, and interpretability. These metrics are evaluated on general known models showing more robust and sensible evaluations than previous metrics, compatible with visual results. They conclude that for a proper evaluation of disentanglement, ground truth factors should be used. An in-depth analysis [Zaidi et al., 2020] of disentanglement metrics established a taxonomy of metric families, underlying strengths and shortcomings, and propose a guide for selecting appropriate metrics depending on the application. Unfortunately, the analysis only focus on supervised metrics due to the lack of metrics for unsupervised scenarios (note that all the metrics mentioned require ground-truth latent factors).

Chapter 4

Architectures for disentanglement learning

After reviewing both the fundamental concepts involved in the field, and the most recent state of the art, we will detail here the design of our model. This chapter is centered around the design of a model that alleviates some of the problems mentioned in Chapter 3. First, the details of the state of the art model chosen as base model are explained. After that, our modifications to improve the performance and efficiency are proposed. Finally, a brief analysis on the weights initialization is made. We choose MONet [Burgess et al., 2019] as the base architecture due to its ability to learn not only the position of the elements but also to disentangle different appearance properties (like colour, shape, and size) while other approaches only learn 'What' without discerning further between its properties. The importance of MONet is also shown on how it is used as base-model for others like Genesis [Engelcke et al., 2019] and Constellation [Whittington et al., 2021].

4.1 MONet

The Multi-Object Network (*MONet*) is an unsupervised neural network-based model that decomposes a scene into a specified number K of slots or compositional parts, and learns a coded latent representation to model each of the parts. It has two different modules. An attention module that segments the input image to generate a spatial mask for each part $k \in \{0, \dots, K\}$ of the scene; and the component Variational Autoencoder (VAE), which given the input image and the spatial masks, independently models each of the parts.

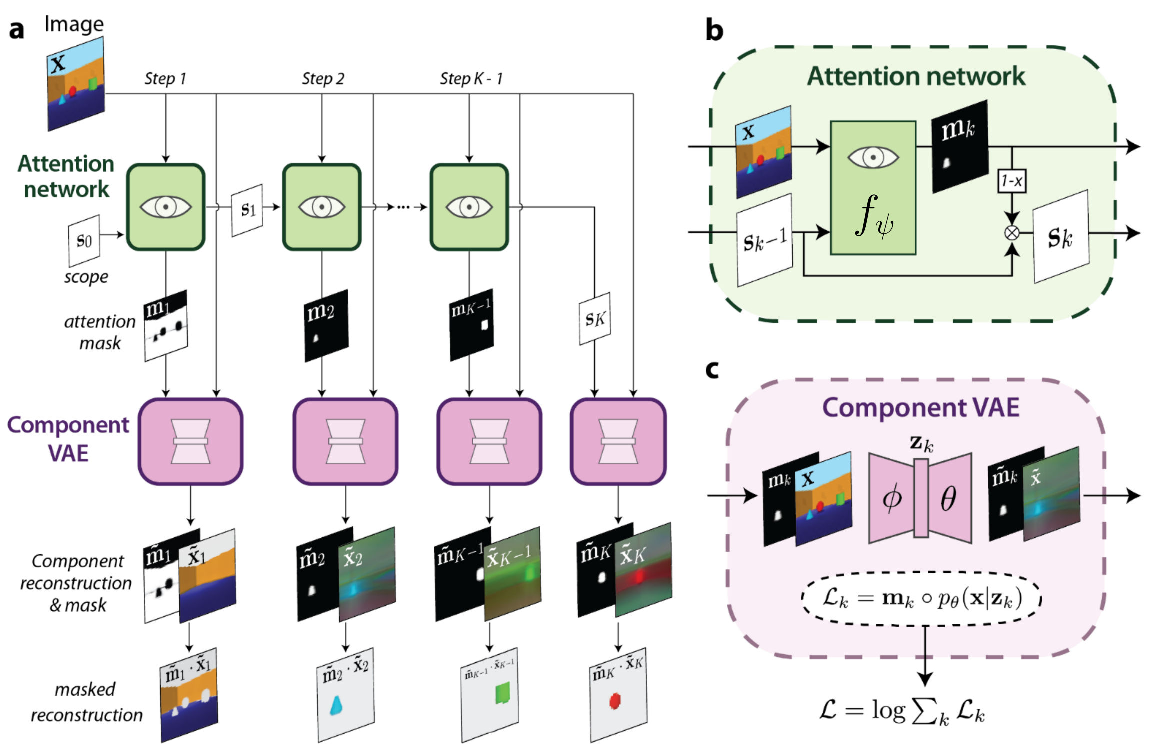

a
Image
Step 1
Step 2
Step K - 1
Attention network
scope
attention mask
Component VAE
Component reconstruction & mask
masked reconstruction
b
Attention network
1-x
c
Component VAE
$\mathcal{L}_k = \mathbf{m}_k \circ p_\theta(\mathbf{x}|\mathbf{z}_k)$
$\mathcal{L} = \log \sum_k \mathcal{L}_k$

Figure 4.1: Schematich of MONet made by [Burgess et al., 2019]. (a) Overall model architecture. The attention network recurrently generates a set of masks to condition the component VAE. Each mask corresponds to a component on the image, and labels which pixels to focus on to represent and reconstruct for that component. (b) In each step, the attention module takes as inputs the input image and a scope that quantifies how much each pixel has been explained. Then the attention network generates an attention mask, and the mask and the scope are combined to compute a new scope for the next step. (c) For each generated mask, the component VAE takes as input the mask and the input image, computes the latent variables and tries to reconstruct both the mask, and region of the image of the mask's component. It is pressured to model only the masked region by applying the mask to weight the component likelihood in the loss. As a consequence, it is unconstrained outside of the masked region, enabling it to model occluded regions.

4.1.1 Attention

The attention module takes as input the scene images, and it iteratively computes segmentation maps for each one of the K slots, ensuring that the whole image is explained. This is done through an autoregressive process with an ongoing state, the scope $\boldsymbol{s}_k$, that tracks which parts of the image have yet to be explained and is updated after each attention step.

At each step, the previous scope $\boldsymbol{s}_{k-1}$ and the input image $\boldsymbol{x}$ are the input to a UNet segmentation network f parametrised by ψ, which generates the logits $\boldsymbol{\alpha}_k$. These are first normalized with an activation function σ, and then used to update the scope,

$$\boldsymbol{\alpha}_k = f_\psi(\boldsymbol{x}, \boldsymbol{s}_{k-1}) \tag{4.1}$$

$$\boldsymbol{s}_k = \boldsymbol{s}_{k-1}(\mathbf{1} - \sigma(\boldsymbol{\alpha}_k)), \tag{4.2}$$

and to compute the mask $\boldsymbol{m}_k$ for the k^{th} component,

$$\boldsymbol{m}_k = \boldsymbol{s}_{k-1}\sigma(\boldsymbol{\alpha}_k). \tag{4.3}$$

For the first step $k = 1$, the scope is set as the whole image $\boldsymbol{s}_0 = \mathbf{1}$; and for the last step $k = K$ the mask is the remaining scope $\boldsymbol{m}_k = \boldsymbol{s}_{K-1}$, to ensure that $\sum_{k=1}^{K} \boldsymbol{m}_k = \mathbf{1}$ and that the whole image is explained.

Implementation details.

The attention module relies on a UNet-based neural network to compute the mask $\boldsymbol{m}_k$ for each time step k. The network follows a standard U-Net [Ronneberger et al., 2015] architecture with five blocks each on the down-sampling and up-sampling paths (except for CLEVR, where it is increased to six blocks each). Each block consist of a 3×3 bias-free convolution with stride 1, followed by instance normalization [Ulyanov et al., 2017] with a learned bias term, followed by a ReLU activation, and finally downsampled or upsampled by a factor of 2 using nearest neighbour-resizing (no resize occurs in the last block of each path).
Skip tensors are collected from each block in the downsampling path after the ReLU activation function. These are concatenated with input tensors along the upsampling blocks before the convolutional layer.
A Multi Layer Perceptron (MLP) with 3 layers serves as the non-skip connection between the downsampling and the upsampling paths. The input of the MLP is the last skip tensor from the downsampling path after flattening. The intermediate hidden layers were sized $(128, 128)$, and the final output is then reshaped to match that of the last skip tensor. These are concatenated together and fed into the upsampling path.
Following the upsampling path, a final 1×1 convolution with stride 1 and single output channel transforms the U-Net output into the logits $\boldsymbol{\alpha}_k$, that are passed to the activation function. This is implemented using a sigmoid activation function.

As it is usual in machine learning, to improve the conditioning and speed of the operations, these are computed in the logarithmic space. This transforms multiplications in sums, and divisions in substractions, which are less computationally expensive. Therefore the operations performed to compute the scope and mask, Eq. (4.2) and (4.3), are simplified to

$$\log \boldsymbol{s}_k = \log \boldsymbol{s}_{k-1} + \log(\mathbf{1} - \sigma(\boldsymbol{\alpha}_k)) \tag{4.4}$$

and

$$\log(\boldsymbol{m}_k) = \log(\boldsymbol{s}_{k-1}) + \log(\sigma(\boldsymbol{\alpha}_k)). \tag{4.5}$$

Using the logarithmic sigmoid as activation function, a mathematical simplification can be performed to reduce the number of operations, as will be show in the following demonstration. To compute only $\log \sigma(\boldsymbol{\alpha})$ and reduce the log operations performed, Eq. (4.4) can be further simplified:

$$\begin{aligned}
\log \boldsymbol{s}_k &= \log \boldsymbol{s}_{k-1} + \log\left(1 - \sigma(\boldsymbol{\alpha}_k)\right) \\
&= \log \boldsymbol{s}_{k-1} + \log\left(1 - \frac{1}{1 + e^{-\boldsymbol{\alpha}_k}}\right) \\
&= \log \boldsymbol{s}_{k-1} + \log\left(\frac{1 + e^{-\boldsymbol{\alpha}_k} - 1}{1 + e^{-\boldsymbol{\alpha}_k}}\right) \\
&= \log \boldsymbol{s}_{k-1} + \log\left(e^{-\boldsymbol{\alpha}_k}\right) - \log\left(1 + e^{-\boldsymbol{\alpha}_k}\right) \\
&= \log \boldsymbol{s}_{k-1} - \boldsymbol{\alpha}_k + \log\left(\frac{1}{1 + e^{-\boldsymbol{\alpha}_k}}\right) \\
&= \log \boldsymbol{s}_{k-1} - \boldsymbol{\alpha}_k + \log\left(\sigma(\boldsymbol{\alpha}_k)\right).
\end{aligned} \tag{4.6}$$

The final result of Eq. (4.6) is implemented on the neural network.

Finally, to adapt to the increase in complexity on the CLEVR dataset, the attention network is modified to include six blocks in both the encoder and decoder [Burgess et al., 2019]. As before, a downsampling/upsampling is performed after each block, except for the last block in each path.

4.1.2 Component VAE

The component VAE is composed of an encoder parametrised by ϕ and a decoder parametrised by θ. The encoder parametrises a distribution over each slot's component latents $\boldsymbol{z}_k$, conditioned on the input image $\boldsymbol{x}$ and the attention masks $\boldsymbol{m}_k$. The attention masks are used to condition the VAE on which regions of the image the representation should focus via its latent posterior $q_\phi(\boldsymbol{z}_k|\boldsymbol{x}, \boldsymbol{m}_k)$. Furthermore, the VAE is also required to model the attention masks over the K components, where their distribution $p(\boldsymbol{c}|\boldsymbol{m}_k)$ is the probability that pixels belong to a particular component k, i.e. $\boldsymbol{m}_k = p(\boldsymbol{c} = k|\boldsymbol{m}_k)$.

The decoder learns to reconstruct both the component part of the input image $\tilde{\boldsymbol{x}}_k$ and the attention mask $\tilde{\boldsymbol{m}}_k$ for each of the K slots, given the component latents $\boldsymbol{z}_k$ computed by the encoder. Finally, these are combined to generate a reconstructed image $\tilde{\boldsymbol{x}}$,

$$\tilde{\boldsymbol{x}} = \sum_K \tilde{\boldsymbol{m}}_k \circ \tilde{\boldsymbol{x}}_k,$$

where $\circ$ is the Hadamard product.

Implementation details

Our encoder is a standard Convolutional Neural Network (CNN) with 3×3 kernels, stride 2, and ReLU activations. The CNN has 4 layers with 32, 32, 64 and 64 output channels respectively. The output of the encoder is flattened and fed to the bottleneck. This is a 2-layer MLP with output sizes of 256 and 32. The MLP output parametrizes the mean $\boldsymbol{\mu}$ and the logarithm of the standard deviation $\log \boldsymbol{\sigma}$ of a 16 dimensional Gaussian latent posterior. The decoder is a spatial broadcast decoder [Watters et al., 2019] which transforms the sampled latent vectors $\boldsymbol{z}_k$ into the reconstructed image component and mask distributions. The input to the broadcast decoder is a spatial tiling of $\boldsymbol{z}_k$ concatenated with a pair of coordinates channels – one for each spatial transformation – ranging from -1 to 1. These go through a four-layer CNN with no padding, 3×3 kernels, stride 1, 32 output channels, and ReLU activations. The height and width of the input are both made 8 pixels larger than the target image size to accommodate for the lack of padding. A final 1×1 convolutional layer transforms the output into 4 channels: 3 RGB channels for the means of the image components $\hat{\boldsymbol{x}}_k$, and 1 for the logits used for the softmax operation to compute the reconstructed attention masks $\hat{\boldsymbol{m}}_k$. The output component distribution was an independent pixel-wise Gaussian with fixed scales. For the experiments, the first "background" component scale was fixed at $\sigma_{bg} = 0.09$, and for the $K - 1$ remaining "foreground" components, the scale was fixed at $\sigma_{fg} = 0.11$.

4.1.3 Loss function

MONet is trained in an unsupervised way, using an extension of a standard VAE loss. The loss has three terms: 1) the negative log likelihood; 2) the KL divergence of the latent posterior; and 2) the KL divergence of the attention mask.

The first term of the loss is the negative log likelihood of the component VAE decoder. The likelihood term $p_\theta(\boldsymbol{x}|\boldsymbol{z}_k)$ is weighted according to the masks $\boldsymbol{m}_k$ generated by the attention network, such that the parts of the image outside of each components segmentation are unconstrained. The second term is the KL divergence of the latent posterior generated by the component VAE encoder (factorised across slots), with the latent prior. This KL divergence is weighted with an hyperparameter β which can be tuned to encourage learning of disentangled representations, following $\beta - VAE$ [Higgins et al., 2017]. The last term is the KL divergence between the attention mask distribution $q_\psi(\boldsymbol{c}|\boldsymbol{x})$ and the component VAE's decoded mask distribution $p_\theta(\boldsymbol{c}|\boldsymbol{z}_k)$, with $\boldsymbol{c}$ being the set of possible components. This element is also weighted by a tunable hyperparameter γ that modulates how closely the VAE must model the attention mask distribution.

$$\mathcal{L}\left(\phi,\theta,\psi,\boldsymbol{x}\right) = -\log\left(\sum_{k=1}^{K}\boldsymbol{m}_k p_\theta(\boldsymbol{x}|\boldsymbol{z}_k)\right) + \beta D_{KL}\left(\prod_{k=1}^{K} q_\phi(\boldsymbol{z}_k|\boldsymbol{x},\boldsymbol{m}_k)||p(\boldsymbol{z})\right) + \gamma D_{KL}\left(q_\phi(\boldsymbol{c}|\boldsymbol{x})||p_\theta(\boldsymbol{c}|\boldsymbol{z}_k)\right) \tag{4.7}$$

The loss weights used were $\beta = 0.5$, $\gamma = 0.5$, the same than Burgess et al.

4.2 MONet-RN50

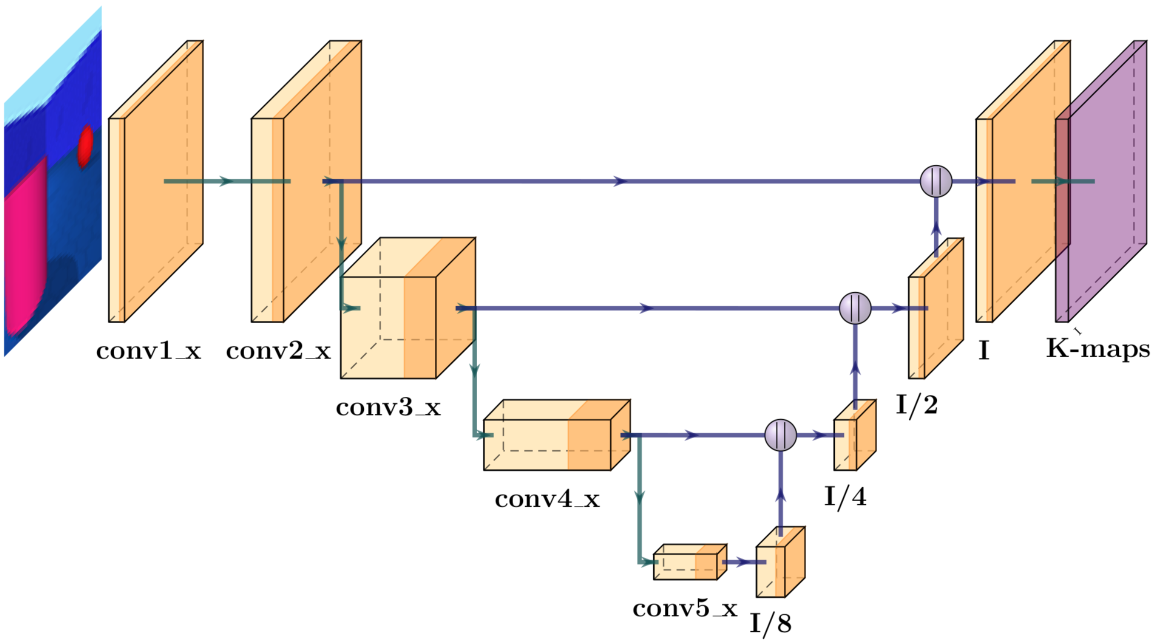


Figure 4.2: MONet-RN5* schematic. The encoder, left part of the image, is composed of the four blocks of ResNet [He et al., 2016]. The internal layers and connections of ResNet's blocks are not shown for simplicity. The decoder, right side, is made of four transposed convolutions. Each one of them doubles the size of its input feature vector, up to recover the original size of the input image **I**.

The iterative nature of the attention module correlates its computational cost with the number of components to be detected. This way, when the number of slots for which the model is trained increases, the computational cost will also increase. It is an $\mathcal{O}(n)$ dependence on the number of operations in both the forward and backward passes, with respect to the number of slots, which can be confirmed by the metrics measured in section 5.5. Furthermore, once the model has been trained for a given number of slots, even if the number can be increased at inference time, the models suffers from a drop in performance, as stated by Burges et al. [Burgess et al., 2019]. To reduce the computational cost of MONet, we propose to change the iterative attention module, for a segmentation network which directly predicts the attention mask for each slot. In this model the training cost will not increase linearly with the number of slots. Now increasing the number of slots can be done by simply changing the output of the last layer of the segmentation network.

To further reduce the computational cost, we propose to apply transfer learning. Instead of training the segmentation network from scratch, we use the weights of a pre-trained model. This will help to the convergence of the segmentation network in unsupervised small datasets; and accelerate the training process as it alredy learned in the supervised pre-training how to extract general features from the input image.

We denote our new model as MONet-RN50, implemented using a segmentation network as attention module. Our segmentation network follows an encoder-decoder architecture. As encoder we use a ResNet50 architecture without the last fully connected layer. Instead, the encoded representation generated by ResNet50 is fed to a decoder that generates K segmentation maps. The decoder is a set of 4 transposed convolutional layers followed by a convolution. Each transposed convolutional layer decreases the number of channels and increases the size of the image. Finally the convolutional layer generates an output with K channels, one per slot or disentangled component. Regarding the component VAE and the loss function we kept the same structure used by MONet.

Even if this model shows a great ability to learn representations, see Chapter 5, its straightforward architecture struggles to learn in datasets with small amounts of data (CLEVR), and the performance highly decreases. To tackle this problem we performed two modifications. First, we reduced the dimensionality of the decoder decreasing the number of output channels after each layer. Second, following [Ronneberger et al., 2015], skip-connections were added between the encoder (ResNet50) and decoder of the attention network. This improves the gradient flow and the model convergence, without greatly increasing the number of parameters nor the GPU RAM use, as will be shown in section 5.5. The improved network is called MONet-RN50*, see Figure 4.2.

Implementation details

The segmentation network is composed of a ResNet50 encoder and a decoder. The encoder keeps the same structure than the original ResNet50 architecture, except for the last fully-connected layer and the previous instance normalization. The decoder is made of four transposed convolutional layers with kernel 5×5, stride 2, input padding 2 and outpout padding 1, and bias true. The input of the first layer is the output of ResNet50 encoder, with 2048 channels. The number of channels of the output of each of the four layers is $(512, 256, 128, 64)$ respectively. Finally, the output of the decoder is passed through a final convolutional layer with kernel 1×1, stride 1, and K channels as output, and a sigomid activation function.

To add skip connections, ResNet50 architecture has been slightly modified to add to the output the intermediate tensors generated by each of its four main blocks. The dimensionality of the decoder has been decreased by reducing the number of output channels of each layer. Now the number of channels of the outputs is $(512, 128, 64, 16)$. Finally, the last convolutional layer keeps the same architecture than MONet-RN50.

4.3 Weights initialization

The first experiments performed were not able to conclude the training properly due to numerical errors. These experiments were performed using Pytorch's default configuration for weights initialization. By further revising the literature, we found that MONet authors used a special distribution to initialize the model, although they did not discern how did they choose some configuration parameters.

The weights of the developed models (mainly linear and 2D-convolutional layers) have been initialized using a truncated normal distribution $\mathcal{N}(\mu, \sigma, b_l, b_h)$ where μ is the mean, σ the standard deviation, and b_l and b_h the lower and higher bounds respectively, as described in MONet [Burgess et al., 2019]. This type of distribution avoids values near the tails, which saturate when passing trough a sigmoid activation function, leading to information losses.

To initialize the weights, the parameters of the distribution can be selected following different heuristics. Two of them are:

- An adaptation of Pytorch's default configuration [Torch Contributors, 2021a, Torch Contributors, 2021b].

- Kaiming initialization [He et al., 2015], an extension of Xavier initialization [Glorot and Bengio, 2010] for inputs that does not have zero mean (something that happens when using ReLu activation function).

Pytorch's default configuration initializes the layers with an uniform distribution $\mathcal{U}(-\sqrt{k}, \sqrt{k})$. For a linear layer

$$k = \frac{1}{in\_features},$$

with $in\_features$ the number of input features [Torch Contributors, 2021b]. For a convolutional layer

$$k = \frac{1}{\mathcal{C}_{in} * \prod_{i=0}^{1} ker\_size[i]},$$

with $\mathcal{C}_{in}$ being the number of input channels, and ker_size the kernel size [Torch Contributors, 2021a]. These initializations are adapted to the truncated normal distribution $\mathcal{N}(0, \sqrt{k/3}, -\sqrt{k}, \sqrt{k})$ keeping the same bounds and mean, and selecting $\sqrt{3}\sigma = \sqrt{k} \rightarrow \sigma = \sqrt{k/3}$ to ensure that 90% of the population falls inside the distribution, and only the tails are cut out. Kaiming initialization differs in that it computes

$$k = \frac{2}{n_{fout}},$$

for both convolutional and linear layers, with n_{fout} the number of output features of the given layer.

The authors of MONet do not clarify which heuristic do they use. We performed a study to evaluate them and chose the one with the best performance.

Chapter 5

Results

The objective of this chapter is to explain and analyze the different experiments that have been performed to evaluate the developed models and algorithms. First the datasets used on the evaluation are presented, and the metrics used to quantify the performance are explained. After that, the evaluation of the performance on each of the main datasets is performed. Then, an analysis on the computational footprint is performed. And finally, a brief analysis on the more realistic datset is done.

5.1 Datasets

The developed models have been evaluated on three datasets of different complexity: *Objects Room*; *CLEVR*; and *SceneNet*.

Objects Room [Burgess et al., 2019] is a multi-object extension of the the 3d-shapes dataset [Burgess and Kim, 2018] on the synthetic MuJoCo environment. It consist of 64×64 RGB static scene images of a cubic room with coloured floor, walls and a number of objects visible. Above the wall a blue sky is visible. The position of the camera is randomly sampled on a ring inside the room, always facing towards the center, but with the vertical orientation sampled from a uniform distribution in $(-25°, 22°)$. All the scene elements except for the sky are randomly coloured, with each colour uniformly sampled in HSV colour-space, with H in $(0, 1)$, S in $(0.75, 1)$ and V always in 1.0. The objects are randomly shaped (six different shapes), sized, and arranged around the room (avoiding complete overlap). The main set is composed of 1 million images with 3 objects in the room, making $1 - 3$ objects visible. This set is split in three subsets for training, validation, and test; each made of 980,000, 10,000, and 10,000 images respectively. To further evaluate the generalization capability of the model a second test set is available, *Same colour*, for which there are between 4 and 6 objects all of them of the same colour, has been used. Some samples of both the standard dataset and the *Same colour* dataset can be seen in Figure 5.1.

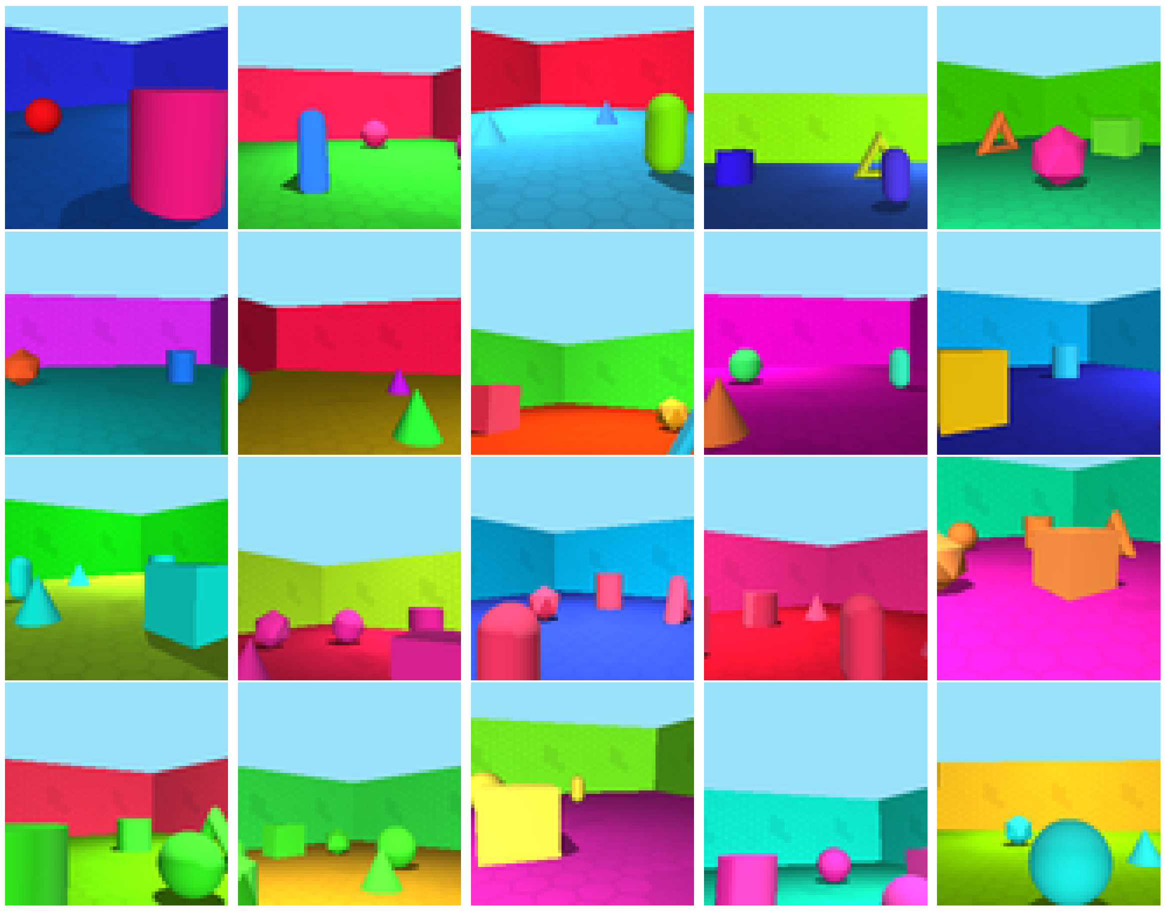


Figure 5.1: Samples of the *Objects room* dataset. The first two rows are the standard dataset, and the third and fourth are the *Same Colour* test set.

CLEVR with masks is a version of the CLEVR dataset [Johnson et al., 2017] with segmentation mask annotations. It is a dataset of simple 3D rendered objects, with more complex colours than *Objects Room* as it represents different materials. It contains 90,000 images (70,000 for the training sets, and 10,000 each for the validation and test sets) with a resolution of 240×320 pixels. To ensure that most of the frame was occupied by objects, the images are croped left and right at the x-coordinate (29,221) and y-coordinate (64,256) to create 192×192 images, with between 3 and 10 visible objects per image. These are then resized using bilinear interpolation to 128×128. The objects are characterized in terms of shape (cube, cylinder, or sphere), size (small or large), material (rubber or metal), colour (8 different colours: red, cyan, green, blue, brown, gray, purple, yellow), position (continuous) and rotation (continuous). Some samples of the normal dataset can be seen in Figure 5.2.

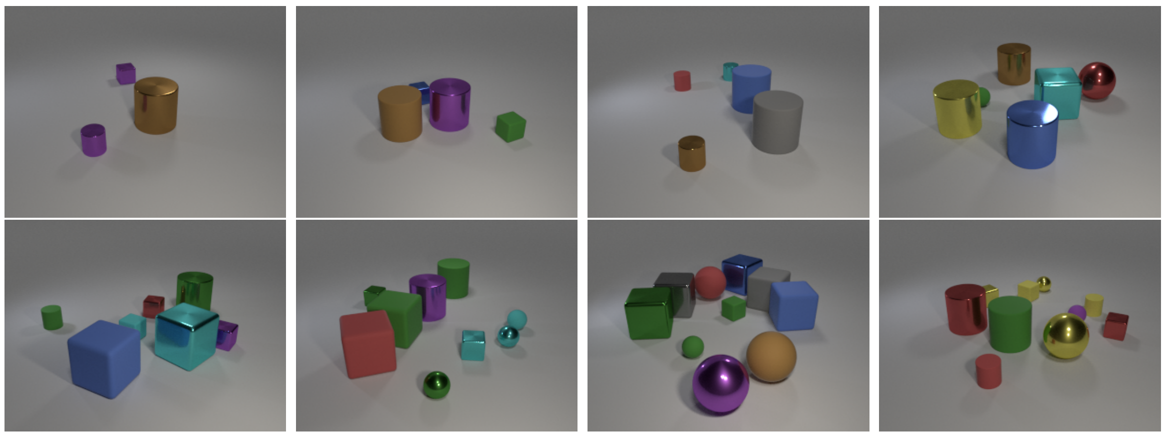


Figure 5.2: Samples of the *CLEVR* dataset. The images are ordered from left to right, up to down, in increasing number of visible object, except for the last two which both have ten elements. The different factors of variation can be observed: position, orientation, colour, shape, size, material, and number of visible elements.

SceneNet [McCormac et al., 2017] is a dataset composed of 5 Million rendered photorealistic RGB-D images from over 15K trajectories in synthetic layouts with physically simulated poses. Each layout also has random lighting, camera trajectories, and textures. For our experiments we used a subset of 300,000 randomly selected images. Like *CLEVR* the images of an original resolution of 240×320 are croped and resized to create 192×192 images.

5.2 Metrics

The models are evaluated using different metrics for segmentation, image reconstruction, and computational footprint. Despite there being different metrics for evaluating the disentanglement (Beta-Vae [Higgins et al., 2017], Factor-VAE [Kim and Mnih, 2018], Mutual Information Gap [Chen et al., 2018], Modularity [Ridgeway and Mozer, 2018] and DCI [Eastwood and Williams, 2018]) all of them have been designed for supervised evaluation. Therefore, to evaluate disentanglement we perform only qualitative analysis based on visual inspection of the images as is usual on the field [Higgins et al., 2017, Burgess et al., 2019, Greff et al., 2019, Kabra et al., 2021].

Segmentation. To quantify segmentation quality the Adjusted Rand Index (ARI) [Rand, 1971, Hubert and Arabie, 1985], and the Mean Segmentation Covering (MSC) [Arbelaez et al., 2010, Engelcke et al., 2019] have been used. *ARI* is a measure of clustering similarity that ranges from 0 to 1 (being 1 a perfect clustering), and can handle arbitrary permutations. It is applied as a measure of instance segmentation quality by treating each foreground pixel as one point and its segmentation as cluster assignment. *MSC* is an unweighted variation of the *segmentation covering* metric [Arbelaez et al., 2010]. It is based on the Intersection Over Union (IOU) between pair of two masks S and S', with S being the segmentation masks of the ground truth foreground, and S' the predicted segmentation masks. The covering of S by S' is defined as

$$C(S' \to S) = \frac{1}{|S|} \sum_{R \in S} \underset{R' \in S'}{max} \, \mathrm{IOU}(R, R'),$$

where $|S|$ denotes the number of non empty masks in S. It is important to note, that unlike the ARI this metrics does penalize the over-segmentation of ground truth objects as this decreases the IOU for a pair of masks.

For ease of visualization, both *ARI* and *MSC* are expressed in percentage.

Reconstruction. To evaluate the ability to reconstruct the scenes given the latent factors the Mean Squared reconstruction Error (MSE) [Greff et al., 2019] has been used. *MSE* evaluates the ability of a model to reconstruct an image computing the per-pixel mean squared error between the original image and the reconstruction.

Computational efficiency. Finally the computational cost will be evaluated in terms of Float point Operations (FLOP) in the forward pass, execution time, number of parameters of the models and GPU RAM use. The FLOP are not computed for the backward pass due to the lack of tools to measure that metric.

5.3 Objects Room

First, the two configurations for weights initialization have been evaluated on the validation set. The objective of this experiment was to discern if there was a significant difference between them. The two initialization heuristics, have been trained both on MONet and MONet-RN50 (our first model developed) to minimize the model's bias on the results. After a configuration has been chosen, the three models have been trained on *Objects Room*'s, and evaluated on the different test sets available. For the comparisons two other models of the state of the art are used: Genesis v2, and Slot Attention. While these models reported metrics for ARI and MSC, MONet reported only qualitative analysis which motivates to fully train the model to evaluate it properly.

Weights ablation.

Both MONet and MONet-RN50 have been trained on *Objects Room*'s training set, and evaluated on the validation set, to compare the two different initialization algorithms and its benefits. Both models have been trained with both initializations for 33 epochs with a batch size of 64 images, equivalent to 0.5×10^6 training iterations. The training process has been repeated with four different random seeds to minimize the aleatoric impact on the metrics.

Evaluating the models on the validation set, the quantitative evaluation in Table 5.1 shows that despite our initial belief that Kaiming initialization would improve convergence, there is not a statistical significant difference. Both initializations achieve similar performance on two of the three proposed metrics: *ARI*, *MSC*, and *MSE*.

However, comparing the metrics for the two architectures, **our MONet-RN50 clearly has a better average performance than the baseline MONet** for *ARI* and *MSE*, and a similar performance for *MSC*. Despite that, the standard deviation between the different runs is so high that there is not margin to state that it is clearly better. Finally, a qualitative analysis of the generated images and masks, Figure 5.3 and appendix A.1, shows that even if the images generated by the different models look alike, MONet-RN50 achieves better representations. By better, we mean that it learns more disentangled representations, and does not rely so much on the segmentation masks to reconstruct an element. It can clearly be seen for the generation of the blue triangle with a hole in Figure 5.3: all the models are able to generate a triangle with a hole once the mask has been applied, although MONet-RN50 with Kaiming initialization (S4, last column) is the only one that had already generated a triangle with a hole, without the mask; the other model had only generated triangular forms, or even a blue coloured zone, without really decoding the properties of the triangle.

Architecture	**Initialization**	**ARI**	**MSC**	**MSE** $\times 10^{-4}$
MONet	Default	78.20±9.55	75.38±4.92	11.221±4.158
	Kaiming	79.16±4.03	73.71±3.92	18.401±11.863
MONet-RN50	Default	**84.06**±2.40	68.99±16.46	**7.212**±0.813
	Kaiming	83.78±2.00	**76.34**±10.77	8.234±2.502
		↑ Bigger better	↑ Bigger better	↓ Smaller better

Table 5.1: Comparison of the weights initialization impact on the reconstruction and segmentation performance of MONet and MONet-RN50.

Given this qualitative superiority that can be appreciated, see appendix A.1 for further images, even if there is a lack of empirical metric evidence about Kaiming heuristic being better, we have decided that its theoretical bases is enough to use it to initialize the different models that have been trained further on.

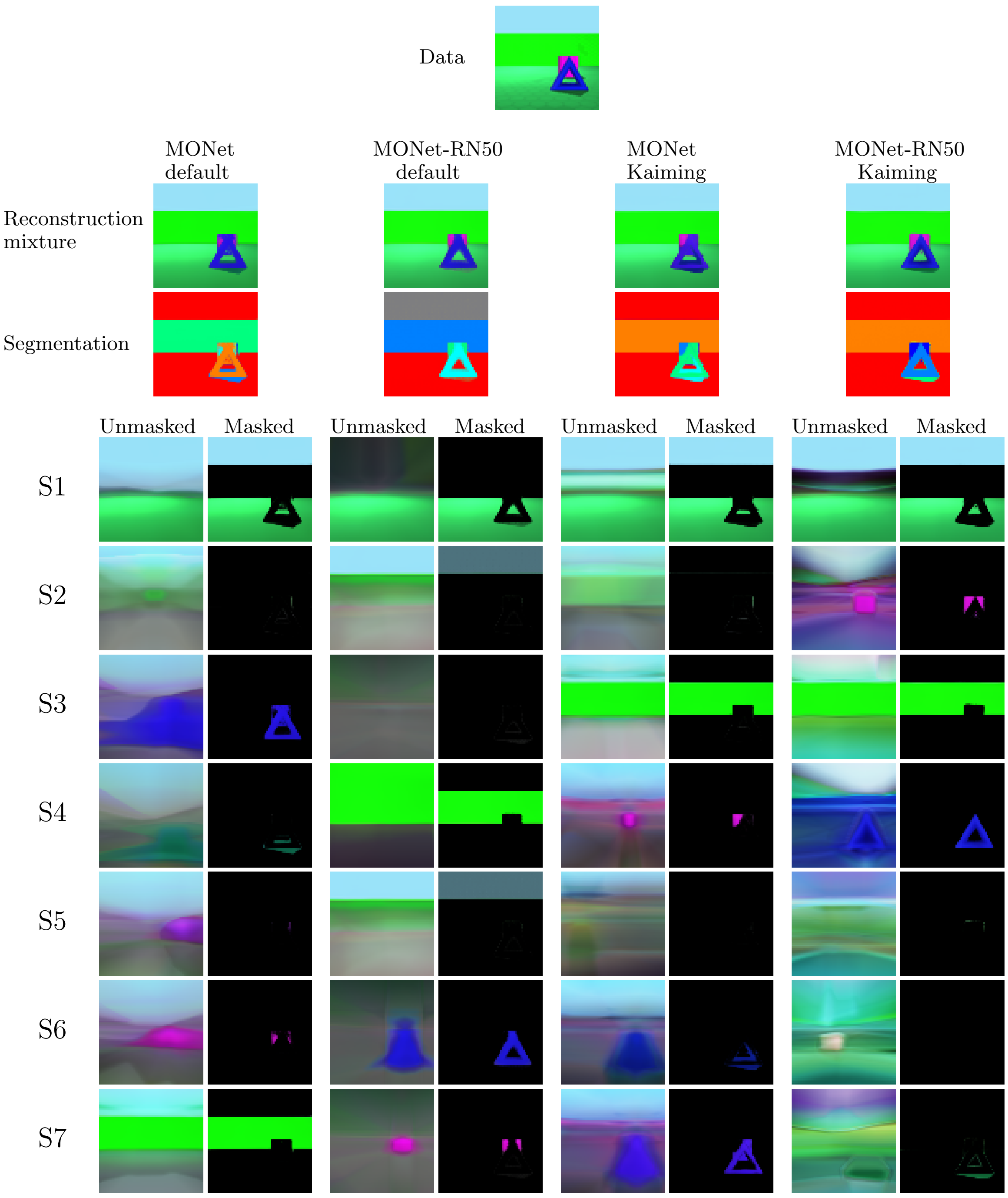


Figure 5.3: Decomposition results on *Objects Room* comparison. Results from MONet and MONet-RN50 initialized with Pytorch's default algorithm and with Kaiming et al's algorithm. Each example shows the output of a different configuration. *Reconstruction mixture* shows the sum of components from all slots, weighted by the learned masks from the attention network. *Segmentation* shows a colour-coded map summarising the attention masks m_k. Rows labelled S1-S7 shows the reconstruction components of each slot. Unmasked versions are shown side-by-side with corresponding versions that are masked with the VAE's reconstructed masks $\tilde{m}_k$

Performance comparison

After the ablation, the three models have been trained up to 66 epochs with a batch size of 64 images, equivalent to 1×10^6 training iterations. This time, the training process has been repeated with three different random seeds to minimize the aleatoric impact on the metrics. The resulting models have been evaluated on Objects Room test set, and on the set Same Colour of Objects Room, where all the objects are of the same colour.

On the general test set, **it can be seen how our MONet-RN50* clearly outperforms all the other models of the state of the art**, see Table 5.2. MONet-RN50 has a good performance on ARI but with a fall in MSC and MSE. This means that even if MONet-RN50 generates good segmentation masks, it is prone to oversegment the scene elements (captured by MSC), and is not as good as the other models on generating a reconstruction (captured by the high MSE). Instead, MONet-RN50* is able to achieve high values for both ARI and MSC, meaning that the masks fit more accurately the objects without oversegmenting. Furthermore, having the lowest MSE implies that MONet-RN50* reconstructions have the highest quality between the three trained models (the only ones that report this metric). As a side note, it can be seen how MONet-RN50* not only has good metrics, but also keeps a low value for the standard deviation, which shows its better convergence.

Model	**ARI**	**MSC**	**MSE** $\times 10^{-3}$
MONet	83.44±1.58	80.77±0.64	0.712±0.153
MONet-RN50	85.78±0.77	76.40±13.48	0.819±0.325
MONet-RN50*	**86.88±1.23**	**84.10±0.73**	**0.517±0.042**
Genesis v2	85±1	59±1	–
Slot Attention	79±2	64±13	–
	↑ Bigger better	↑ Bigger better	↓ Smaller better

Table 5.2: Segmentation metrics on objects room test set. Genesis v2 and Slot Attention do not report the MSE metric.

On the *Same colour* test set, MONet-RN50* still outperforms the other methods although by a smaller margin (see Table 5.3). Its performance here is very similar to MONet's, and even MONet tends to oversegment less, which is shown in its highest MSC (although with a really high standard deviation). Note that Genesis v2 and Slot Attention do not report metrics on this set.

Disentanglement. Even if our objective was to implement a model able to generate disentangled representations, the developed work has been centered on the Attention module and not on the Component module which is in charge of disentangling the

Model	**ARI**	**MSC**	**MSE** $\times 10^{-3}$
MONet	78.83±2.53	**56.99±5.60**	6.162±0.417
MONet-RN50	80.95±1.70	51.74±8.84	7.639±0.127
MONet-RN50*	**81.58±1.30**	55.61±1.34	**6.155±0.896**
	↑ Bigger better	↑ Bigger better	↓ Smaller better

Table 5.3: Segmentation and reconstruction metrics on *Objects Room Same Colour* test set.

properties of the elements. The changes performed were directed towards improving the segmentation, and as a result, the convergence and efficiency of the model. They should not have affected the Component VAE learning and the overall ability of the model to disentangle properties.

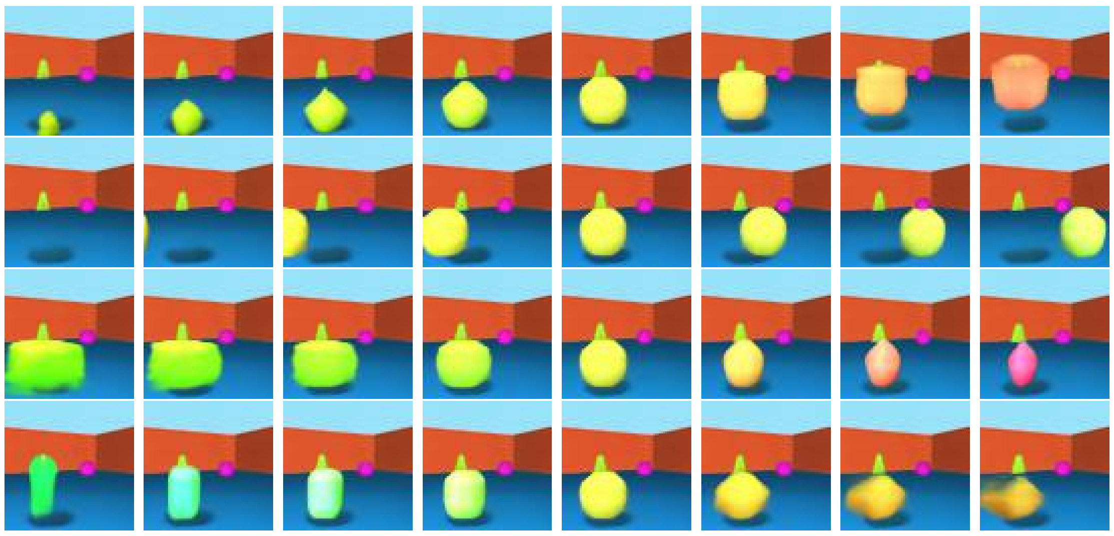


Figure 5.4: Disentanglement example of MONet on *Objects Room*. Sample of the fifth slot of the processed image. Each row shows a different latent variable that can be associated with a property of the element segmented on the fifth slot. From up to down: size (z_{5-2}); horizontal position (z_{5-3}); vertical position (z_{5-10}); and shape (z_{5-13}).

This can be confirmed through a qualitative analysis using visual inspection. A randomly sampled image has been fed to the best seed of each of the three architectures. Then its latent vectors have been modified for several values to see how it affects the final image. After that, a visual inspection of the properties affected has been performed. The comparison is done searching which latent represents a property of the image, and analyzing how the property evolves for its different values. Figures 5.4, 5.6 and 5.5 show the latent variables that better encode a property of the represented element, for each of the three models. All the models are able to learn to more or less disentangle the size, and horizontal position (first and second rows respectively). MONet-RN50 and MONet-RN50* are also able to learn the vertical position (third row of Figure 5.6 and 5.5), while none of the latent dimensions of MONet encodes

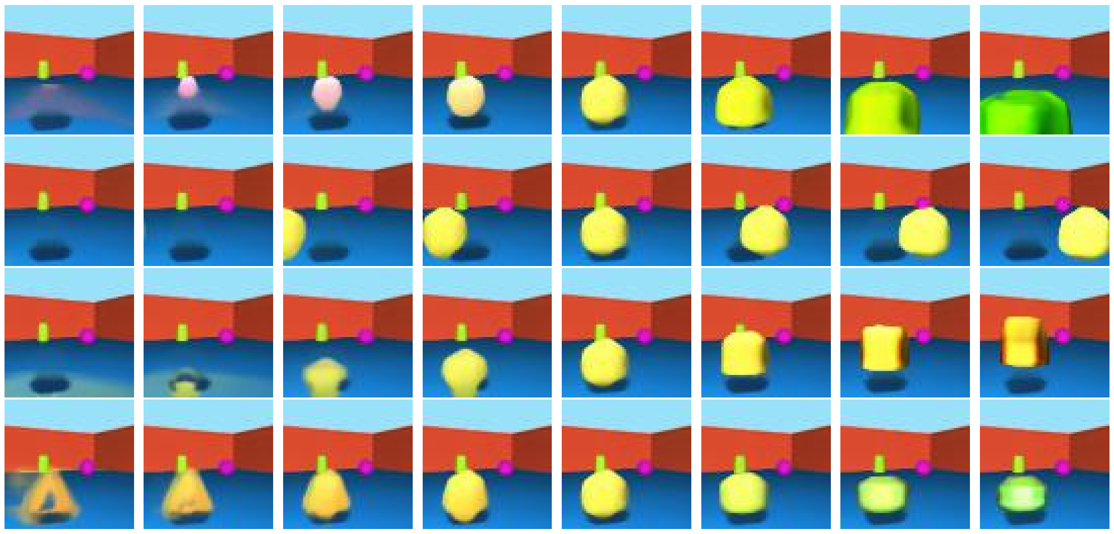


Figure 5.5: Disentanglement example of MONet-RN50 on *Objects Room*. Sample of the fifth slot of the processed image. Each row shows a different latent variable that can be associated with a property of the element segmented on the fifth slot. From up to down: size (z_{5-12}); horizontal position (z_{5-4}); vertical position (z_{5-5}); and shape (z_{5-10}).

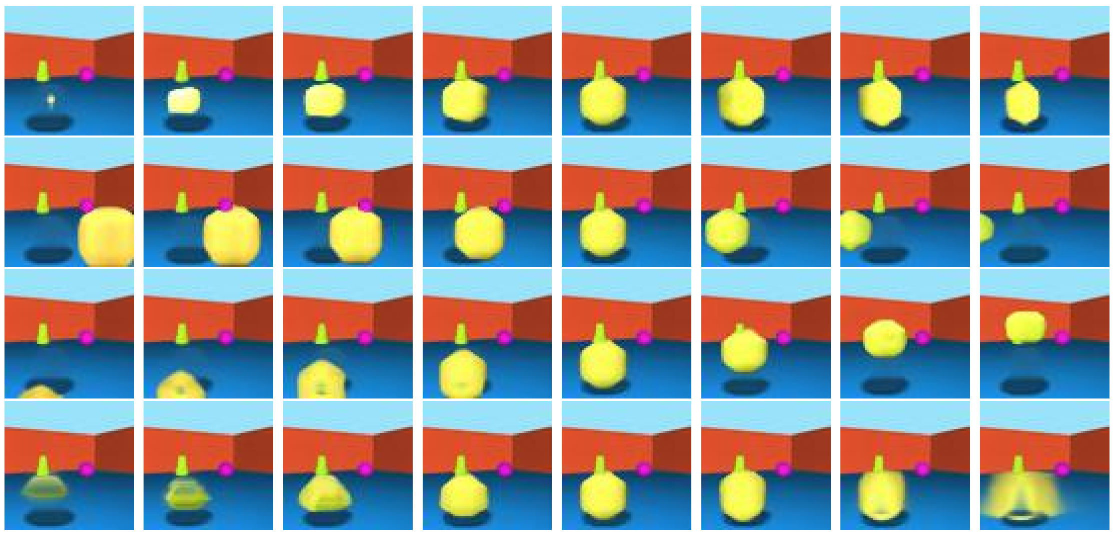


Figure 5.6: Disentanglement example of MONet-RN50* on *Objects Room*. Sample of the second slot of the processed image. Each row shows a different latent variable that can be associated with a property of the element segmented on the second slot. From up to down: size (z_{2-13}); horizontal position (z_{2-11}); vertical position (z_{2-10}); and shape (z_{2-12}).

that property. Finally, all the models have somehow encoded the shape of the element (fourth row), although this property has not been disentangled at all. Not only it can be modified through different latent variables (none of disentangled examples keeps exactly the same shape), but also when it is modified it affects another property like the colour.

Despite the differences on the way that each architecture disentangles each property, our analysis suggests that all the architectures have a similar performance, as was expected. Nevertheless, it cannot be clearly said without a quantitative evaluation.

5.4 CLEVR

To further evaluate the ability of the developed models on more complex scenarios, they have been trained and tested on *CLEVR*. This dataset not only has a more complex environment with more elements, different shapes, and more realistic colours, but it is also composed of far less data (only 70 $\times 10^3$ training samples against the 980×10^3 of Objects Room. Following MONet's implementation [Burgess et al., 2019], its attention module has been extended adding an extra block, as explained at the end of the Implementation details in Section 4.1.1. The models have been trained with a batch size of 32 images, and for 229 epochs, the equivalent to 0.5×10^6 training iterations on *CLEVR* dataset.

Convergence

The small size of this dataset makes convergence of the models far more difficult. When training on this more complex dataset, a new phenomenon can be observed: depending on the random seed, both MONet and MONet-RN50 models can converge to local minima where a single slot segments all the elements, and the rest of the slots do not segment anything except for the background, see Figures 5.7 and 5.8. This phenomenon does not affect MONet-RN50* which converges to a similar local minimum despite changing the random seed for initializing the weights.

Segmenting all the elements together does not allow the component VAE to learn complex features as it forces the given slot to represent the features of all the objects segmented. It generates a mixture of colours and completely relies on the generated mask to shape the learnt objects. Depending on how spatially close the objects are from one another and how similar their colours are the final representation will look more or less accurate.

Changing the random initialization of the model improves MONet convergence but not MONet-RN50. MONet is able to converge to a different solution where, usually, each slot explains a different object, see Figure 5.9. Segmenting the objects individually allows the model to learn the shape property of the different objects, which can be seen on how it is able to learn and generate whole objects, and then use the mask to occlude such parts that are not seen in the final image.

On the other side, MONet-RN50* has a steadier performance that is not so much affected by the randomness. Nevertheless, the lack of training data affects its ability

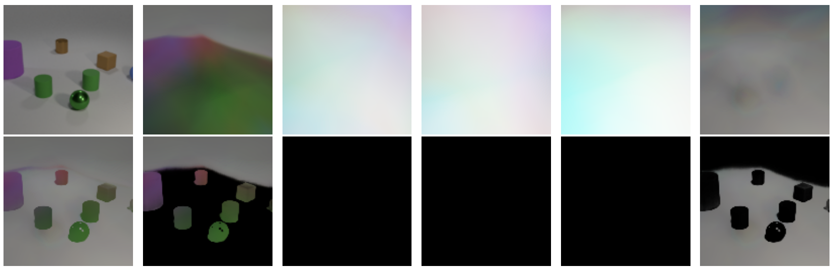


Figure 5.7: Generated images of MONet on a local minimum on *CLEVR*. The first column from the left shows the input image (up) and reconstruction mixture (down). The next of the columns show the pair of unmasked (up) and masked (down) generated images. From left to right the slots are: first, second, third, fourth, and last. The rest of the slots are note shown because the do not provide further information.

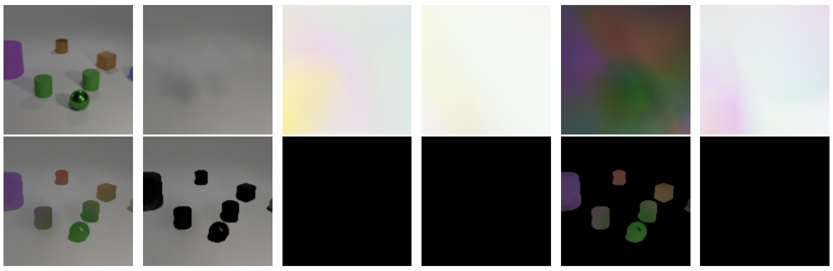


Figure 5.8: Generated images of MONet-RN50 on a local minimum on *CLEVR*. The first column from the left shows the input image (up) and reconstruction mixture (down). The next of the columns show the pair of unmasked (up) and masked (down) generated images. From left to right the slots are: first, second, third, fourth, and last. The rest of the slots are note shown because they do not provide further information.

to converge to a solution where each slot segments a single element, as can be seen of Figure 5.9. Instead, MONet-RN50* converges to a solution where multiple objects can be explained on the same slot (slot 8 of the figure) and multiple slots explain the same object (slots 5 and 11 of the figure). This phenomenon also affects MONet, but with a lower impact , as can be seen by the fact that even if a slots can attempt to explain multiple objects (slot 7 of Figure 5.9) each object is explained only by one slot. This analysis is supported by further images in Appendix B.1. Despite that, it is interesting to see how both models are still able to learn good representations of the different elements, Figure 5.9, even if they do not disentangle the properties as good as in *Objects Room*. This can be seen in Figures 5.11 and 5.10 where some latent variables are analyzed. Both models learn to modify some properties like shape, size or colour, but are unable to modify only one at the time. The rest of the dimensions can be seen in Appendix B.2.

The better convergence of MONet-RN50* is reflected on the metrics of Table 5.4, which show how the model achieves a great improvement on the average performance with a lower standard deviation. However, the standard deviation of MONet, due to its convergence instability, is too high to draw solid conclusions. A far greater sample of trained models would be needed, although that implies a computational effort outside of the scope of this Master Thesis.

Model	**ARI**	**MSC**	**MSE** $\times 10^{-3}$
MONet	52.56±26.56	40.11±33.79	1.505±0.461
MONet-RN50*	69.32±7.70	56.46±8.70	0.630±0.081
	↑ Bigger better	↑ Bigger better	↓ Smaller better

Table 5.4: Segmentation and reconstruction metrics on of MONet and MONet-RN50* on *CLEVR* test set. Three different random seeds have been trained.

5.5 Efficiency comparison

The computational cost of MONet and MONet-RN50* has been evaluated on *Objects Room.* Two aspects have been evaluated: the execution and energy cost, measured in FLOP and time; and the memory footprint measured in GBytes of GPU RAM used.

First, for a batch of 64 images, the execution time per epoch and number of FLOP in the forward pass have been computed. To evaluate the dependency of the performance with respect to the number of slots the measures have been taken for four different configurations: 7, 11, 14, and 18. Due to the fact that both architectures use the same component module, the number of FLOP does not represent the total FLOP of the whole architecture, but rather only accounts for the attention module.

The results of the evaluation can be seen in Figure 5.12. On the left plot, it can be seen how for MONet the number of FLOP has a linear dependency with respect to the number of slots. Instead for MONet-RN50* the number of FLOP is almost constant. This is thanks to the fact that increasing the number of slots only implies changing the last layer of the attention network with a negligible effect on the overall network complexity. Regarding the execution time, right plot, now MONet-RN50* also increases linearly due to the small dependency of the component module on the number of slots. Nevertheless, the execution time of MONet is 30% smaller for 7 slots and 60% for 18 slots, which represents and important speed-up on both the training and inference time.

Finally, the evaluation of the memory footprint has been done for a single number of slots. This is because the architectures are designed to use the same networks to process the different slots. Therefore neither of the models has a dependency on the number of slots: MONet has a constant number of parameter; and MONet-RN50 almost constant due to the negligible effect of the last layer. The results of the evaluation, Table 5.5, show how despite MONet-RN50* having almost five times more parameters, its memory footprint is smaller than MONet at training time, allowing to train bigger batches and therefore further decrease the training time. This is a

Model	**Number of parameters (in millions)**	**GPU RAM (GBytes)**	
		Training	Inference
MONet	13.926	3.80	1.69
MONet-RN50	54.024	4.17	2.16
MONet-RN50*	55.791	2.59	2.17

Table 5.5: Comparison of the model size and memory use of MONet and MONet-RN50 for a batch size of 1 image. The number of parameters reports only the size of the Attention module. The GPU memory use is split between training and inference.

consequence of the big computational graph generated to compute the gradient of the iterative algorithm on MONet. This graph is computed only on training, and therefore when performing inference MONet does have a lower GPU RAM use. Furthermore, it also shows how MONet-RN50* not only has almost the same number of parameters than MONet-RN50, but it even has a lower GPU RAM use. This lower use is a direct consequence of reducing the dimensionality of the decoder's layers.

5.6 SceneNet

Finally, to evaluate the potential to apply the developed model on more realistic data, an evaluation has been performed on a subset of *SceneNet*.

The cualitative results of the training, see Figure 5.13, show how MONet-RN50* is able to learn to generate decent reconstructions. However, note that the model has learn to segment based on the colour, without any further disentanglement. This is mainly due to high complexity of the dataset, where each type of element is seen only once in most cases. The lack of structured patterns does not allow the model to learn common information nor properties.

Therefore, to be able to implement disentangled representation learning algorithms on real scenarios, a structured dataset where an element is present multiple times with different properties is essential.

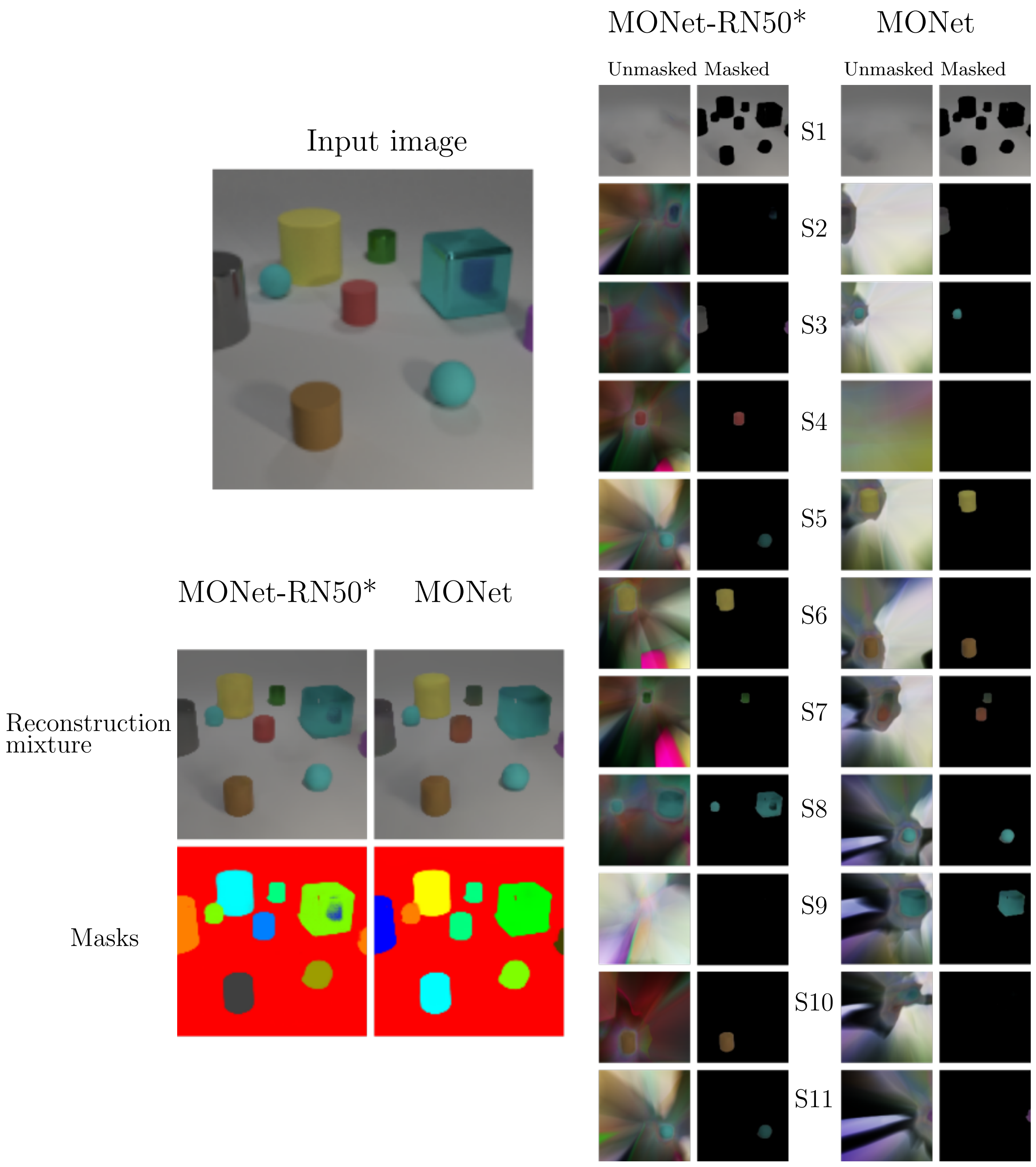


Figure 5.9: Decomposition results on *CLEVR* dataset comparison. Results from MONet and MONet-RN50*. *Reconstruction mixture* shows the sum of components from all slots, weighted by the learned masks from the attention network. *Segmentation* shows a colour-coded map summarising the attention masks m_k. Rows labelled S1-S11 shows the reconstruction components of each slot. Unmasked versions are shown side-by-side with corresponding versions that are masked with the VAE's reconstructed masks $\tilde{m}_k$

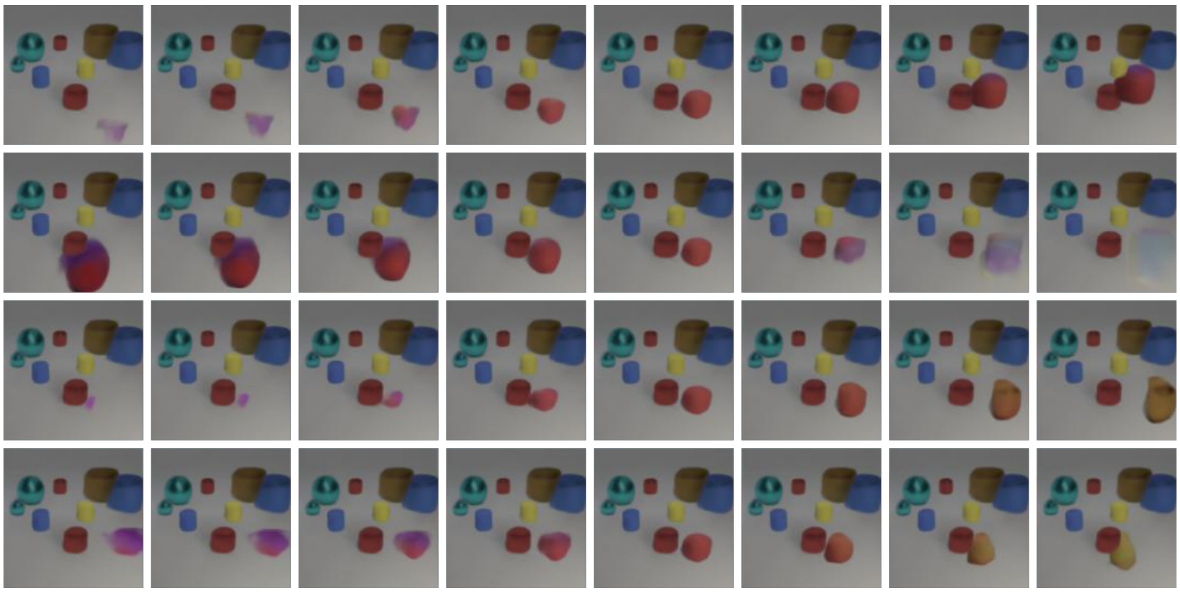


Figure 5.10: Disentanglement evaluation on MONet-RN50* on *CLEVR*. From up to down all the latent factors of slot 9, right red cilinder, for z_{9-1}, z_{9-7}, z_{9-10}, and z_{9-14},

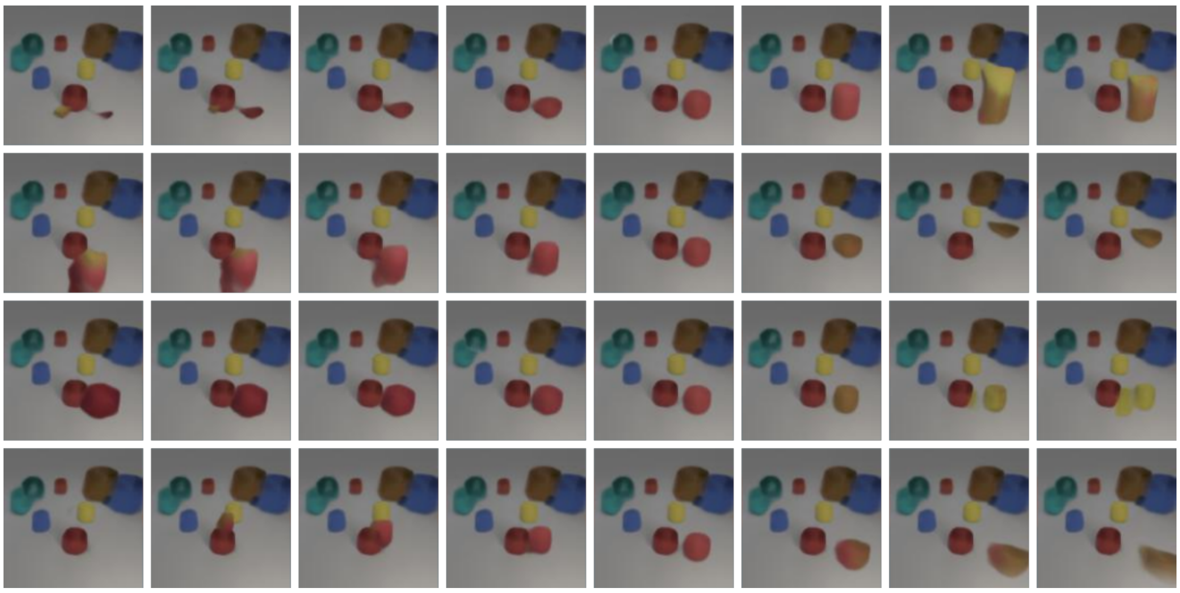


Figure 5.11: Disentanglement evaluation on MONet-RN50* on *CLEVR*. From up to down all the latent factors of slot 9, right red cilinder, for z_{9-0}, z_{9-1}, z_{9-5}, and z_{9-10},

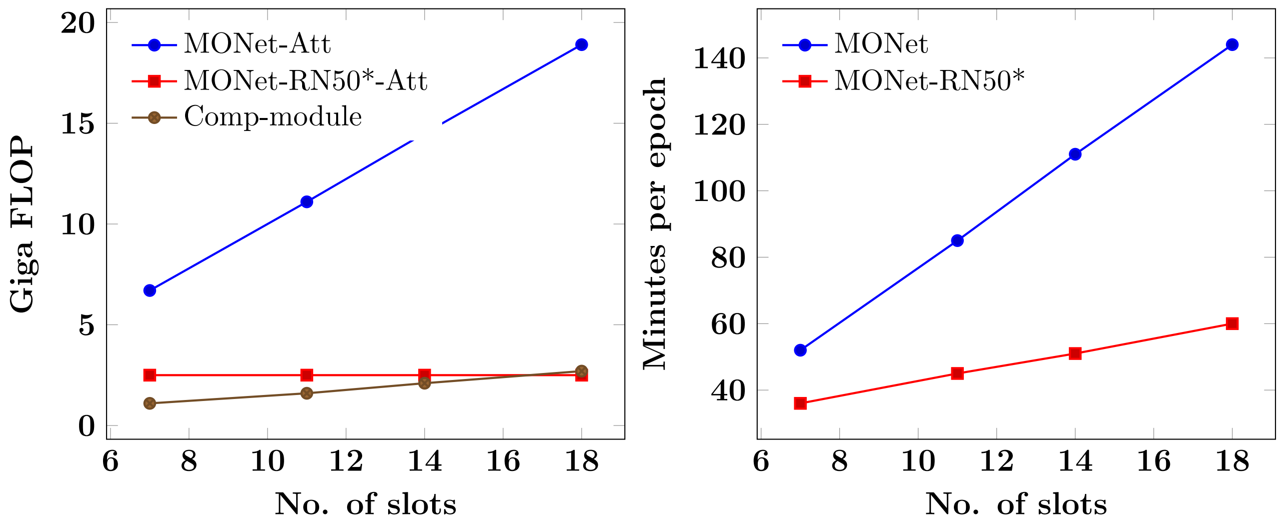


Figure 5.12: Comparison of the computational cost in terms of FLOP and time. Left: Comparison of FLOP per number of slots of the attention modules (MONet-Att and MONet-RN50*-Att) and the component module (Comp-module). Right: Comparison of the training time per number of slots of MONet and MONet-RN50* architectures. The training time is reported in in minutes per epoch. The FLOP have been measure with a batch size of 1 image, while the time with a batch of 64 images.

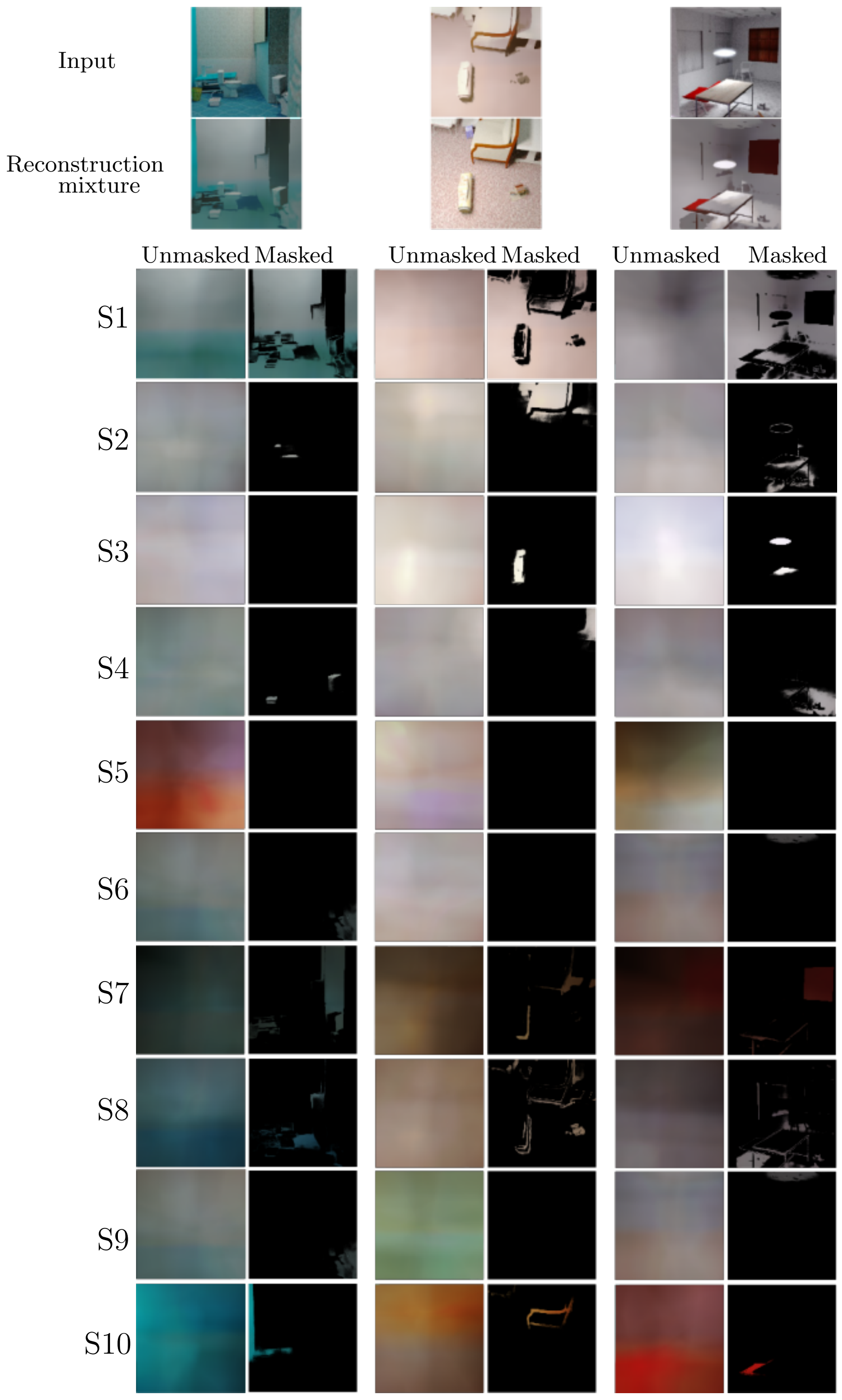


Figure 5.13: Segmentation examples of MONet-RN50* on *SceneNet*. Each example shows the output of a different configuration. Reconstruction mixture shows the sum of components from all slots, weighted by the learned masks from the attention network. Rows labelled S1-S7 shows the reconstruction components of each slot. Unmasked versions are shown side-by-side with corresponding versions that are masked with the VAE's reconstructed masks $\tilde{m}_k$

Chapter 6

Conclusions

In this master thesis it has been proposed a novel model, based on the MONet architecture, for disentangled representations learning. The modification is based on replacing MONet's iterative segmentation algorithm for an encoder-decoder pre-trained segmentation network, and profit from transfer learning when training overparametrized neural models without supervision. The experimental results have shown that, when enough data is available, our model MONet-RN50* improves MONet's performance for learning and reconstructing scenes, and matches its performance to disentangle scenes properties. In addition, our model has a smaller computational footprint, reducing the training time by 30% for the basic configuration, and even up to 60% for more complex ones. Despite this, it has been shown that for small datasets the model is not able to properly learn segmentation maps to disentangle the elements of the scene.

Regarding future work, there are several future lines that can be explored. All developed algorithms share a limitation on the number of objects that can be detected: there is always a maximum set before starting training. We have started working on developing an algorithm to detect a variable number of elements on a scene with an unsupervised segmentation, which would remove the necessity of training new models with higher maxima for each different dataset. From a theoretical viewpoint it is essential to come up with a formal and unified definition of disentanglement, which could be based on Information Theory. Complementary, it will be important to develop metrics for unsupervised disentanglement learning, given the inability of segmentation metrics to capture disentanglement. Finally, to really profit from disentangled representations learning, it would be needed to port the models studied in this master thesis to more realistic setups. To be able to learn the properties, the new training data should be structured enough for the model to see different combinations of the elements properties but with some common factors to extrapolate information. For example, if the model only sees an object with the same colour during training, it is impossible for it to learn to dissociate colour from shape.

Bibliography

[Ali Eslami et al., 2016] Ali Eslami, S. M., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, pages 3233–3241.

[Anciukevicius et al., 2020] Anciukevicius, T., Lampert, C. H., and Henderson, P. (2020). Object-Centric Image Generation with Factored Depths, Locations, and Appearances.

[Arbelaez et al., 2010] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). Contour Detection and Hierarchical Image Segmentation. Technical Report UCB/EECS-2010-17, EECS Department, University of California, Berkeley.

[Bermudez Contreras et al., 2020] Bermudez Contreras, E., Clark, B., and Wilber, A. (2020). The neuroscience of spatial navigation and the relationship to artificial intelligence. *Frontiers in Computational Neuroscience*, 14:63.

[Bourlard and Kamp, 1988] Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294.

[Burgess and Kim, 2018] Burgess, C. and Kim, H. (2018). 3d shapes dataset. `https://github.com/deepmind/3dshapes-dataset/`.

[Burgess et al., 2018] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. (2018). Understanding disentangling in β-vae.

[Burgess et al., 2019] Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation.

[Cadena et al., 2016] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

[Cahill et al., 2013] Cahill, A. E., Aiello-Lammens, M. E., Fisher-Reid, M. C., Hua, X., Karanewsky, C. J., Ryu, H. Y., Sbeglia, G. C., Spagnolo, F., Waldron, J. B., Warsi, O., and Wiens, J. J. (2013). How does climate change cause extinction? *Proceedings. Biological sciences*, 280(1750):20121890.

[Chen et al., 2018] Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*.

[Cowls et al., 2021] Cowls, J., Tsamados, A., Taddeo, M., and Floridi, L. (2021). The ai gambit—leveraging artificial intelligence to combat climate change: Opportunities, challenges, and recommendations. *Challenges, and Recommendations (March 15, 2021)*.

[Crawford and Pineau, 2019] Crawford, E. and Pineau, J. (2019). Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks Background : Variational Autoencoders.

[Dhar, 2020] Dhar, P. (2020). The carbon impact of artificial intelligence. *Nature Machine Intelligence*, 2(8):423–425.

[Do and Tran, 2020] Do, K. and Tran, T. (2020). Theory and evaluation metrics for learning disentangled representations. In *ICLR*.

[Eastwood and Williams, 2018] Eastwood, C. and Williams, C. K. I. (2018). A framework for the quantitative evaluation of disentangled representations. In *ICLR*.

[Engelcke et al., 2021] Engelcke, M., Jones, O. P., and Posner, I. (2021). GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement.

[Engelcke et al., 2019] Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2019). GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. pages 1–17.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

[Goodfellow et al., 2016] Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. `http://www.deeplearningbook.org`.

[Greff et al., 2019] Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:4317–4343.

[Hardy, 2003] Hardy, J. T. (2003). *Climate change: causes, effects, and solutions*. John Wiley & Sons.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Higgins et al., 2018] Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a Definition of Disentangled Representations. pages 1–29.

[Higgins et al., 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). β-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*.

[Hinton and Zemel, 1993] Hinton, G. E. and Zemel, R. S. (1993). Autoencoders, minimum description length and helmholtz free energy. In *NIPS*.

[Hubert and Arabie, 1985] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

[Jiang et al., 2020] Jiang, J., Janghorbani, S., de Melo, G., and Ahn, S. (2020). SCALOR: Generative World Models with Scalable Object Representations. pages 1–22.

[Johnson et al., 2017] Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, pages 1988–1997. IEEE Computer Society.

[Kabra et al., 2021] Kabra, R., Zoran, D., Erdogan, G., Matthey, L., Creswell, A., Botvinick, M., Lerchner, A., and Burgess, C. P. (2021). SIMONe: View-Invariant, Temporally-Abstracted Object Representations via Unsupervised Video Decomposition. pages 1–29.

[Kim and Mnih, 2018] Kim, H. and Mnih, A. (2018). Disentangling by factorising. *35th International Conference on Machine Learning, ICML 2018*, 6:4153–4171.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, (Ml):1–14.

[Kingma and Welling, 2019] Kingma, D. P. and Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392.

[Kosiorek et al., 2018] Kosiorek, A. R., Kim, H., Posner, I., and Teh, Y. W. (2018). Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 2018-Decem(Nips):8606–8616.

[Kosiorek et al., 2019] Kosiorek, A. R., Sabour, S., Teh, Y. W., and Hinton, G. E. (2019). Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*, volume 32.

[Kumar et al., 2018] Kumar, A., Sattigeri, P., and Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, (2017).

[Lecun, 1987] Lecun, Y. (1987). *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models).* Universite P. et M. Curie (Paris 6).

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.

[Lin et al., 2020] Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020). SPACE: Unsupervised Object-Oriented Scene Representation via Spatial Attention and Decomposition. pages 1–22.

[Locatello et al., 2019] Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. *RML@ICLR 2019 Workshop - Reproducibility in Machine Learning.*

[Locatello et al., 2020] Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS):1–27.

[Marino et al., 2018] Marino, J., Cvitkovic, M., and Yue, Y. (2018). A general method for amortizing variational filtering. *Advances in Neural Information Processing Systems*, 2018-Decem(Nips):7857–7868.

[Masson-Delmotte et al., 2018] Masson-Delmotte, V., Zhai, P., Pörtner, H.-O., Roberts, D., Skea, J., Shukla, P. R., Pirani, A., Moufouma-Okia, W., Péan, C., Pidcock, R., et al. (2018). Global warming of 1.5 c. *An IPCC Special Report on the impacts of global warming of*, 1(5).

[McCormac et al., 2017] McCormac, J., Handa, A., Leutenegger, S., and J.Davison, A. (2017). Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?

[Mur-Artal and Tardós, 2017] Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* MIT Press.

[Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *ISMAR '11 Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE.

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

[Redmon et al., 2015] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. cite arxiv:1506.02640.

[Ridgeway and Mozer, 2018] Ridgeway, K. and Mozer, M. C. (2018). Learning deep disentangled embeddings with the f-statistic loss. In *NeurIPS.*

[Ring et al., 2012] Ring, M. J., Lindner, D., Cross, E. F., Schlesinger, M. E., et al. (2012). Causes of the global warming observed since the 19th century. *Atmospheric and Climate Sciences*, 2(04):401.

[Ripple et al., 2019] Ripple, W. J., Wolf, C., Newsome, T. M., Barnard, P., and Moomaw, W. R. (2019). World Scientists' Warning of a Climate Emergency. *BioScience*, 70(1):8–12.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

[Sepliarskaia et al., 2019] Sepliarskaia, A., Kiseleva, J., and Rijke, M. D. (2019). How Not to Measure Disentanglement. In *arxiv*.

[Shukla et al., 2019] Shukla, P., Skea, J., Calvo Buendia, E., Masson-Delmotte, V., Pörtner, H., Roberts, D., Zhai, P., Slade, R., Connors, S., Van Diemen, R., et al. (2019). Ipcc, 2019: Climate change and land: an ipcc special report on climate change, desertification, land degradation, sustainable land management, food security, and greenhouse gas fluxes in terrestrial ecosystems.

[Stern and Kaufmann, 2014] Stern, D. I. and Kaufmann, R. K. (2014). Anthropogenic and natural causes of climate change. *Climatic change*, 122(1):257–269.

[Strubell et al., 2019] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. *CoRR*, abs/1906.02243.

[ThomasGriffin, 2020] ThomasGriffin (2020). Why we should care about the environmental impact of ai. `https://www.forbes.com/sites/forbestechcouncil/2020/08/17/why-we-should-care-about-the-environmental-impact-of-ai/`.

[Thompson et al., 2020] Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F. (2020). The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*.

[Toews, 2020] Toews, R. (2020). Deep learning's carbon emissions problem. `https://www.forbes.com/sites/robtoews/2020/06/17/deep-learnings-climate-change-problem/`.

[Torch Contributors, 2021a] Torch Contributors (2021a). 2d convolutional layer class. `https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html?highlight=conv#torch.nn.Conv2d`.

[Torch Contributors, 2021b] Torch Contributors (2021b). Dense linear layer class. https://pytorch.org/docs/1.10.0/generated/torch.nn.Linear.html?highlight=linear#torch.nn.Linear.

[Ulyanov et al., 2017] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2017). Instance normalization: The missing ingredient for fast stylization.

[United Nations, 2015] United Nations (2015). 2030 agenda for sustainable development. https://sdgs.un.org/2030agenda.

[Veerapaneni et al., 2019] Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J. B., and Levine, S. (2019). Entity Abstraction in Visual Model-Based Reinforcement Learning. (CoRL):1–18.

[Watters et al., 2019] Watters, N., Matthey, L., Burgess, C. P., and Lerchner, A. (2019). Spatial Broadcast Decoder: A Simple Architecture for Learning Disentangled Representations in VAEs. pages 1–35.

[Whittington et al., 2021] Whittington, J. C. R., Kabra, R., Matthey, L., Burgess, C. P., and Lerchner, A. (2021). Constellation: Learning relational abstractions over objects for compositional imagination.

[World Health Organization et al., 2014] World Health Organization et al. (2014). Quantitative risk assessment of the effects of climate change on selected causes of death, 2030s and 2050s.

[Zaidi et al., 2020] Zaidi, J., Boilard, J., Gagnon, G., and Carbonneau, M.-A. (2020). Measuring Disentanglement: A Review of Metrics. pages 1–19.

Appendices

Appendix A

Objects Room images

A.1 Weight initialization images

This section shows further examples of the segmentation performance on the weights ablation comparison. All figures follow the same pattern. Each example shows the output of a different configuration. *Reconstruction mixture* shows the sum of components from all slots, weighted by the learned masks from the attention network. *Segmentation* shows a colour-coded map summarising the attention masks m_k. Rows labelled S1-S7 shows the reconstruction components of each slot. Unmasked versions are shown side-by-side with corresponding versions that are masked with the VAE's reconstructed masks $\tilde{m}_k$

A.2 Disentanglement images

This section shows further images for the disentanglement results from MONet, MONet-RN50, and MONet-RN50* on *Objects Room*.

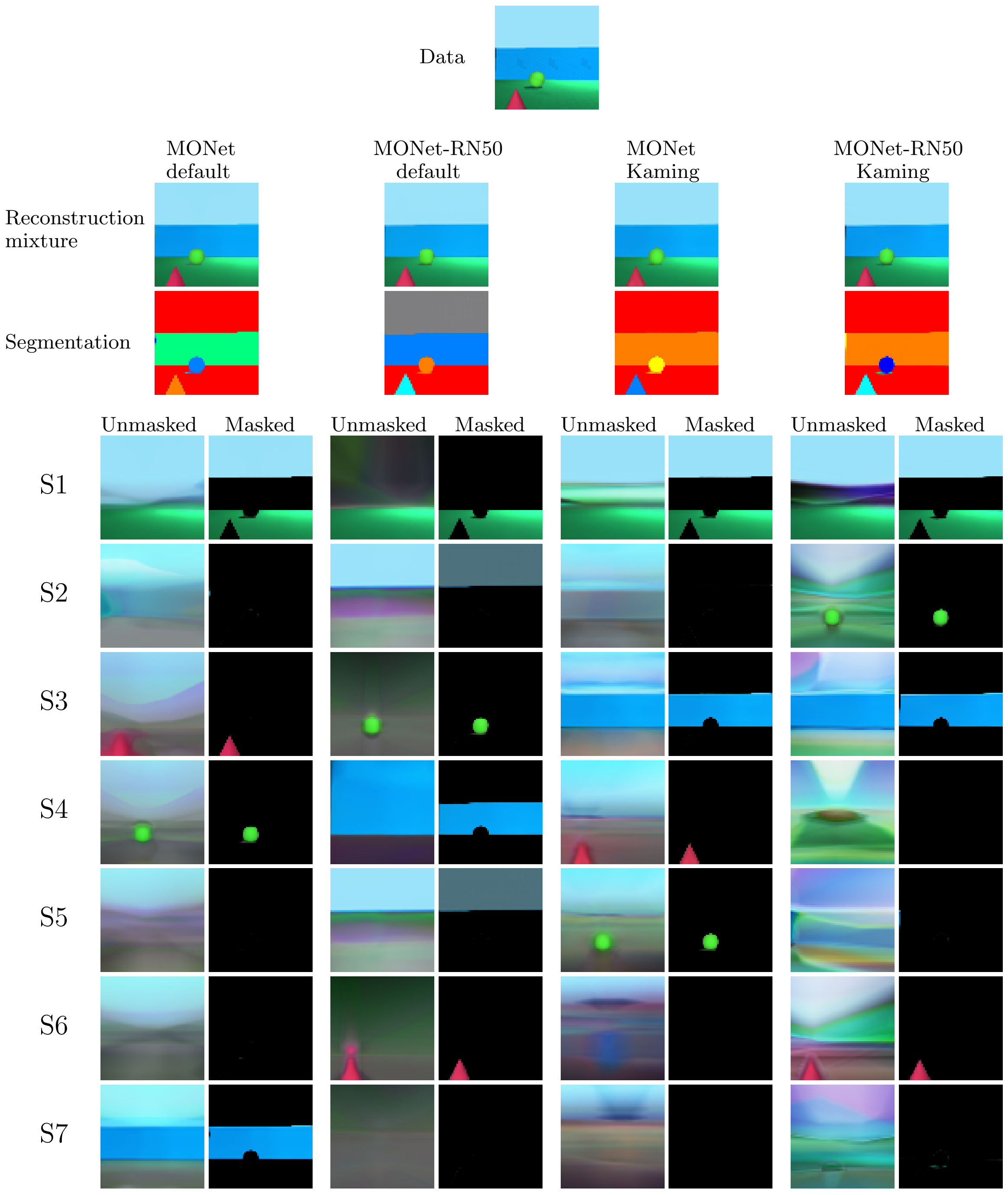


Figure A.1: Decomposition results on *Objects Room* for weight ablation. Extra 1.

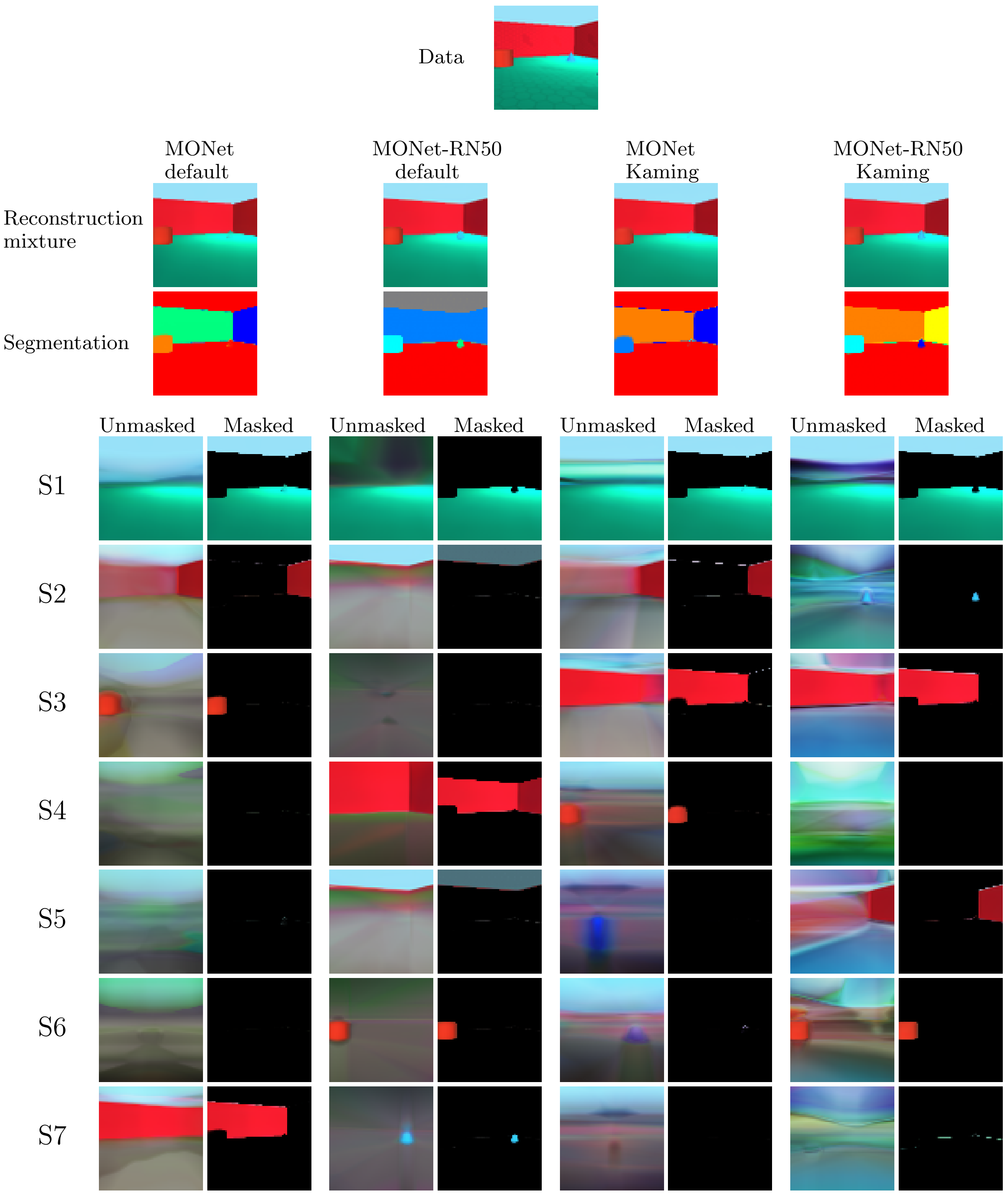


Figure A.2: Decomposition results on *Objects Room* for weight ablation. Extra 2.

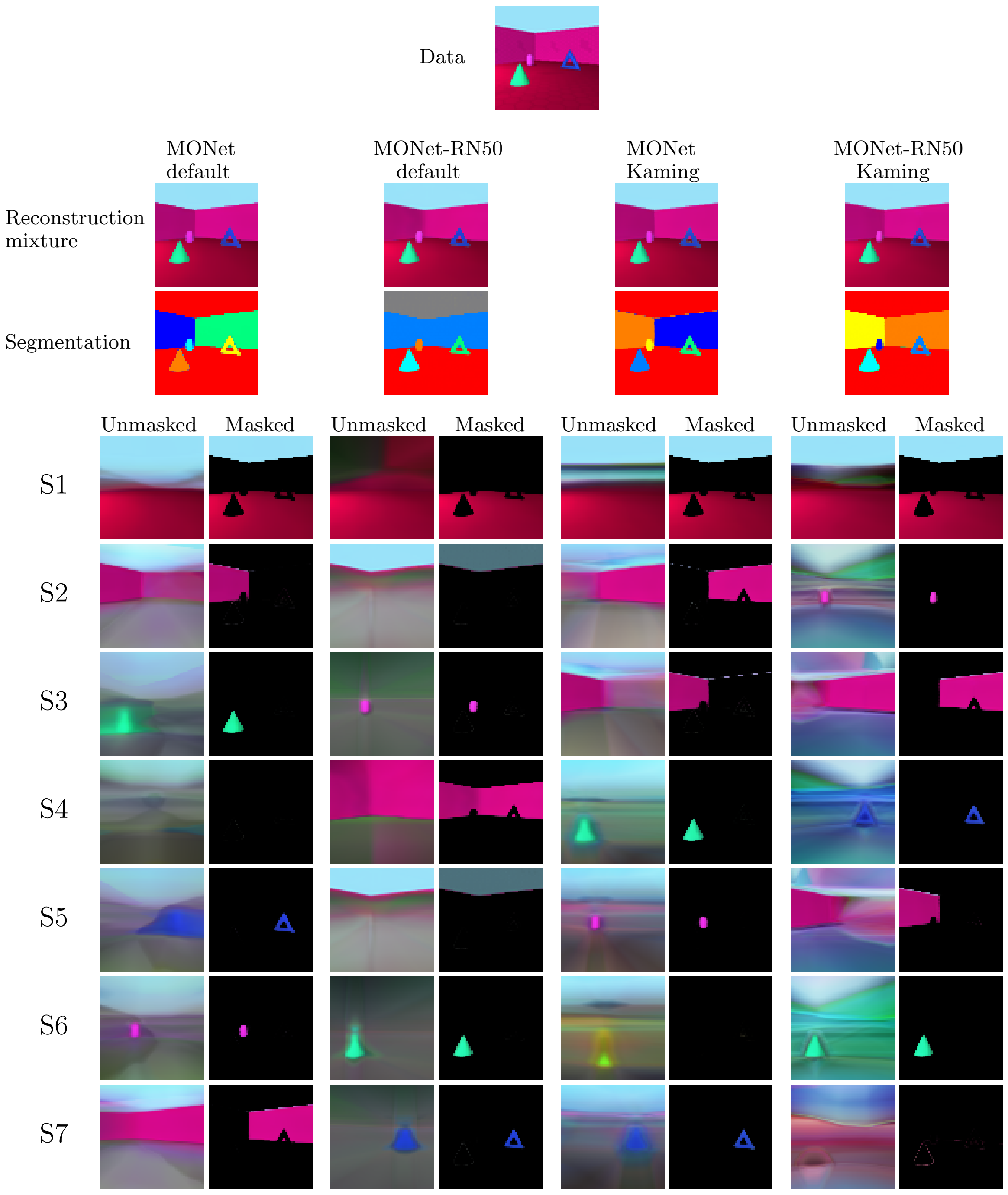


Figure A.3: Decomposition results on *Objects Room* for weight ablation. Extra 3.

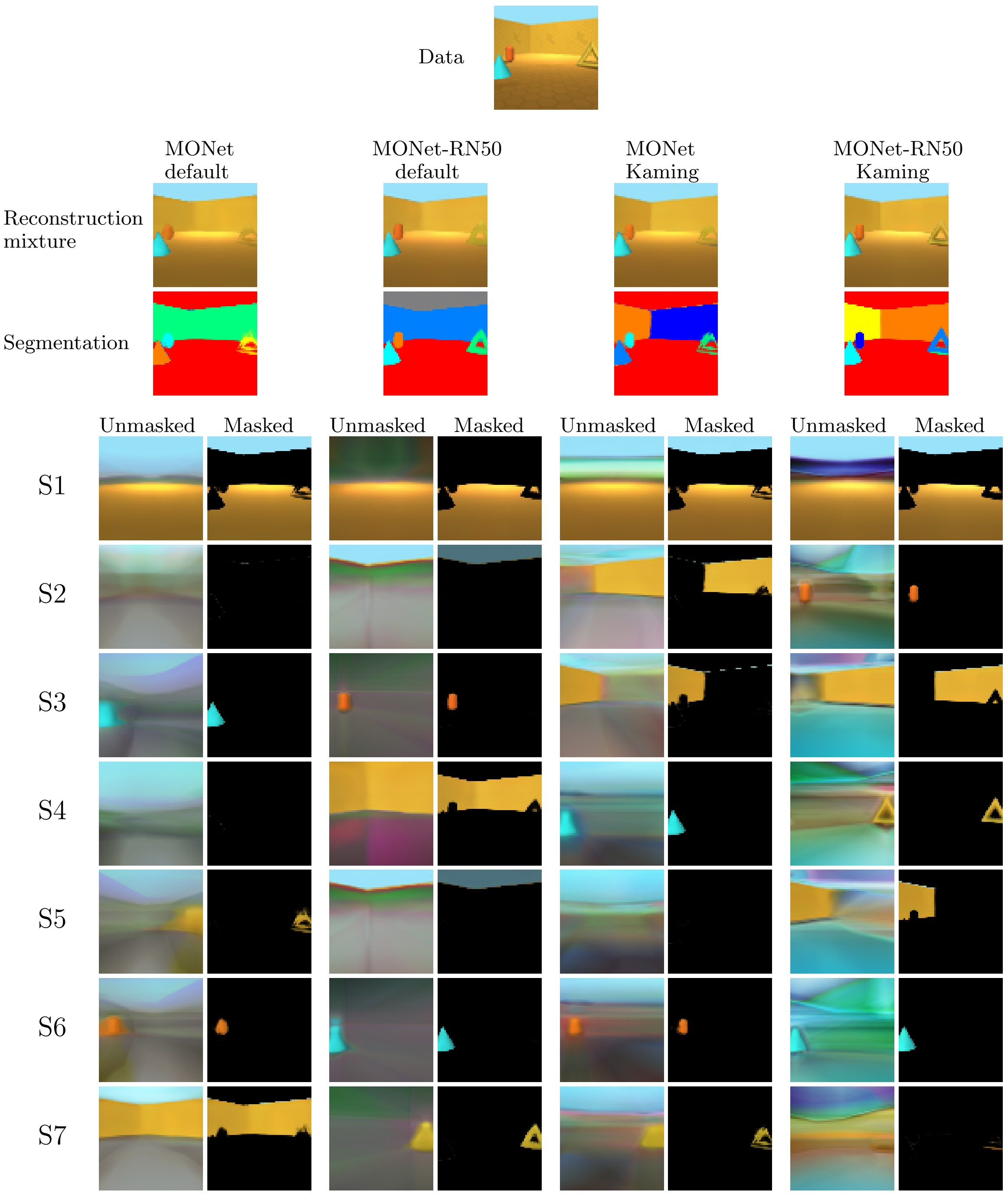


Figure A.4: Decomposition results on *Objects Room* for weight ablation. Extra 4.

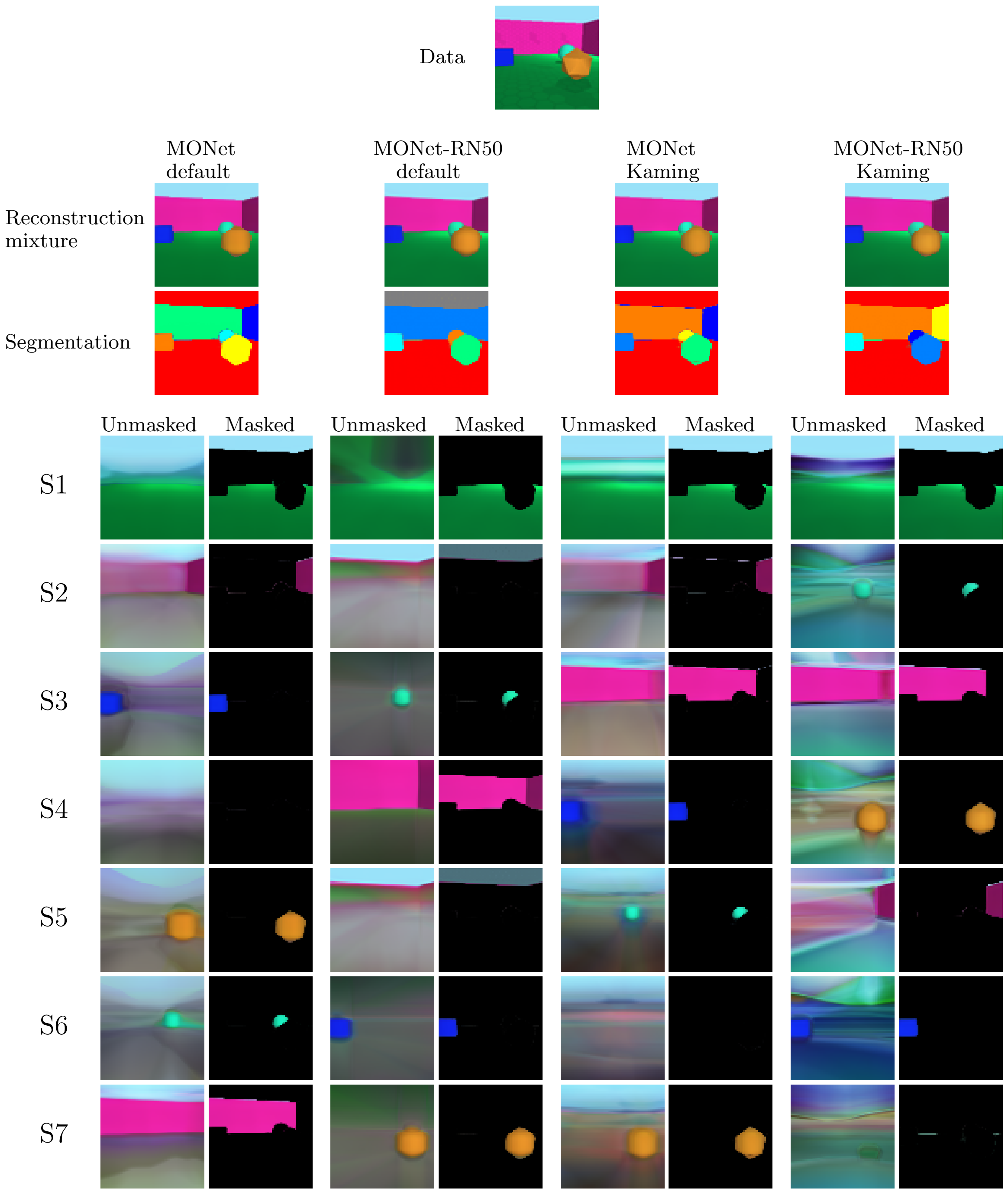


Figure A.5: Decomposition results on *Objects Room* for weight ablation. Extra 5.

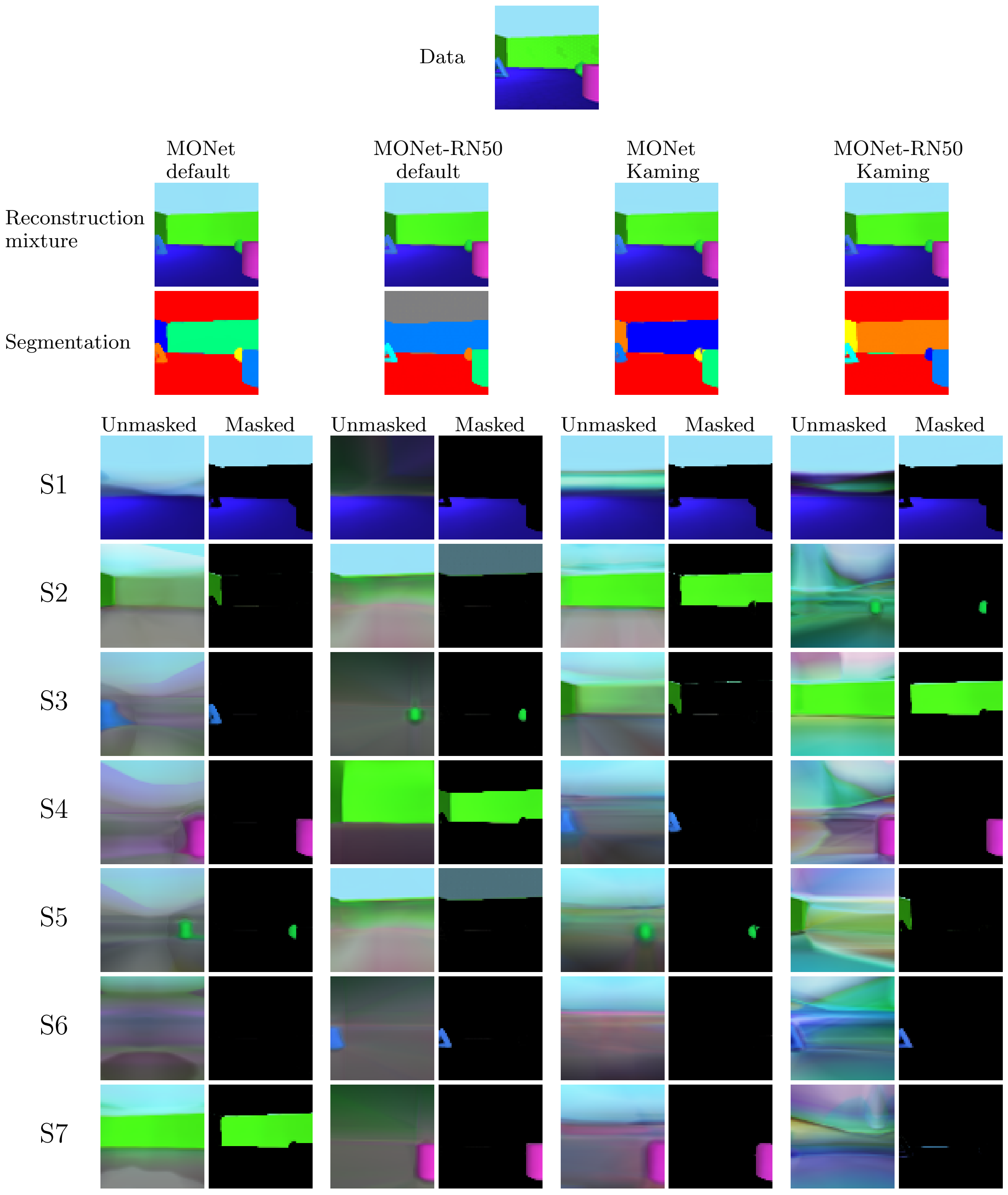


Figure A.6: Decomposition results on *Objects Room* for weight ablation. Extra 6.

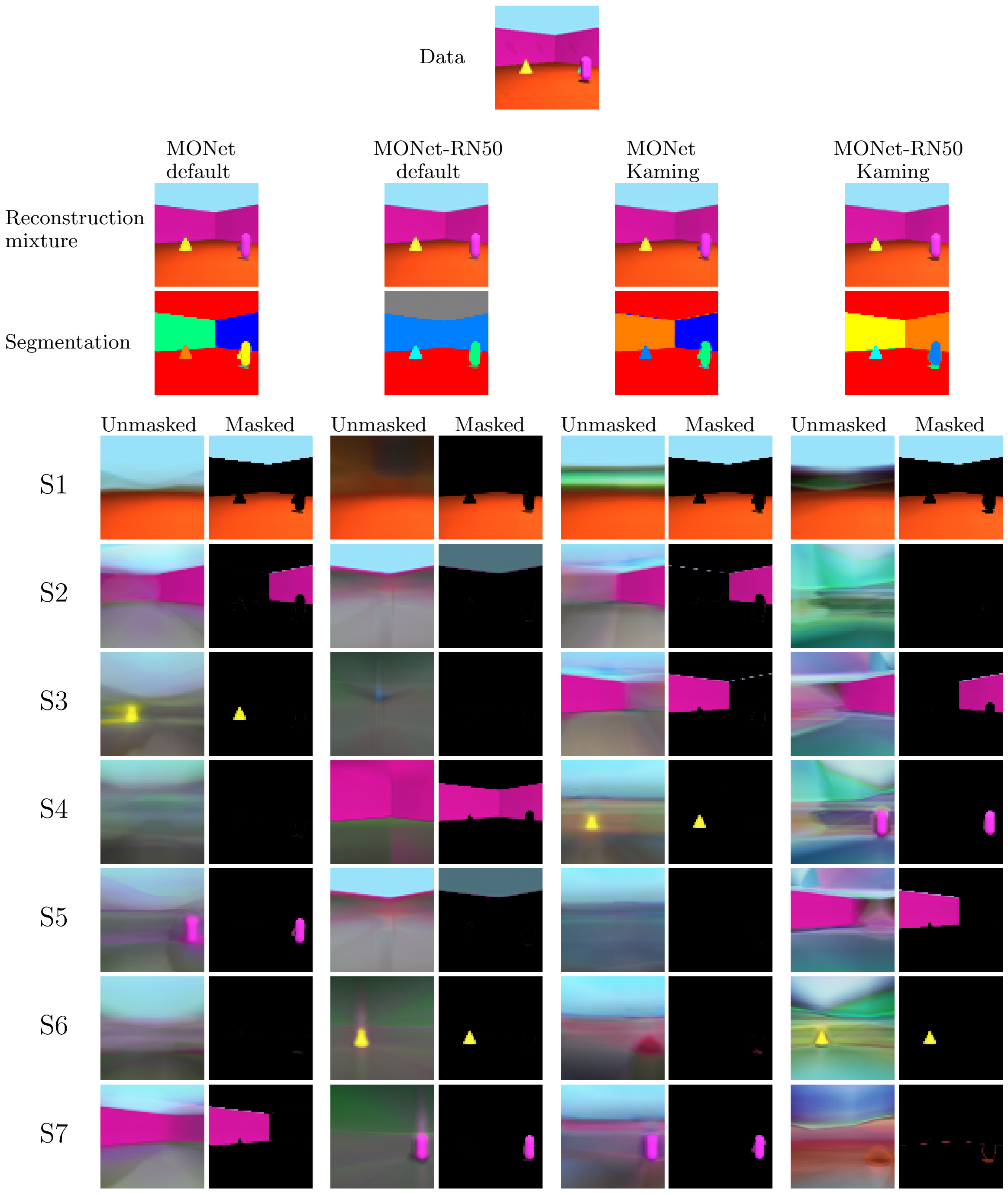


Figure A.7: Decomposition results on *Objects Room* for weight ablation. Extra 7.

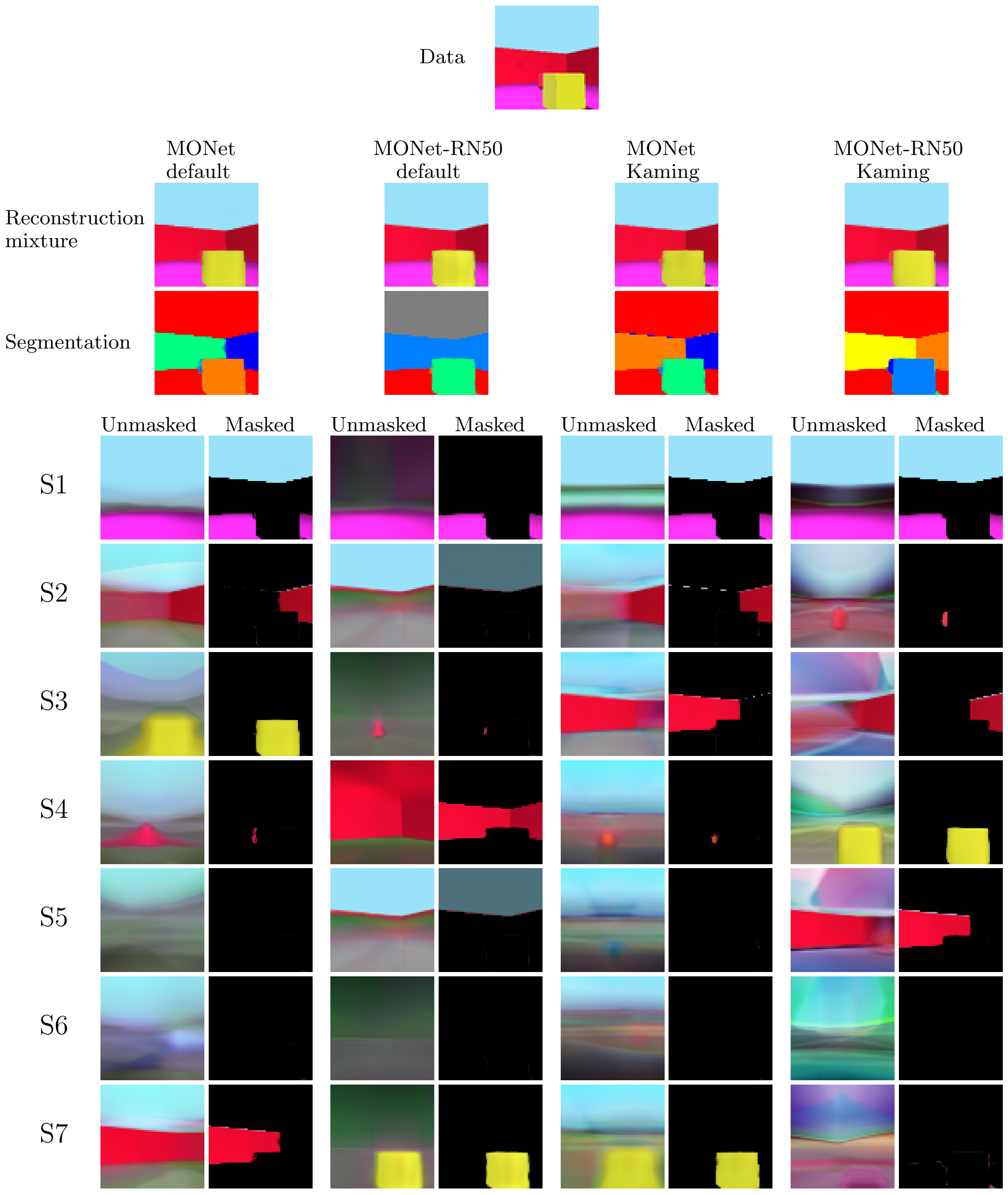


Figure A.8: Decomposition results on *Objects Room* for weight ablation. Extra 8.

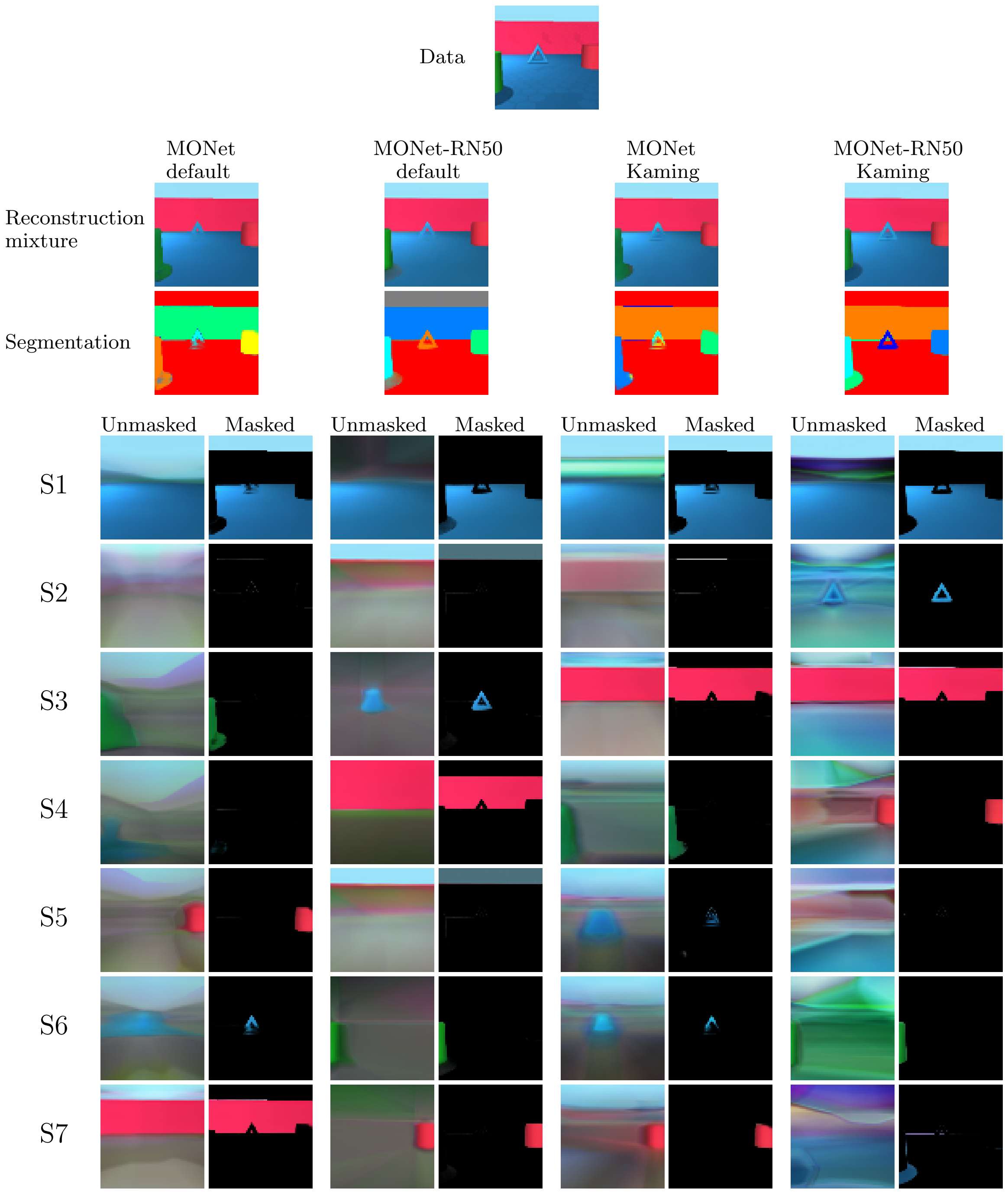


Figure A.9: Decomposition results on *Objects Room* for weight ablation. Extra 9.

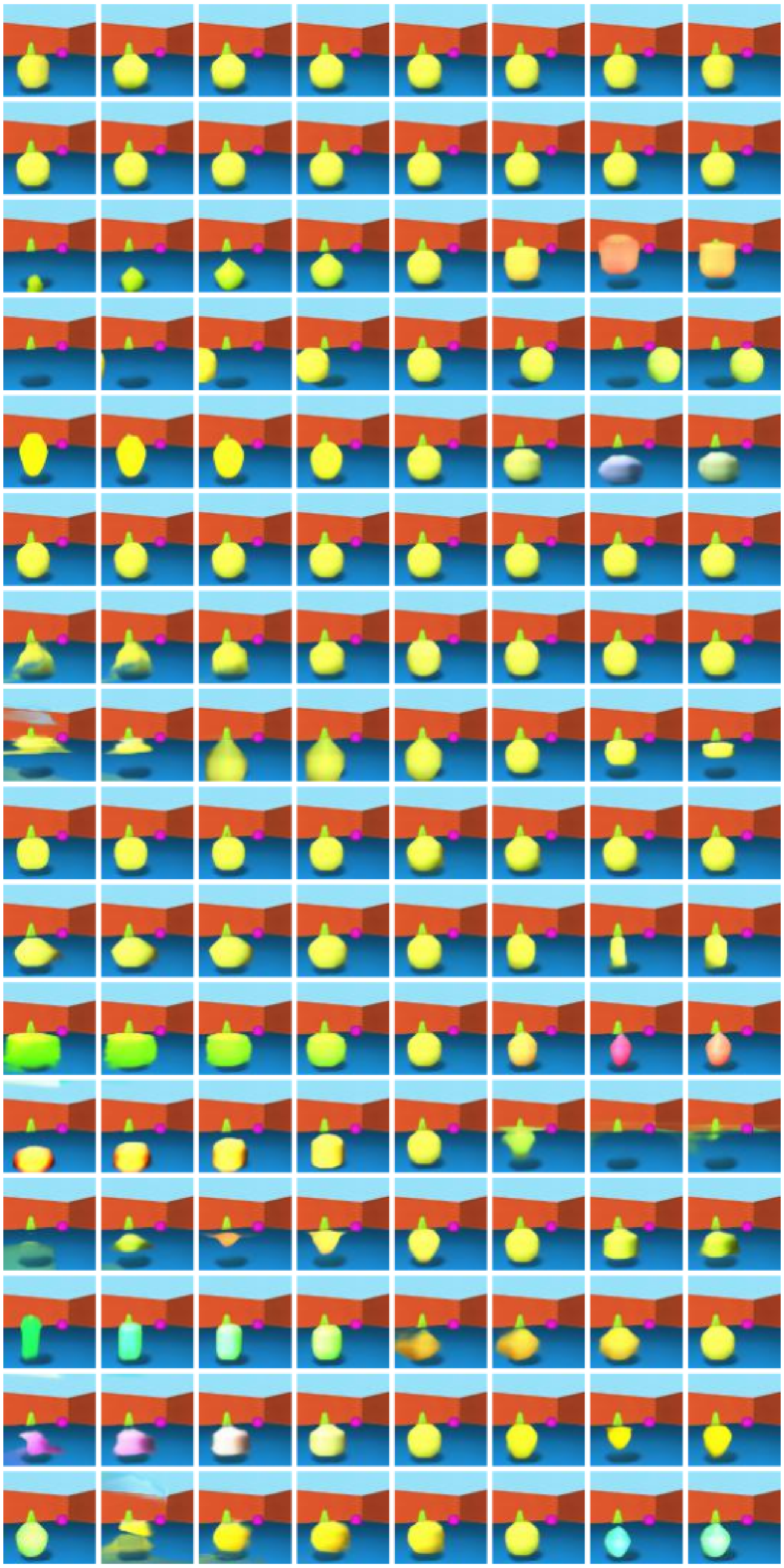


Figure A.10: Disentanglement evaluation on MONet on *Objects Room*. From up to down all the latent factors of slot 5, yellow sphere, for $z_5 - i$, from $i = 1$ up to $i = 16$

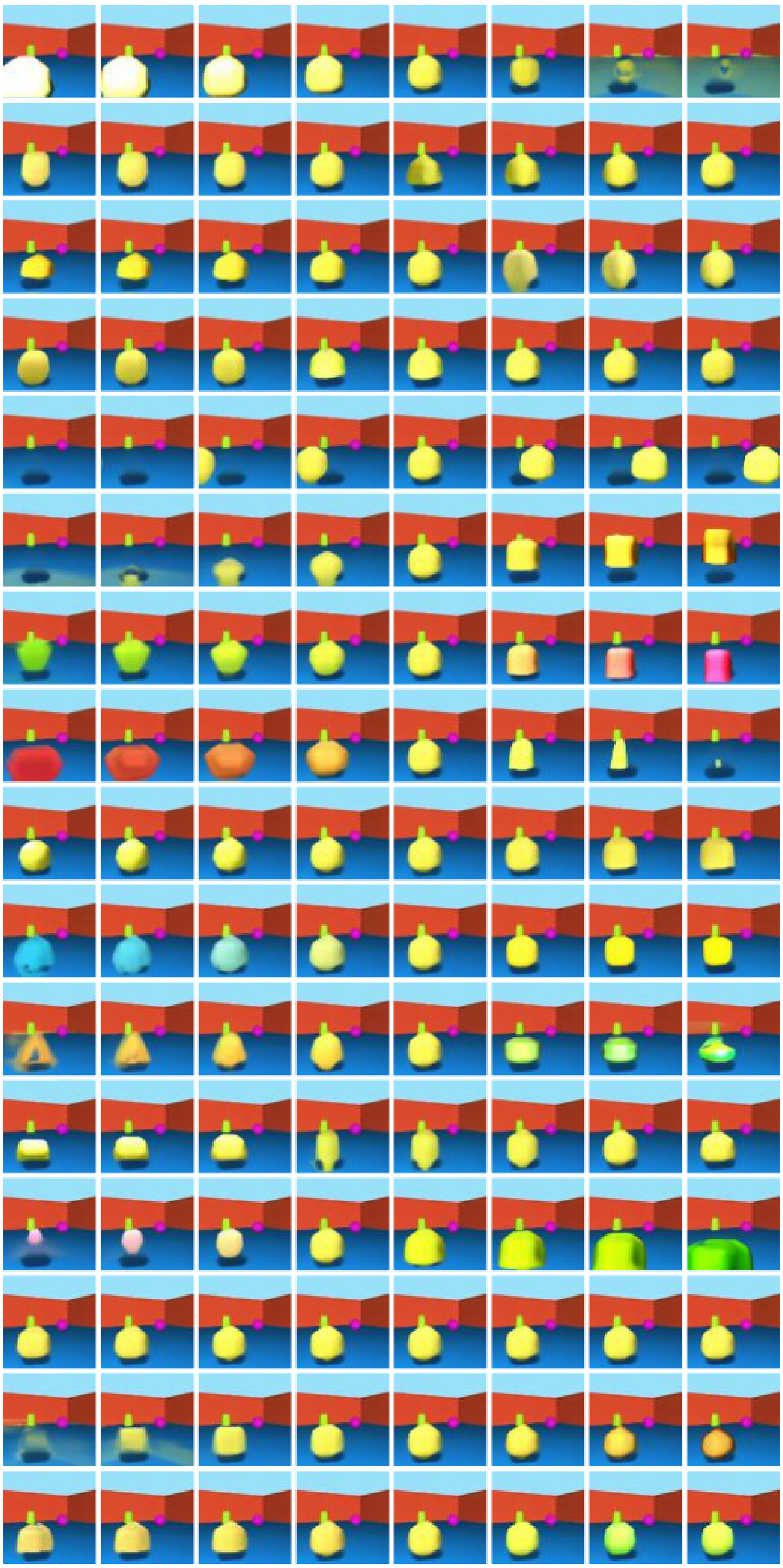


Figure A.11: Disentanglement evaluation on MONet-RN50 on *Objects Room*. From up to down all the latent factors of slot 5, yellow sphere, for $z_5 - i$, from $i = 1$ up to $i = 16$

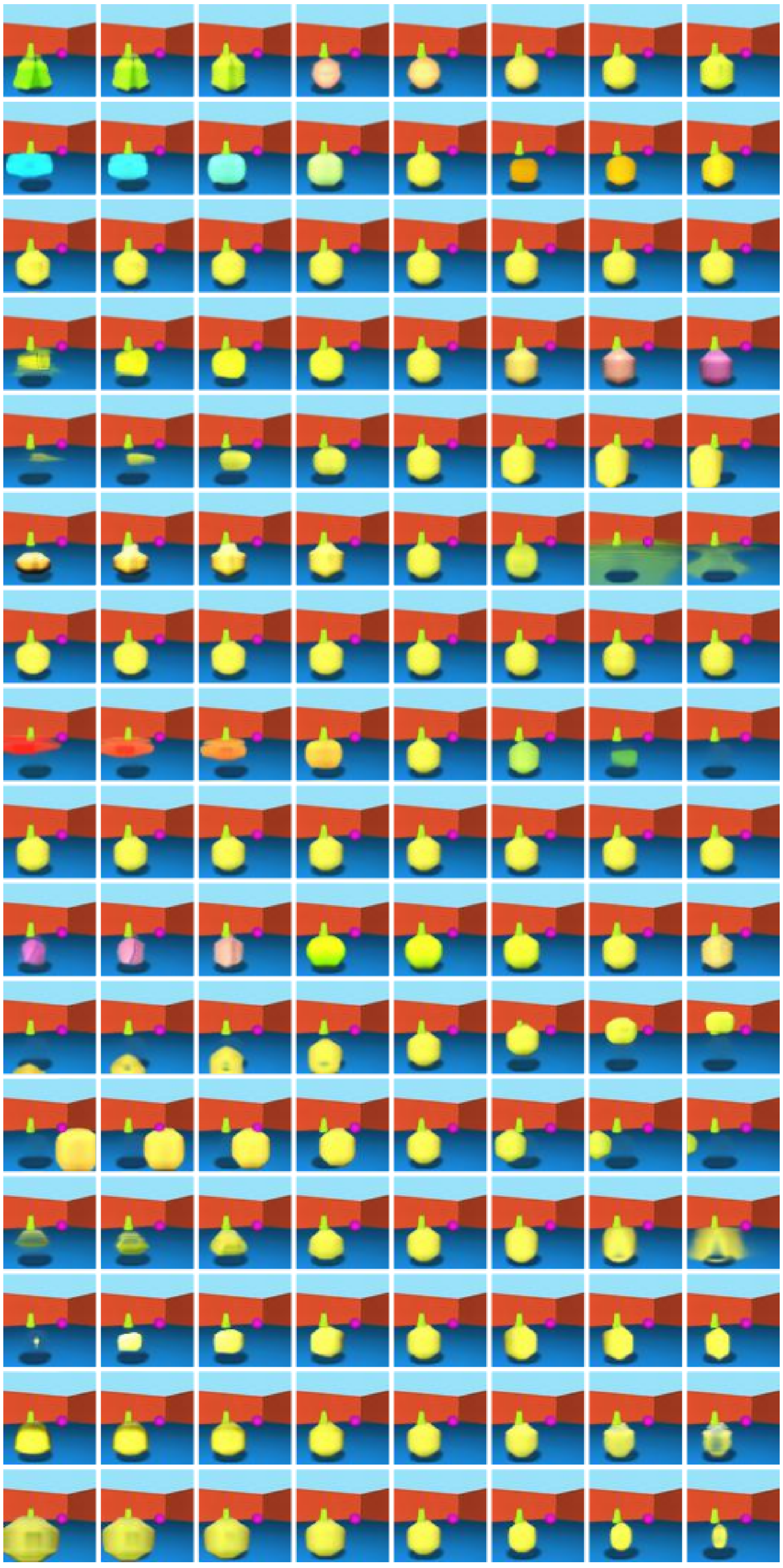


Figure A.12: Disentanglement evaluation on MONet-RN50* on *Objects Room*. From up to down all the latent factors of slot 2, yellow sphere, for $z_2 - i$, from $i = 1$ up to $i = 16$

Appendix B

Clevr images

B.1 MONet and MONet-RN50* segmentation comparison

This section shows further images for the segmentation results from MONet and MONet-RN50* on *Clever*. All the figures follow the same structure. *Reconstruction mixture* shows the sum of components from all slots, weighted by the learned masks from the attention network. *Segmentation* shows a colour-coded map summarising the attention masks m_k. Rows labelled S1-S11 shows the reconstruction components of each slot. Unmasked versions are shown side-by-side with corresponding versions that are masked with the VAE's reconstructed masks $\tilde{m}_k$

B.2 MONet and MONet-RN50* disentangled comparison

This section shows further images for the disentanglement results from MONet and MONet-RN50* on *Clever*.

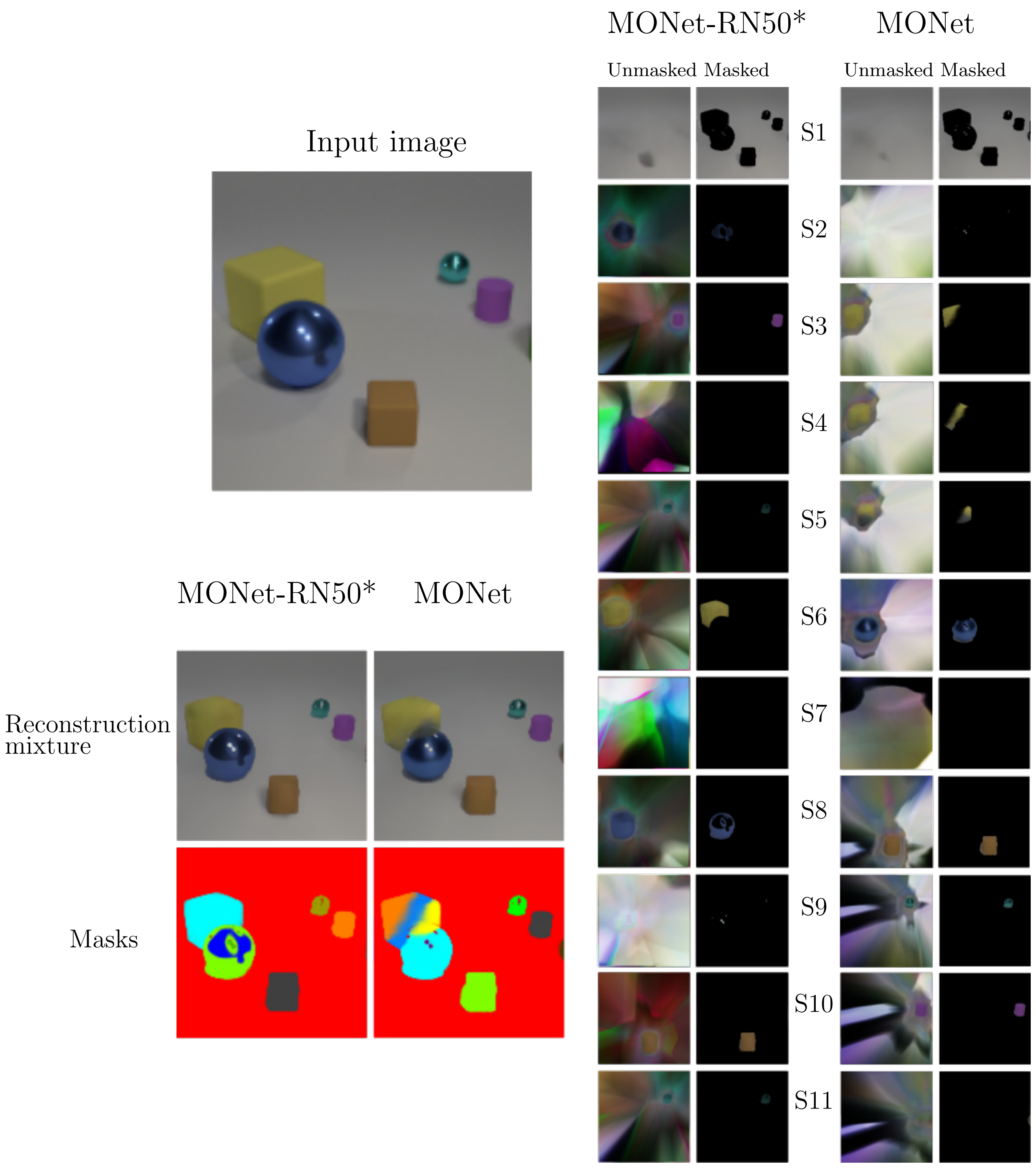


Figure B.1: Decomposition results of 5 objects on *CLEVR* comparison.

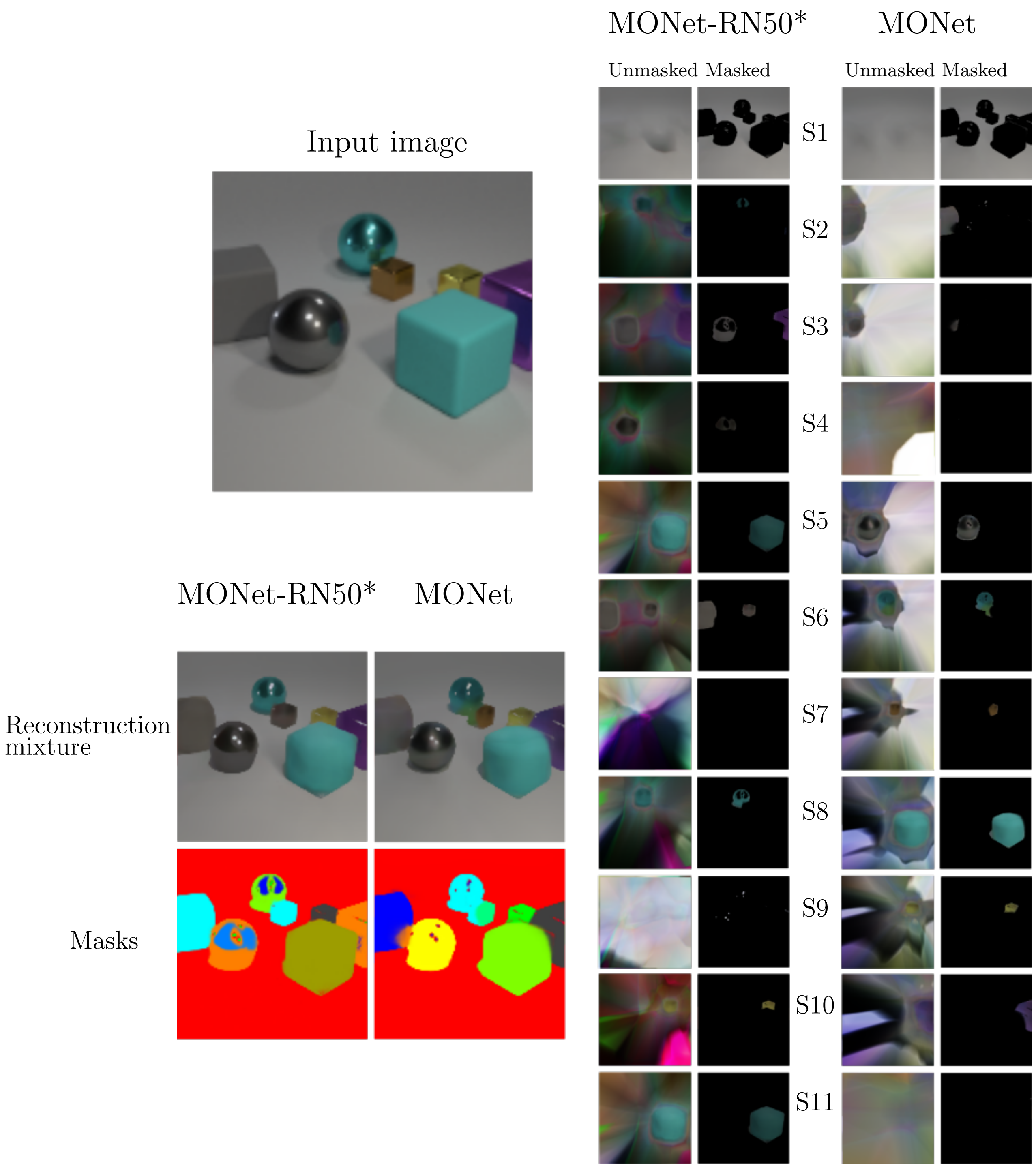


Figure B.2: Decomposition results of 7 objects on *CLEVR* comparison.

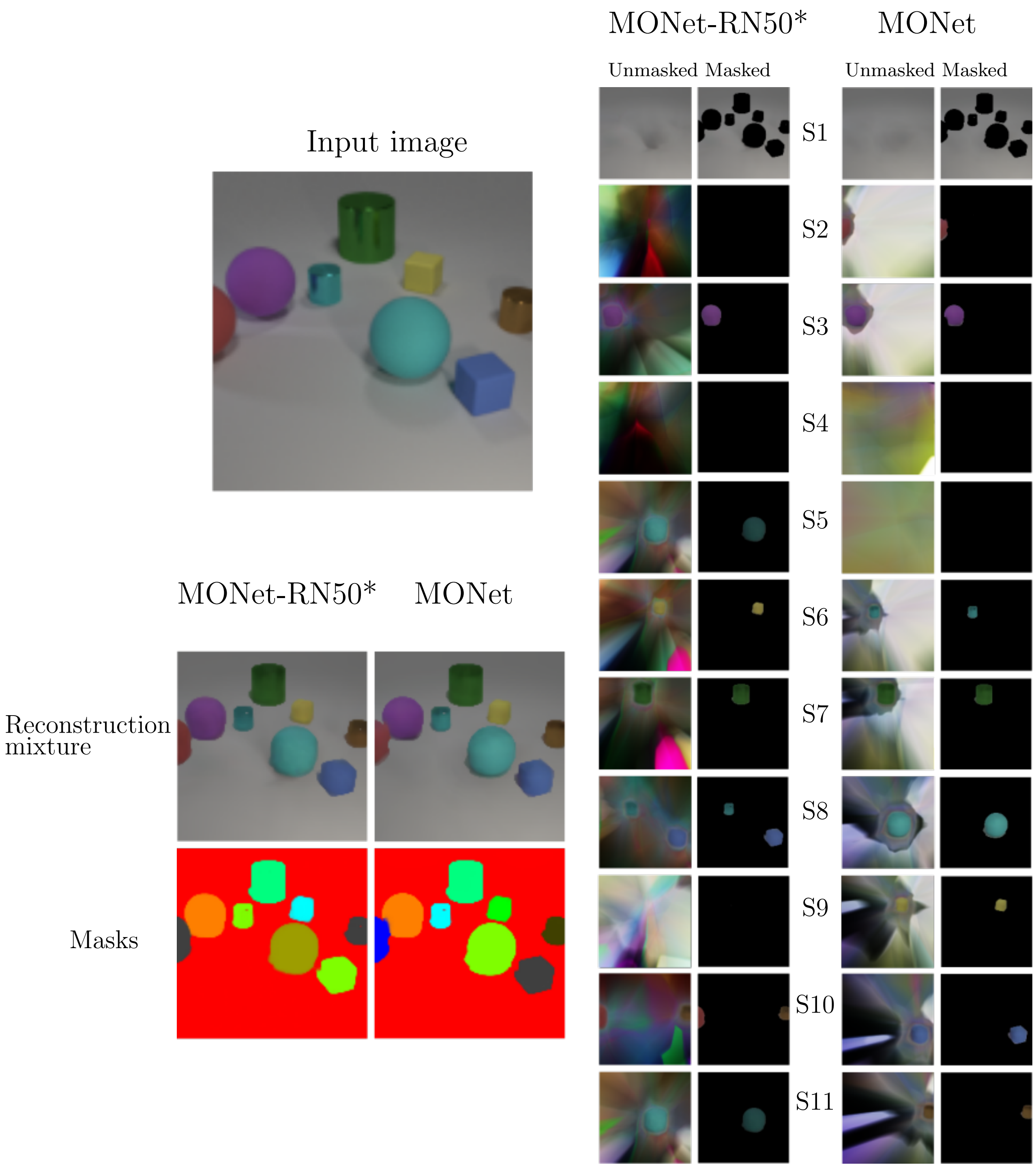


Figure B.3: Decomposition results of 8 objects on *CLEVR* comparison.

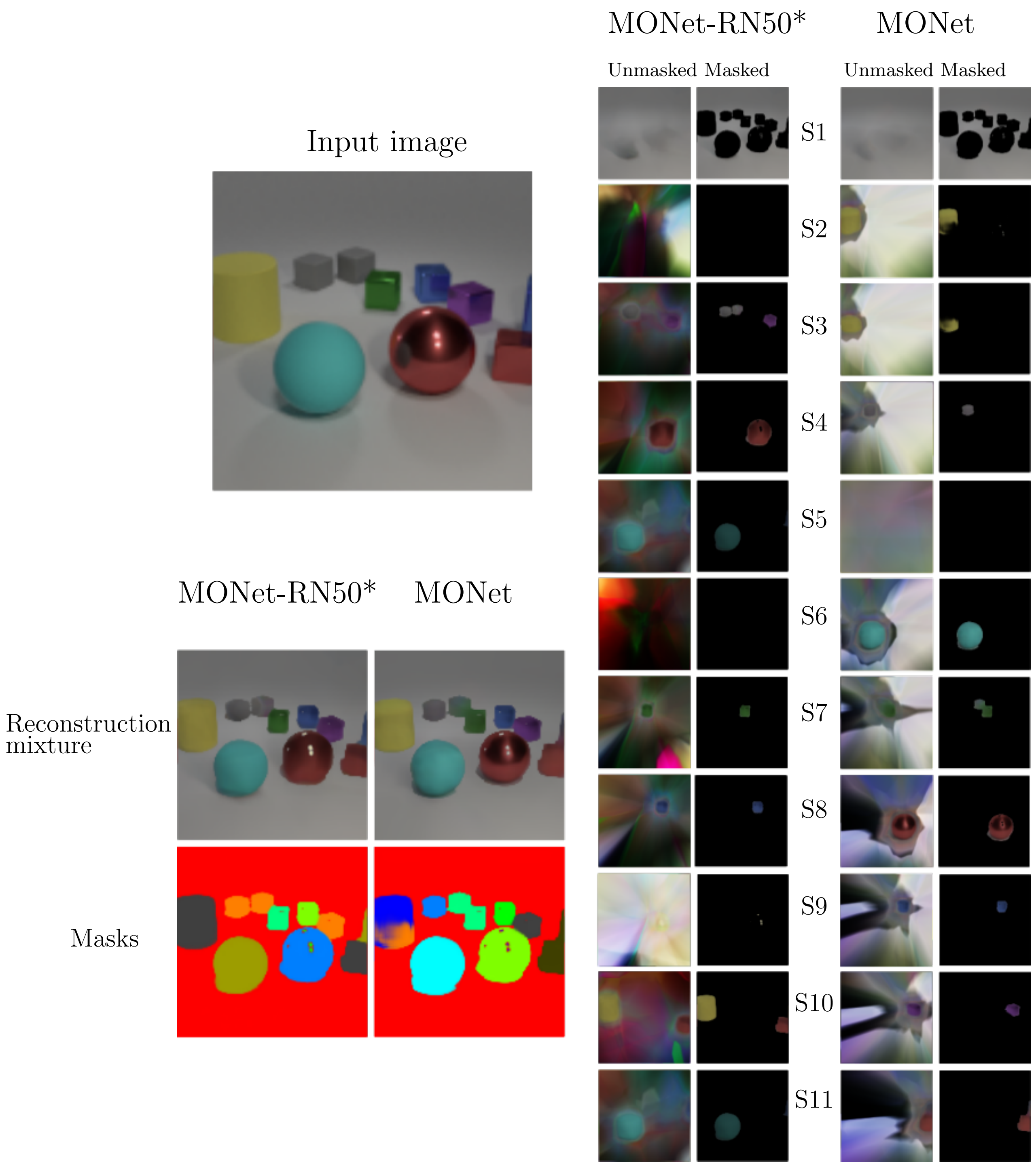


Figure B.4: Decomposition results of 9 objects on *CLEVR* comparison.

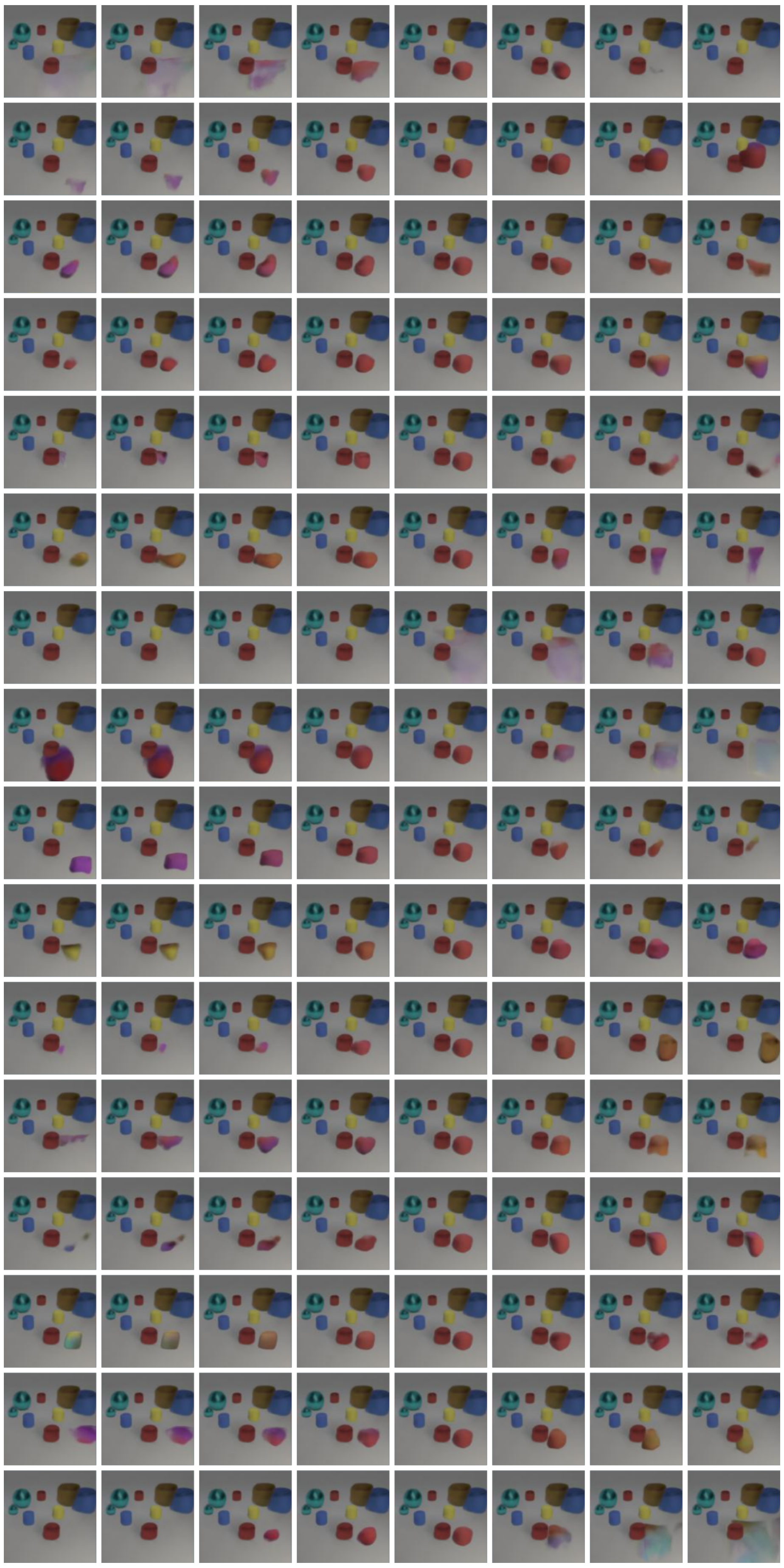


Figure B.5: Disentanglement evaluation on MONet-RN50* on *CLEVR*.From up to down all the latent factors of slot 8, right red cilinder, for $z_8 - i$, from $i = 1$ up to $i = 16$

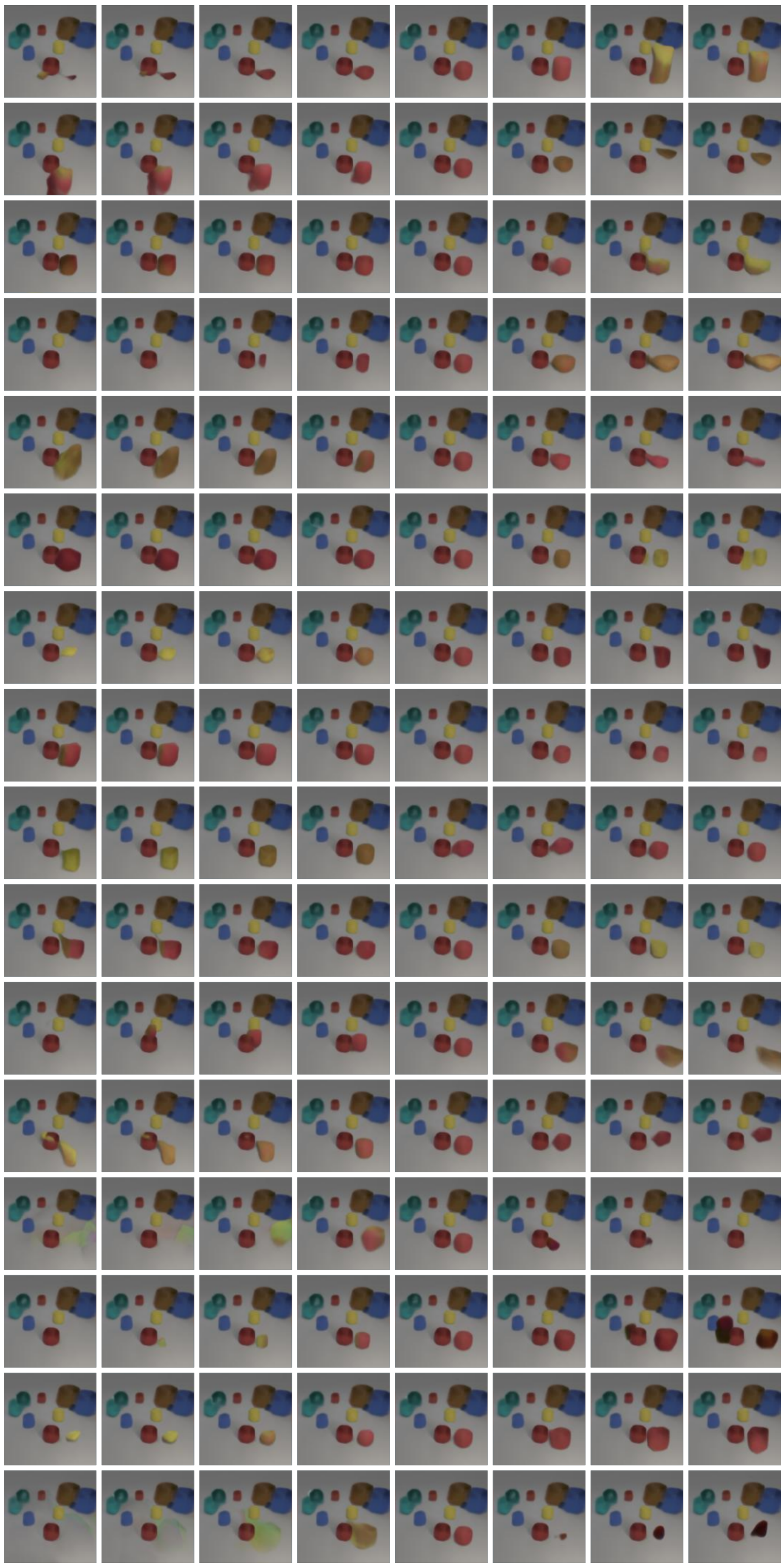


Figure B.6: Disentanglement evaluation on MONet-RN50* on *CLEVR*. From up to down all the latent factors of slot 9, right red cilinder, for $z_9 - i$, from $i = 1$ up to $i = 16$