Learning Structured Neural Representations for Visual Reasoning Tasks

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Sjoerd van Steenkiste

under the supervision of
Jürgen Schmidhuber

November 2020

Dissertation Committee

Cesare Alippi	Università della Svizzera Italiana, Switzerland
Natasha Sharygina	Università della Svizzera Italiana, Switzerland
Bernhard Schölkopf	Max Planck Institute for Intelligent Systems, Germany
Michael C. Mozer	University of Colorado Boulder, USA
Leslie P. Kaelbling	Massachusetts Institute of Technology, USA

Dissertation accepted on 23 November 2020

Research Advisor	PhD Program Director
Jürgen Schmidhuber	**Silvia Santini**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Simon Jacob “Sjoerd” van Steenkiste
Lugano, 23 November 2020

Abstract

Deep neural networks learn representations of data to facilitate problem-solving in their respective domains. However, they struggle to acquire a structured representation based on more symbolic entities, which are commonly understood as core abstractions central to human capacity for generalization. This dissertation studies this issue for visual reasoning tasks. Inspired by how humans solve these tasks, we propose to learn structured neural representations that distinguish objects: abstract visual building blocks that can separately be composed and reasoned with. We investigate the limitations of current deep neural networks at effectively discovering, representing, and relating these more symbolic entities, and present several improvements.

To address the problem of discovering and representing objects, we propose two novel approaches. In one case, we formalize this problem as a pixel-level clustering problem and formulate a neural differentiable clustering algorithm that solves it. We demonstrate how, unlike standard representation learning techniques, it can be trained to learn about objects in an unsupervised manner and acquire corresponding representations that can be treated as symbols for reasoning. In the other case, we adopt a purely generative approach and demonstrate how a neural network equipped with the right inductive bias can learn about objects in the process of synthesizing images, even in complex visual settings.

Concerning the problem of relating symbolic entities with neural networks, we investigate how object representations can help facilitate building structured models for common-sense physical reasoning that generalize more systematically. We extend our previous representation learning approach to facilitate model building in this way and demonstrate how it can learn about general relations between objects to reason about their (future) physical interactions.

Finally, we investigate the utility of a representational format that isolates independent sources of information for encoding the features of individual objects. We conduct a large-scale study of such 'disentangled' representations that includes various methods and metrics on two new abstract visual reasoning tasks. Our results indicate that better disentanglement enables quicker learning using fewer samples.

Acknowledgements

I would like to thank Jürgen Schmidhuber for providing me with the opportunity to pursue a PhD under his supervision, and for giving me the freedom to research the topics in this dissertation. Through our discussions, I learned to think critically and to develop my own understanding of AI.

Next, I would like to thank Klaus Greff in particular, for being my main collaborator and a friend throughout all these years. Several of the ideas we published together would not have existed if it were not for his initial curiosity and tenaciousness regarding the binding problem in neural networks. Through countless hours of discussion and collaboration, we went far beyond what I could have independently achieved.

I would also like to thank my other colleagues at IDSIA for providing an inspirational research environment throughout the years. It were the interactions with Jan Koutník, Bas Steunebrink, Rupesh Srivastava, and Marijn Stollenga that led me to join IDSIA, and with Paulo Rauber and Imanol Schlag that kept me going. I would also like to thank Paulo Rauber for feedback on this dissertation and Louis Kirsch, Aleksandar Stanić, Róbert Csordás, and Anand Gopalakrishnan for collaborations.

Throughout my PhD, I was fortunate to intern at Google Brain in Zürich with Karol Kurach and Sylvain Gelly, which I would like to thank for this opportunity and as collaborators. I am also grateful for collaborations with many others at Google, including Olivier Bachem, Francesco Locatello, Thomas Unterthiner, Raphaël Marinier, and Marcin Michalski. I would like to thank Michael Chang for collaborating during his visit at IDSIA in the summer of 2018.

Many thanks go out to Kari for helping me balance work with much needed distraction during all these years, and to Wiebke for her patience and support. I would also like to thank Stefano and Dina for their continued support and good faith.

Finally, I would like to thank my brother, Job, and my parents, Ben & Marion, for their unconditional support and enthusiasm, and for all the memories we have made over the years. Without you, none of this would have been possible.

Contents

Contents **vii**

1 Introduction **1**
1.1 Problem Statement and Contributions 3
1.2 Structure of the Dissertation 6

2 Background **9**
2.1 Notation 9
2.2 Machine Learning 10
2.2.1 Statistical Modeling 11
2.2.2 Classification 19
2.3 Neural Networks 20
2.3.1 Architectures 21
2.3.2 Learning 25

3 Challenges & Related Work **31**
3.1 The Binding Problem 33
3.1.1 Importance of Symbols 33
3.1.2 Symbolic Processing in Connectionist Methods 34
3.1.3 The Binding Problem in Connectionist Methods 36
3.2 Representation 39
3.2.1 Representational Format 40
3.2.2 Representational Dynamics 41
3.2.3 Methods 43
3.3 Segregation 48
3.3.1 Objects 49
3.3.2 Segregation Dynamics 51
3.3.3 Methods 53
3.4 Composition 59
3.4.1 Structure 60

3.4.2 Reasoning . . . 64
3.4.3 Methods . . . 66
3.5 Disentangling Factors of Variation . . . 71
3.5.1 Informative Factors of Variation . . . 71
3.5.2 Learning Disentangled Representations . . . 72

4 Neural Expectation Maximization 75
4.1 Method . . . 76
4.1.1 Neural Spatial Mixture Model . . . 77
4.1.2 Expectation Maximization . . . 77
4.1.3 Trainable Clustering Procedure . . . 79
4.1.4 Training Objective . . . 80
4.2 Related work . . . 82
4.3 Experiments . . . 83
4.3.1 Static Shapes . . . 84
4.3.2 Flying Shapes . . . 85
4.3.3 Flying MNIST . . . 87
4.4 Discussion . . . 89

5 Relational Neural Expectation Maximization 91
5.1 Method . . . 92
5.1.1 RNN-EM as a Predictive World Model . . . 92
5.1.2 Interaction Function . . . 95
5.2 Related Work . . . 97
5.3 Experiments . . . 99
5.3.1 Bouncing Balls . . . 100
5.3.2 Hidden Factors . . . 103
5.3.3 Space Invaders . . . 103
5.4 Discussion . . . 104

6 Object Compositionality in GANs 107
6.1 Method . . . 108
6.1.1 Generative Adversarial Networks . . . 108
6.1.2 Incorporating Architectural Structure . . . 109
6.2 Related Work . . . 113
6.3 Experiments . . . 115
6.3.1 Qualitative Analysis . . . 118
6.3.2 Quantitative Analysis . . . 121
6.4 Discussion . . . 124

7 Evaluating Disentangled Representations 127
7.1 Methodology . . . 128
7.1.1 Disentanglement . . . 129
7.1.2 Abstract Visual Reasoning . . . 130
7.2 Results . . . 134
7.2.1 Learning Disentangled Representations . . . 134
7.2.2 Abstract Visual Reasoning . . . 135
7.3 Discussion . . . 141

8 Conclusion 143
8.1 Future Directions . . . 146

A Additional Experiment Details 149
A.1 Neural Expectation Maximization . . . 149
A.1.1 Static Shapes . . . 149
A.1.2 Flying Shapes . . . 150
A.1.3 Flying MNIST . . . 151
A.2 Relational Neural Expectation Maximization . . . 152
A.2.1 Bouncing Balls . . . 152
A.2.2 Space Invaders . . . 154
A.3 Object Compositionality in GANs . . . 155
A.3.1 Model specifications . . . 155
A.3.2 Hyperparameter Configurations . . . 156
A.3.3 Instance Segmentation . . . 157
A.3.4 Human Study . . . 158
A.4 Evaluating Disentangled Representations . . . 160
A.4.1 Architectures . . . 160
A.4.2 Abstract Visual Reasoning Data . . . 162

B Additional Results 165
B.1 Object Compositionality in GANs . . . 165
B.1.1 FID Study . . . 165
B.1.2 Human Study - Properties . . . 168
B.1.3 Examples of Generated Images . . . 171
B.2 Evaluating Disentangled Representations . . . 186
B.2.1 Learning Representations . . . 186
B.2.2 Abstract Visual Reasoning . . . 188

Bibliography 193

Chapter 1

Introduction

The study of *Artificial Intelligence* (AI) is concerned with programming machines to behave intelligently [Russell et al., 1995; Hutter, 2004]. Broadly speaking, such behavior is characterized by the capacity to achieve meaningful goals in a variety of situations, and by the ability to improve based on prior experiences. The most prominent examples of intelligent behavior are found among animals and humans, which have acquired general 'programs' throughout the course of evolution that enable them to learn from experiences and act as general problem solvers. In its most general form, the study of AI is therefore also concerned with certain aspects of neuroscience and cognitive psychology, which focus on understanding human intelligence and cognition.

The possibility of (an) AI to rival or surpass human capacity for problem-solving offers tremendous potential for automation and innovation in numerous domains. Conversely, the typical more 'narrow' approach to AI that is concerned with a particular domain, or problem setting, and that may sooner yield promising application, can serve as an intermediate step towards achieving this more general overarching goal. Indeed, several decades of AI research have led to significant advances in natural language understanding, computer vision, planning, and automated reasoning.

In recent years, *artificial Neural Networks* (NNs) have re-emerged as a promising technique for AI in several of these domains [Schmidhuber, 2015a]. Modern NNs consist of simple connected nodes (neurons) organized in layers that each compute a non-linear transformation of their input (i.e. the output of the previous layer) based on the weights on the incoming connections. Like other *Machine Learning* (ML) approaches [Mitchell et al., 1997], they can be programmed to produce the desired input-output mapping (in this case by adjusting their weights) through *learning from data*. This makes it possible to teach NNs to perform tasks

that are otherwise difficult to manually program efficiently, such as recognizing objects in images, but for which corresponding data (e.g. in the form of correct input-output pairs) is more easily obtained.

Part of the reason for the success of recent 'deep' neural networks — neural networks consisting of many layers — is that their intermediate layers form alternative descriptions, or *representations*, of the input data [Ivakhnenko and Lapa, 1965; Bengio et al., 2013]. Indeed, the importance of using the right representation (or features) for machine learning is well-established in the literature [Murphy, 2012]. For example, a representation that focuses only on relevant information content is less susceptible to noise and generally easier to learn from. Similarly, when considering more abstract 'high-level' features that are robust to certain changes in the input, it becomes easier to learn programs that *generalize* to other inputs. It is for this reason that a lot of previous work in ML has focused on engineering input features, whose primary purpose was to enrich the input data based on prior knowledge about relevant aspects for solving a given task (e.g. Lowe [1989]; Dalal and Triggs [2005] for vision).

Deep neural networks are capable of *learning* representations of the input data together with the desired output for the task under consideration. This significantly reduces the need for feature engineering and has led neural networks to become the default choice for learning directly from unprocessed data, such as raw visual images [Ciresan et al., 2011, 2012; Krizhevsky et al., 2012; Sharif Razavian et al., 2014]. It also makes NNs an attractive choice for *representation learning* specifically, where the main goal is to learn useful representations of the input data for the purpose of some other 'down-stream' application [Lee et al., 2009; Bengio et al., 2013]. In this case, learning can take place through more generic unsupervised learning objectives that do not require access to data that is labeled or include rewards [Schmidhuber, 1991b, 1992c; Vincent et al., 2008]. This is highly advantageous, since human labor is costly, especially when considering that deep neural networks trained on unprocessed data require access to large amounts of inputs to even learn a single task. Nonetheless, in either of these cases, several key challenges remain.

One major challenge is that it can not be guaranteed that the desired representation simply emerges as a by-product of learning, even when large amounts of supervised data are provided. Spurious correlations due to data set bias may corrupt the learned representation, while in other cases not enough data can be provided so that all possible invariances are observed. These issues are known to have important consequences for generalization, while at the same time being difficult to address during learning (since only a finite amount of data is considered) without incorporating additional assumptions [Jo and Bengio, 2017; Lake and Baroni, 2018].

Indeed, in neural networks the quality of the learned representations is primarily due to *inductive bias* [Mitchell, 1980], requiring careful consideration of the neural network architecture, objective function, and optimization procedure.

A related challenge is encountered specifically in the unsupervised case, where representations are learned based on some auxiliary task. In this case, to ensure that the learned representations turn out useful for other down-stream tasks, it is important to incorporate prior knowledge (often based on assumptions) about how such tasks are solved and the corresponding utility of a particular representational format [Kansky et al., 2017; Locatello et al., 2018]. This then also implies a trade-off, where more useful representations can often be obtained by making additional assumptions about a particular task, but which may come at the cost of their relevance to other tasks. Striking the right balance between utility and generality is another challenge that must be addressed in this framework.

1.1 Problem Statement and Contributions

The central focus of this dissertation is on *unsupervised representation learning for (relational) reasoning tasks from vision*[1]. The capacity to perform reasoning is a core aspect of human cognition and an essential ingredient to how humans solve many everyday tasks. Traditionally, this has made reasoning one of the focus areas of AI [Russell et al., 1995]. In the real world, reasoning often proceeds from vision, which offers a rich source of information that can readily be obtained and is not contingent on any form of pre-processing. On the contrary, it is difficult to reason directly in terms of low-level features, such as the color values of individual pixels. This has led researchers to consider alternative representations based on manual feature engineering or large-scale supervised learning for the purpose of solving a specific task. Unfortunately, this approach is highly laborious and therefore difficult to extend to the much broader class of visual reasoning tasks as encountered in the real world, which makes unsupervised representation learning an important problem in this domain.

To arrive at a good representation for solving visual reasoning tasks (e.g. for common-sense physical reasoning) in the unsupervised case, we will draw inspiration from how humans learn to solve such tasks. In particular, we note how human perception is structured around the discovery and representation of *objects*, which serve as abstract 'building blocks' for many aspects of cognition [Spelke and Kinzler, 2007]. For example, they allow humans to view a complex visual

[1] We define reasoning as the process of making inferences about the world based on one's understanding of how various entities influence each other.

scene in terms of separate (independent) parts and describe relations between them, which can then be reasoned with. Two important properties of objects for the purpose of representing complex visual information are their modularity and their generality. The former ensures that they can be composed in different ways to efficiently describe a variety of encountered scenes, which enables humans to reuse knowledge about an object across many different contexts. The latter implies that the notion of an object can be used to describe a wide variety of visual appearances according to a shared format, which allows them to easily be compared, and reasoned with.

We hypothesize that a representation based on objects[2] offers similar benefits to neural networks for visual reasoning tasks. Indeed, there are interesting parallels between the idea of incorporating 'object representations' (that are modular and can be composed) and the use of symbols in more classic *symbolic* approaches to AI [Nillson, 1980]. While the latter relied on hand-crafted symbols and rules of manipulation that required a prohibitive amount of human engineering, their underlying compositionality (due to treating reasoning as a symbol manipulation process) did allow them to generalize in predictable and systematic ways akin to humans. In contrast, while standard neural network approaches that learn representations are naturally able to overcome this issue of 'symbol grounding' (i.e. how to obtain abstractions that have meaning in the real world [Harnad, 1990]), their failure at generalizing more systematically suggest that they lack an inductive bias to acquire the ability to process information more symbolically [Lake and Baroni, 2018; Santoro et al., 2018b].

To this extent, we make the following contributions:

The Binding Problem in Artificial Neural Networks We investigate the inability of existing neural networks to effectively form, represent, and relate symbol-like entities (i.e. objects). We propose that the underlying cause for this is the binding problem: The inability to dynamically and flexibly bind information that is distributed throughout the network. The binding problem affects their capacity to form meaningful entities from unstructured sensory inputs (*segregation*), to maintain this separation of information at a representational level (*representation*), and to use these entities to perform new inferences, predictions, and behaviors (*composition*). Based on connections to neuroscience and cognitive psychology (e.g. Treisman [1999]), where the binding problem has been extensively studied in

[2]We define objects as abstract patterns in the (visual) input that serve as modular building blocks (i.e. they are self-contained and reusable independent of context) for solving a particular task, in the sense that they can be separately intervened upon or reasoned with.

the context of the human brain, we work towards a solution to the binding problem in neural networks and identify several important challenges and requirements. We also survey relevant mechanisms from the machine learning literature that either directly or indirectly already address some of these challenges. Our analysis provides a starting point for identifying the right inductive bias to enable neural networks to process information more symbolically and thereby generalize in a more systematic fashion.

Discovering and Representing Objects We propose two novel approaches to discover and represent objects in a way that allows them to be treated as symbols for reasoning, unlike the representations learned by standard neural network approaches. Learning about objects is particularly difficult in the unsupervised case, where the notion of an object must be acquired solely through observing statistical regularities in the data. Moreover, due to the binding problem, it is not trivial how to best encode their information with a neural network.

In order to learn about objects, we will focus on their functional role as abstract computational units that are modular and reusable across many different contexts. In one case, this allows us to treat this problem as a pixel-level clustering problem, where pixels are related through belonging to the same cluster, and where each cluster corresponds to a particular object. We propose a trainable clustering procedure called *Neural Expectation Maximization* (N-EM) that can be formalized as maximum likelihood estimation (using generalized EM [Dempster et al., 1977]) in a spatial mixture model, where each component is parametrized by a shared neural network. The weights of the neural network act as the similarity function according to which to cluster pixels, and we demonstrate how one can backpropagate gradients through the generalized EM procedure so that they can be adjusted to learn about objects in a purely unsupervised fashion.

We also investigate an alternative approach to learning about objects based on more powerful implicit generative models. In this case, this requires learning a generative process that explicitly considers such abstractions at a representational level, which can be addressed by incorporating a corresponding inductive bias that encourages the learned generative process to be compositional. We propose a structured neural network generator that allows for compositionality in this way and demonstrate how it learns about objects, relations, and background. Using this approach, we find that it is possible to learn about objects in more complex visual settings that previous approaches (including N-EM) find difficult. We also demonstrate how to leverage the learned structured generative process, which is now interpretable and semantically understood, to perform inference and recover

information about individual objects without additional supervision.

Common-sense Physical Reasoning We propose a novel approach to learning structured models for common-sense physical reasoning that takes advantage of the underlying compositionality of learned object representations to generalize in more predictable and systematic ways. To address this problem, a neural network should learn about possible relations between objects in a way that is general and reusable, and incorporate a mechanism for dynamically evaluating corresponding interactions at a representational level. Our neural approach combines N-EM with a compositional interaction function that is capable of modeling interactions between objects in this way. It decomposes complex interactions in the environment into much simpler local pair-wise interactions between individual objects and dynamically decides which objects will interact. We demonstrate how this enables it to learn a structured world model capable of making predictions about physical interactions between objects, without access to supervision, in a way that applies to scenes consisting of more or fewer objects.

Abstract Visual Reasoning using Disentangled Representations We study the usefulness of representations that are *disentangled*, which provide a particular representational format for representing information about visual scenes. In a disentangled representation, information about informative factors of variation (e.g. the color, or shape of an object) can be readily accessed and is robust to changes in the input that do not affect this factor. While 'disentanglement' may be used to refer to different ways of imposing structure at a representational level, here we are mainly concerned with disentanglement as a way of encoding information about the features of individual objects. We conduct a large-scale evaluation of disentangled representations on two abstract visual reasoning tasks (similar to Raven's Progressive Matrices [Raven, 1941]) that challenge the current capabilities of state-of-the-art deep neural networks. On these tasks, we demonstrate how representations that are more disentangled yield better sample-efficiency for learning the considered down-stream reasoning tasks.

1.2 Structure of the Dissertation

In Chapter 2 we provide a succinct overview of relevant concepts from statistical modeling and deep learning, which serve as technical background material. A reader that is already familiar with these concepts is welcome to skip this chapter or refer back to it on a case-by-case basis as indicated in the proceeding chapters.

In Chapter 3 we provide a detailed overview of the challenges and related work relevant to the topic of this dissertation. The focus of this chapter is on the binding problem, which will be introduced there since it provides a useful framework for identifying important challenges and requirements for incorporating object representations in neural networks. We also survey relevant mechanisms from the machine learning literature that either directly or indirectly already address some of these challenges. Towards the end of this chapter, we provide an overview of the challenges and related work for learning disentangled representations.

In Chapter 4 we introduce *Neural Expectation Maximization* (N-EM), which offers a trainable clustering approach based on neural networks. We demonstrate how N-EM is able to learn about objects in a purely unsupervised fashion and can be applied to images and videos to compute object representations.

In Chapter 5 we introduce *Relational Neural Expectation Maximization* (R-NEM), which combines N-EM with a relational mechanism to learn a structured model for common-sense physical reasoning. We demonstrate how it is able to leverage object representations to learn about physical interactions in a way that can be extrapolated to scenes consisting of more or fewer objects.

In Chapter 6 we introduce an alternative approach to learning about objects based on more powerful implicit generative models. In particular, we propose several modifications to a standard neural network generator to enable it to learn about objects, relations, and background, in the process of synthesizing complex visual scenes. We also demonstrate how to perform inference in this model and extract information about individual objects in unseen images.

In Chapter 7 we investigate the benefits of disentangled representations as a format for encoding information about individual objects on several abstract visual reasoning tasks. We conduct a large-scale study that spans multiple different approaches to learning disentangled representations, notions of disentanglement, data sets and hyperparameters. We present compelling evidence that more disentangled representations are beneficial for down-stream abstract visual reasoning tasks in the few-sample regime.

Chapter 8 provides a summary of our contributions and offers an outlook on promising directions for future research.

Chapter 2

Background

This chapter reviews background material on Machine Learning (ML) and Neural Networks (NNs) that is relevant to the methods developed in this dissertation. Regarding ML, most emphasis will be placed on unsupervised learning using statistical models, while for NNs we limit ourselves to a discussion of basic concepts. The reader is referred to Bishop [2006]; Bousquet et al. [2011]; Goodfellow et al. [2016] for a more detailed exposition of these topics. Prior knowledge about probability theory and statistics is assumed and we recommend Casella and Berger [2002] for an overview.

2.1 Notation

Throughout this dissertation, we will adopt the following notation. We will write a variable x as x if it is a scalar, $\boldsymbol{x}$ for (column) vectors, and $\boldsymbol{X}$ for a matrix or higher-order tensor. Capital letters are used to denote scalar random variables, e.g. X, while vector-valued or higher-order random variables are written as $\boldsymbol{X}$. Regarding the latter, we will clarify if $\boldsymbol{X}$ is a random variable or not if this is can not be determined from the surrounding context. Individual elements of vector-valued, or higher-order tensors (or random variables) are accessed via indices i, j, k. For example, we will write $\boldsymbol{X}_{i,j}$ to refer to the i, j-th element of a matrix $\boldsymbol{X}$. To select the j-th slice, we will write $\boldsymbol{X}_{:,j}$. In other cases, we will make use of subscripts to denote separate variables e.g. $x_1, \ldots, x_N$, which will be made clear from the surrounding context. The use of the element-wise product (or Hadamard product) will be indicated using $\odot$, otherwise, the standard dot-product or matrix product is assumed.

We will write $X \sim p_X$ to denote that X is distributed according to the probability density (or mass) function p_X. The dependency on X is typically clear from the

surrounding context and will normally be omitted. Similarly, depending on whether X is a discrete or a continuous random variable, we will use p_X to either refer to a probability mass function (pmf) or probability density function (pdf). We will use the common shorthand $p(x)$ to refer to $p(X = x)$, which is the value of the pdf or pmf of a particular realization of X under p_X. In certain cases, we will also write $p(x)$ to refer to the function $p(\cdot)$ when this improves readability. The possible values that a random variable X can take will be denoted with calligraphy letters, e.g. $\mathcal{X}$.

Often we will assume the existence of a 'true' or otherwise optimal value for a parameter θ, which will be denoted with an asterisk θ^*. The shorthand f_θ is sometimes used to refer to a function f having parameters θ.

In all other circumstances, we will either make use of standard notation or clarify particular notation in the respective chapters.

2.2 Machine Learning

Machine learning is the field of research dedicated to the study of algorithms and statistical models that learn from data. At a high level, the 'machine' may be given by an agent, a model, or a function, whose behavior (output) in response to some input is governed by parameters. Learning then corresponds to the process of changing these parameters in relation to the observed data to achieve the desired behavior.

Broadly speaking, ML can be partitioned into supervised learning, unsupervised learning, and reinforcement learning[1]. The main distinction between supervised and unsupervised learning is due to requiring a data set containing correct input-output pairs in the supervised case. This makes it possible to apply supervised ML directly to learn to solve some task of interest, i.e. via classification (categorical output) or regression (real-valued output). In the unsupervised case, this is not possible, which typically limits its application to discovering patterns in the data to form alternative (potentially more effective) representations. On the other hand, since acquiring supervised data is costly, this alternative way of describing the input data can have a profound impact on 'down-stream' applications that focus on prediction or analysis.

[1]For an overview of reinforcement learning, which is concerned with the study of *agents* that interact with an environment according to a policy as to maximize a form of *reward*, we refer the reader to Kaelbling et al. [1996]; Sutton et al. [1998].

2.2.1 Statistical Modeling

In the context of unsupervised learning, we will primarily concern ourselves with statistical models in this work. Let $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ be a sample drawn from a sequence of random variables $X_1, X_2, \ldots, X_N$ that are *independent and identically distributed (i.i.d.)* according to $p(\cdot \mid \theta^*)$[2]. Our goal is to model $\mathcal{D}$ with a parametrized model m that specifies some probability distribution $p(\cdot \mid \theta, m)$ over $\mathcal{X}$ based on parameters $\theta \in \Theta$[3]. The probability distribution $p(x \mid \theta, m)$ is referred to as the *likelihood*, since it determines the likelihood of observing x under θ. The *prior* $p(\theta \mid m)$ encodes our initial beliefs about likely parameter values before observing the data, which treats θ as a random variable.

Bayesian inference suggests a natural approach to unsupervised learning by computing the *posterior* distribution using Bayes's rule, which updates our beliefs about the parameters upon observing the data:

$$p(\theta \mid \mathcal{D}, m) = \frac{p(\mathcal{D} \mid \theta, m)p(\theta \mid m)}{p(\mathcal{D} \mid m)}. \tag{2.1}$$

The posterior distribution $p(\theta \mid \mathcal{D}, m)$ tells us not only about the most likely parameter value that was used to generate $\mathcal{D}$, but also provides a measure of uncertainty. The quantity in the denominator of (2.1) is referred to as the *model evidence* (or as a *marginal likelihood*), which here acts as a normalization term that is independent of θ. It can be computed by marginalizing (integrating) the joint likelihood $p(\mathcal{D} \mid \theta, m)p(\theta \mid m)$ over all possible parameter values Θ. In order to make predictions about the probability of observing new data (that is i.i.d. as before) we can compute the *posterior predictive*:

$$p(x \mid \mathcal{D}, m) = \int_\Theta p(x \mid \theta, m)p(\theta \mid \mathcal{D}, m)d\theta. \tag{2.2}$$

The posterior predictive weighs the likelihood of observing x under the current model using a particular choice of θ, by our beliefs about θ after observing $\mathcal{D}$ as encoded by the posterior (i.e. following the result of learning).

While making predictions using (2.2) (also known as the fully Bayesian approach) is advantageous for a number of reasons [Bishop, 2006], it is usually difficult to compute the exact posterior due to the normalization term in (2.1). Therefore, a standard technique that we will adopt is to use a point estimate $\hat{\theta}$ in

[2]For simplicity we will mostly focus on the univariate case in this section, although it is straightforward to extend these results to the multivariate setting.

[3]The dependence on m is normally left implicit whenever possible.

place of the full posterior, such as the *Maximum Likelihood (ML)* estimate that is obtained by maximizing the likelihood function $L(\theta \mid \mathscr{D}, m) := p(\mathscr{D} \mid \theta, m)$:

$$\hat{\theta}_{ML} := \arg\max_{\theta} L(\theta \mid \mathscr{D}, m) = \arg\max_{\theta} \prod_{i=1}^{N} p(x_i \mid \theta, m). \tag{2.3}$$

The maximum likelihood estimate corresponds to the largest mode of the posterior when assuming a Uniform prior for θ over Θ. This makes it a natural choice, although note that normally θ is not treated as a random quantity in this framework. During learning it is often easier to use the (natural) logarithm of the likelihood to compute $\hat{\theta}_{ML}$. The product of likelihoods in (2.3) then becomes a sum, while monotonicity of the logarithm leaves the location of the maximum unchanged. When using a point estimate, the predictive distribution reduces to the likelihood of the model at the estimated parameter value.

Let us now briefly consider the following example of computing $\hat{\theta}_{ML}$ for a Gaussian model, which will allow us to introduce some useful results for later in this dissertation.

Gaussian Example We assume that $\mathscr{D} = \{x_1, x_2, \ldots, x_N\}$ is an i.i.d. sample of $p(\cdot \mid \theta^*)$ and use a univariate Gaussian model with mean parameter μ, and variance σ^2, having the following probability density function:

$$\mathscr{N}(x \mid \mu, \sigma^2) := \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}. \tag{2.4}$$

For simplicity we will treat σ^2 as a fixed hyperparameter using $\sigma^2 = 1$, which leaves $\theta = \mu$. In this case, the likelihood becomes:

$$\begin{aligned} L(\theta \mid \mathscr{D}) &= \prod_{i=1}^{N} p(x_i \mid \theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(x_i - \theta)^2\right\} \\ &= \frac{1}{(2\pi)^{N/2}} \exp\left\{-\frac{1}{2}\sum_{i=1}^{N}(x_i - \theta)^2\right\}, \end{aligned} \tag{2.5}$$

where we have left the dependence on the model m implicit.

In order to compute the ML estimate we will make use of (2.3), but then using the log of the likelihood:

$$
\begin{aligned}
\hat{\theta}_{ML} &= \arg\max_{\theta} \log L(\theta \mid \mathscr{D}) \\
&= \arg\max_{\theta} -\frac{N}{2}\log 2\pi - \frac{1}{2}\sum_{i=1}^{N}(x_i-\theta)^2 \\
&= \arg\min_{\theta} \frac{1}{2}\sum_{i=1}^{N}(x_i-\theta)^2 \\
&= \arg\min_{\theta} \frac{1}{2}\sum_{i=1}^{N}(x_i-\bar{x})^2 + \frac{1}{2}\sum_{i=1}^{N}(\bar{x}-\theta)^2 \quad \left(\text{with } \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i\right) \\
&= \frac{1}{N}\sum_{i=1}^{N} x_i.
\end{aligned}
\tag{2.6}
$$

In some cases, as we will encounter, it will be more difficult to compute these estimates analytically. Using gradient-based optimization may then provide a good alternative. The gradient of the log-likelihood of a univariate Gaussian model with fixed variance and mean parameter $\theta = \mu$ is given by:

$$
\begin{aligned}
\frac{d}{d\theta}\log L(\theta \mid \mathscr{D}) &= \frac{d}{d\theta}\left[-N\log\sigma\sqrt{2\pi} - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{x_i-\theta}{\sigma}\right)^2\right] \\
&= -\frac{1}{2}\sum_{i=1}^{N}\frac{d}{d\theta}\left(\frac{x_i-\theta}{\sigma}\right)^2 \\
&= \sum_{i=1}^{N}\left(\frac{x_i-\theta}{\sigma^2}\right).
\end{aligned}
\tag{2.7}
$$

Notice how this gradient points in the direction of $\hat{\theta}_{ML}$, which in this case is given by the sample mean.

Latent Variable Models

The purpose of statistical modeling usually varies between tasks. In some cases the primary goal of learning is *density estimation*, i.e. accurately estimating the probability density function (or pmf in the case of discrete observations) from which the observed data was generated. In other cases, the emphasis is on uncovering 'hidden' latent structure in the data or *representation learning*. Representations are alternative ways of describing the data that focus on particular properties,

which will be of primary interest in this dissertation. The prototypical unsupervised learning techniques to learning representations focus on dimensionality reduction and clustering, which can be expressed using *latent variable models* that additionally include unobserved latent variables.

Consider a latent variable model of the form $p(x, z \mid \theta) = p(x \mid z, \theta)p(z)$, where X captures the observed data, Z the unobserved latent variables, and $\theta \in \Theta$ are the parameters as before. In order to use this latent variable model to model $p(\cdot \mid \theta^*)$ we need to integrate over all possible values that Z can take:

$$p(x \mid \theta) = \int_{\mathcal{Z}} p(x, z \mid \theta) dz = \int_{\mathcal{Z}} p(x \mid z, \theta) p(z) dz. \tag{2.8}$$

The assumptions about the interaction between Z and X dictate the *structure* of the model, and thereby of the learned representation. For example, if we assume that $\boldsymbol{Z}$ is a K-dimensional binary random variable that can take on K possible one-hot encoded states, and that the conditional likelihood $p(x \mid \boldsymbol{Z}_k = 1, \boldsymbol{\theta}_k)$ is Gaussian, then we recover the well-known Gaussian Mixture Model (GMM):

$$p(x \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \mathcal{N}(x \mid \boldsymbol{Z}_k = 1, \mu_k, \sigma_k^2) p(\boldsymbol{Z}_k = 1 \mid \pi_k). \tag{2.9}$$

In this model, we have K sets of parameters $\boldsymbol{\theta} = \{(\mu, \sigma^2)_1, \ldots, (\mu, \sigma^2)_K\}$ for the conditional likelihood and the latent variable $\boldsymbol{Z}$ determines which were used to generate x. The prior $p(\boldsymbol{z} \mid \boldsymbol{\pi})$, having parameters $\boldsymbol{\pi}$, encodes our initial beliefs about the proportion of data assigned to each component. In order to perform inference about $\boldsymbol{z}$ we can compute the posterior $p(\boldsymbol{z} \mid x, \boldsymbol{\theta}, \boldsymbol{\pi})$, which tells us which of the Gaussian components (clusters) x is expected to belong to. Together with the learned cluster centers $\mu_1 \ldots \mu_K$ and variances $\sigma_1^2 \ldots \sigma_K^2$ this provides an alternative description for x. Indeed, we will explore an adaptation of standard GMMs for representation learning in Chapter 4.

Another relevant model that we will make use of in this dissertation is based on the linear Gaussian model. In this case, we assume that $\boldsymbol{Z}$ is Gaussian and that the conditional likelihood $p(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta})$ is also Gaussian, but that its mean linearly depends on $\boldsymbol{z}$ through the matrix $\boldsymbol{W}$ of parameters:

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \int_{\mathcal{Z}} \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{W}\boldsymbol{z}, \boldsymbol{\Sigma}) \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}_{\boldsymbol{z}}, \boldsymbol{\Sigma}_{\boldsymbol{z}}) d\boldsymbol{z}. \tag{2.10}$$

where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{\Sigma})$ and $\boldsymbol{\pi} = (\boldsymbol{\mu}_{\boldsymbol{z}}, \boldsymbol{\Sigma}_{\boldsymbol{z}})$. If $\boldsymbol{x}$ is high-dimensional and $\boldsymbol{z}$ of lower dimension, then we can think of $\boldsymbol{z}$ as being a *compressed* representation of $\boldsymbol{x}$ that can be inferred (after learning $\boldsymbol{\theta}$) by computing $p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\pi})$. Several other

models used for representation learning, such as Factor Analysis and Probabilistic PCA, have a similar functional form [Roweis and Ghahramani, 1999].

Expectation Maximization In principle, we can make use of the same techniques from Bayesian inference to perform unsupervised learning and make predictions when using latent variable models. However, unlike before, $\log p(x \mid \theta)$ now includes an integration (or summation) term, which complicates maximization with respect to θ.

An alternative approach to *Maximum Likelihood Estimation* (MLE) for latent variable models (that is typically easier to work with) can be obtained by formulating a lower-bound based on some auxiliary distribution q:[4]

$$\begin{aligned}\log p(x \mid \theta) &= \log \int_{\mathcal{Z}} p(x, z \mid \theta) dz \\ &= \log \int_{\mathcal{Z}} q(z) \frac{p(x, z \mid \theta)}{q(z)} dz \\ &\geq \int_{\mathcal{Z}} q(z) \log \frac{p(x, z \mid \theta)}{q(z)} dz.\end{aligned} \tag{2.11}$$

where the final expression is obtained by applying Jensen's inequality. This enables us to increase the data log-likelihood $\log p(x \mid \theta)$ and fit the model by maximizing (2.11) with respect to θ.

In order to determine how to choose q, we can make the following observation:

$$\begin{aligned}q^* &= \arg\max_{q(z)} \left[\int_{\mathcal{Z}} q(z) \log \frac{p(x, z \mid \theta)}{q(z)} dz \right] \\ &= \arg\max_{q(z)} \left[\int_{\mathcal{Z}} q(z) \log p(x \mid \theta) + q(z) \log \frac{p(z \mid x, \theta)}{q(z)} dz \right] \\ &= \arg\max_{q(z)} \left[\int_{\mathcal{Z}} q(z) \log \frac{p(z \mid x, \theta)}{q(z)} dz \right] \\ &= \arg\min_{q(z)} D_{KL}[q(z) \,||\, p(z \mid x, \theta)] = p(z \mid x, \theta).\end{aligned} \tag{2.12}$$

where $D_{KL}[q \,||\, p] = \int_{\mathcal{Z}} q(z) \log q(z)/p(z \mid x, \theta) dz$ is the KL divergence, which is minimized if and only if both distributions are identical. Hence, (2.11) is

[4] The conceptually simpler alternative of using Monte Carlo to approximate the expectation $\mathbb{E}_{p(z)}[p(x \mid z, \theta)]$ with a sample average is extremely sample inefficient when x is high-dimensional [Doersch, 2016].

maximized wrt. q when choosing $q(z) = p(z \mid x, \theta)$. In fact, it is easy to see that in this case the lower-bound is tight:

$$\begin{aligned} \log p(x \mid \theta) &\geq \int_{\mathscr{Z}} q(z) \log \frac{p(x, z \mid \theta)}{q(z)} dz \\ &= \int_{\mathscr{Z}} p(z \mid x, \theta) \log p(x \mid \theta) dz = \log p(x \mid \theta). \end{aligned} \tag{2.13}$$

Together, these results suggest a simple iterative algorithm to maximizing $\log p(\mathscr{D} \mid \theta)$ wrt. θ, which is known as the *Expectation Maximization* algorithm [Dempster et al., 1977]. In each iteration, we first compute the *E-step* for each observation $x_i \in \mathscr{D}$ based on the previous parameter estimate θ^{old} (here we assume a single i.i.d. latent variable Z_i for each observation), which tightens the lower-bound in (2.11) while holding the parameters fixed:

$$q_i = \arg\max_{q_i(z_i)} \left[\int_{\mathscr{Z}_i} q_i(z_i) \log \frac{p(x_i, z_i \mid \theta^{\text{old}})}{q_i(z_i)} dz_i \right] = p(z_i \mid x_i, \theta^{\text{old}}). \tag{2.14}$$

Next, we compute the *M-step*, which produces a new parameter estimate by maximizing (2.11) wrt. θ across the data set:

$$\begin{aligned} \theta &= \arg\max_{\theta} \left[\sum_i \int_{\mathscr{Z}_i} q_i(z_i) \log \frac{p(x_i, z_i \mid \theta)}{q(z_i)} dz_i \right] \\ &= \arg\max_{\theta} \left[\sum_i \int_{\mathscr{Z}_i} p(z_i \mid x_i, \theta^{\text{old}}) \log p(x_i, z_i \mid \theta) dz_i \right]. \end{aligned} \tag{2.15}$$

By iterating this EM algorithm until convergence, we are guaranteed to find a local optimum of the data log-likelihood [Wu, 1983]. However, it can sometimes still be difficult to analytically compute (2.15). In this case, we can use a gradient-based approach (similar to before), where we update θ in the direction of steepest ascent by differentiating (2.15) (optionally using a Monte Carlo approximation of the expectation). This procedure, which we will make use of in Chapter 4, is sometimes referred to as *generalized EM*.

Variational Inference

When using latent variable models for representation learning, we will sometimes make us of *Variational Inference (VI)* [Wainwright and Jordan, 2008] (see

Blei et al. [2017]; Zhang et al. [2018a] for more recent overviews). It provides a framework for approximating the posterior distribution over a set of unobserved latent variables $p(z \mid x, \theta)$ by an approximate 'variational' distribution q. In this case, inference can be treated as an *optimization problem* where the goal is to find a good approximate distribution q from a family of distributions $\mathcal{Q}$ over $\mathcal{Z}$ that minimizes some divergence between itself and the true posterior. The standard choice of divergence is the reverse KL-divergence, which yields the following optimization problem:

$$q^* = \arg\min_{q \in \mathcal{Q}} D_{KL}[q(z) \,||\, p(z \mid x, \theta)]. \tag{2.16}$$

The accuracy of the approximation is determined both by the flexibility of the distributions in $\mathcal{Q}$, and by the ability to perform the optimization adequately. Although it is possible to use other divergences to perform variational inference (including the forward KL [Minka, 2001]), we will make use of the reverse KL, which has a closer resemblance to EM, as we will see next.

Note that it is not possible to optimize (2.16) directly due to its dependence on the marginal likelihood $p(x \mid \theta)$ (here due to marginalizing Z), which is usually intractable to compute and the main reason that we can not use the exact posterior in the first place. Indeed, we have that:

$$\begin{aligned} D_{KL}[q(z) \,||\, p(z \mid x, \theta)] &= \int_{\mathcal{Z}} q(z) \log q(z) - q(z) \log p(x, z \mid \theta) dz + \log p(x \mid \theta) \\ &= -L_{ELBO}(q, \theta) + \log p(x \mid \theta), \end{aligned} \tag{2.17}$$

where L_{ELBO} is known as the *Evidence Lower Bound (ELBO)*, defined as:

$$\begin{aligned} L_{ELBO}(q, \theta) &:= \int_{\mathcal{Z}} q(z) \log p(x, z \mid \theta) - q(z) \log q(z) dz \\ &= \int_{\mathcal{Z}} q(z) \log p(x \mid z, \theta) dz - D_{KL}[q(z) \,||\, p(z)]. \end{aligned} \tag{2.18}$$

Because $\log p(x \mid \theta)$ acts as a constant in this optimization problem, we can now optimize (2.16) by maximizing the ELBO wrt. q. Moreover, by moving $L_{ELBO}(q, \theta)$ to the left-hand side in (2.17), it can also be seen how the ELBO forms a valid lower-bound to $\log p(x \mid \theta)$ (hence the name) as the KL-divergence is always positive.

Indeed, we note how the ELBO closely relates to our previous result for EM in (2.11), which was obtained by attempting to estimate the parameters of a latent variable model, i.e. to increase the marginal log-likelihood $\log p(x \mid \theta)$ wrt. θ. Meanwhile, the ELBO was obtained by attempting to find an accurate variational approximation to the posterior $p(z \mid x, \theta)$, i.e. to minimize $D_{KL}[q(z) \,||\, p(z \mid x, \theta)]$ wrt. q. It turns out that these different optimization problems can be related through (2.17) and give rise to a similar objective.

It is straightforward to make use of this connection. In the E-step (2.14) of EM we found that we could tighten the lower-bound on $\log p(x \mid \theta)$ by choosing $q^* = p(z \mid x, \theta)$. Hence, if this is not possible, for example, because it is intractable to compute the exact posterior under the current model, then it can now be seen how we can use a *variational approximation* to the posterior by using the ELBO to optimize q. This gives rise to the *Variational Expectation Maximization* algorithm, which uses (2.18) to minimize the variational objective in (2.16) in the E-step[5].

Choosing a flexible family of variational distributions $\mathcal{Q}$ that also yields tractable optimization of the ELBO may still require careful inspection of the model. An alternative approach is to use "black-box" variational inference using a parametrized family of distributions $\mathcal{Q} = \{q(\cdot \mid \lambda) : \lambda \in \Lambda\}$ [Ranganath et al., 2014]. In this case, we can optimize λ by treating the integral in (2.18) as an expectation and use Monte Carlo approximation to compute stochastic gradients. Unbiased low variance gradients can often be obtained for continuous latent variables by using the 'reparametrization trick' [Kingma and Welling, 2014].

In the standard VI framework, the optimization problem in (2.16) is solved for each sample individually and therefore q_i need not functionally depend on the actual observation. On the other hand, this makes plain VI computationally expensive for learning on large data sets. An alternative approach is to use an *inference model* $f_\phi : x \mapsto \lambda$ (e.g. a neural network having weights ϕ) that *amortizes* variational inference over an entire data set [Gershman and Goodman, 2014]. The inference model outputs variational parameters λ_i for a given observation x_i to implement $q(\cdot \mid x_i, \lambda_i)$. The parameters ϕ of the model are trained to optimize (2.16) for all data points simultaneously.

Variational Auto Encoder A well-known framework, which we will make use of in Chapter 7 for representation learning, is that of *Variational Auto-Encoders* (VAEs) [Kingma and Welling, 2014]. VAEs employ amortized variational inference

[5]Note that the use of VI is not limited to finding ML estimates in latent variable models as is EM. Rather, VI can be used to perform learning in fully Bayesian models, where the model parameters are themselves treated as latent variables that are optimized via (2.16).

using inference models to learn a statistical latent variable model of the observed data. The model is similar to the Linear Gaussian model in that it assumes a continuous vector-valued random variable $\boldsymbol{Z}_i$ for each observation and a conditional likelihood (often Gaussian) whose parameters depend on $\boldsymbol{z}_i$. However, unlike in the linear case, it uses a neural network f_θ to implement this mapping. This makes exact posterior inference intractable and amortized variational inference is used to approximate the posterior. The inference model in this case is also implemented by a neural network f_ϕ, which allows (2.18) to be interpreted as a regularized auto-encoding objective: an observation is 'encoded' by the inference model to obtain $q(\boldsymbol{z}_i \mid f_\phi(\boldsymbol{x}_i))$ (from which we can sample $\boldsymbol{z}_i$) and then 'decoded' by the model to obtain $p(\boldsymbol{x}_i \mid f_\theta(\boldsymbol{z}_i))$ to evaluate the first term. The KL-term then acts as the regularizer. Typically the dependence on the neural network is left implicit and we will write $q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ and $p_\theta(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$ in this case as shorthand.

2.2.2 Classification

In other parts of this dissertation we will make use of supervised learning for classification tasks [Bishop, 2006]. We will also adopt a statistical approach in this case, and assume that $\mathscr{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is a sample drawn from a sequence of random variables $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ that are i.i.d. according to $p(x, y \mid \theta^*)$. Since we are concerned with classification, the Y_i are discrete random variables that each take on one of $\{0, 1, \ldots, C-1\}$ values (class labels). Our goal will be to learn a function $h : \mathscr{X} \mapsto \mathscr{Y}$ (the classifier) which assigns the correct label y to any (new) observation x that is drawn from the corresponding conditional distribution.

In order to decide about h we introduce a *loss function* $L : \mathscr{Y} \times \mathscr{Y} \mapsto \mathbb{R}$, which measures how different the prediction $\hat{y}_i$ is from the true class y_i for an observation x_i. Next, we let the *risk* associated with h be given by:

$$R(h) = \int_{\mathscr{X}} \int_{\mathscr{Y}} L(\hat{y}, y) p(x, y) dy dx. \tag{2.19}$$

This then gives rise to a natural optimization objective for learning $h \in \mathscr{H}$, i.e. find the h that minimizes the risk. In most problems $p(x, y)$ is not available and we will resort to minimizing the *empirical* risk based on $\mathscr{D}$ (i.e. the training data) in stead:

$$R_{emp}(h) = \frac{1}{N} \sum_{i=1}^{N} L(\hat{y}_i, y_i). \tag{2.20}$$

Notice how, in this case of *Empirical Risk Minimization (ERM)*, we are prone to an issue known as *overfitting*. Depending on the function class $\mathscr{H}$, it may be possible to minimize $R_{emp}(h)$ in a way that does not *generalize* to new instances drawn from the same distribution. Similarly, it may occur that the sample $\mathscr{D}$ that is available to us was obtained with low probability (i.e. it is not representative) and can mislead the classifier. The framework of *Probably Approximately Correct (PAC)* learning is concerned with providing guarantees about generalization for a particular hypothesis class in these situations, and we refer the reader to Shalev-Shwartz and Ben-David [2014] for an overview. In this work, we will protect against overfitting by measuring the capacity to generalize explicitly using a separate validation data set, which is standard practice.

There exist multiple approaches to minimizing R_{emp} [Bishop, 2006]. One approach is to learn a *discriminant function* that directly maps inputs to outputs. Another common approach, which we will make use of in this work is to learn a *discriminative model* that models the posterior class probabilities $p(y \mid x, \theta)$ directly (the class label with the highest probability is chosen as the output). In this case, we can choose L as the negative log-likelihood and empirical risk minimization becomes equivalent to maximum likelihood estimation of θ. As we will see, it is natural to train neural networks in this way, i.e. when treating their output for a given input x as a probability distribution over $\mathscr{Y}$. Finally, one can attempt to model the joint $p(x, y \mid \theta)$ in its entirety and use Bayes' rule to derive the posterior class probabilities, which will additionally incorporate uncertainty about θ and help against overfitting. However, similar to before, the issue of obtaining the marginal likelihood prevents us from using this approach in practice.

2.3 Neural Networks

In this disseration, we will focus heavily on machine learning using artificial Neural Networks (NNs). Generally speaking, NNs consist of simple connected nodes (neurons) that each compute an activation. Inputs may be received directly from the environment (e.g. sensory inputs) or indirectly in the form of the outputs (activations) of other nodes in the network. A real-valued multiplicative weight is associated with each connection that, together with its input, determines the activation of each neuron. The activation of a subset of the neurons is treated as the output of the network, which can be adjusted by changing the neural network weights. Hence, it is useful to think of neural networks as simply implementing a mapping from input to output, based on the value of its weights.

In modern neural networks [Schmidhuber, 2015b], neurons are organized in layers, such that each node in a layer is only connected to nodes in the previous layer. Deep neural networks consist of many layers of non-linear transformations and their 'depth' is determined by the number of consecutive layers that are used[6]. Hence, deep neural networks can be viewed as successively transforming the input data to produce the desired output. In this case, the activations of the nodes in each intermediate (or 'hidden') layer provide an alternative representation of the input data, where each successive transformation contributes towards a representation that is better suited to produce the desired output [Lee et al., 2009]. This has recently allowed deep neural networks to overcome the limitations of manual feature engineering in several application domains (e.g. Fernández et al. [2007]; Ciresan et al. [2011]; Hinton et al. [2012]; Ciresan et al. [2012]; Krizhevsky et al. [2012]; He et al. [2017]). More important in the context of this work, is that (deep) neural networks *learn representations* that are also useful for solving other (related) tasks (e.g. Rumelhart et al. [1985]; Schmidhuber [1991a]; Zemel and Hinton [1994]; Bengio et al. [2013]; Zeiler and Fergus [2014]). This allows us to apply them for representation learning purposes directly, which will be our main use case in this work.

2.3.1 Architectures

Many different types of neural network architectures exist. Early approaches were often designed to function as (somewhat) plausible computational models of the human brain and include a number of specific components [Rosenblatt, 1958; Fukushima, 1980]. In contrast, the more modern architectures that we will consider mostly differ in how they combine a number of standard layers that essentially function as composable building blocks. These have little resemblance to how the human brain processes information, although there is evidence that in some cases similar activation patterns can be obtained during information processing [Yamins et al., 2013; Nayebi et al., 2018].

At a fundamental level, one can distinguish between *feedforward neural networks* and *recurrent neural networks*. In feedforward neural networks the first layer receives its input directly from the environment, and the activations of the neurons in the final layer serve as the output of the network. Recurrent Neural Networks (RNNs) additionally incorporate circular dependencies between nodes that are resolved through time. Depending on the choice of architecture and

[6]Recently, this has popularized the term *deep learning* to refer to machine learning using deep neural networks.

availability of data, feedforward neural networks are in principle able to learn any required transformation of the input data [Hornik, 1991; Lu et al., 2017], while an RNN can in principle simulate any Turing machine [Siegelmann and Sontag, 1991].

In the following, we will briefly introduce several standard neural network architectures that we will make use of throughout this dissertation. Variations on these architectures can be obtained by mixing and combining different layer types. We refer the reader to Goodfellow et al. [2016] for a more detailed overview.

Multi-Layer Perceptrons

Fully-connected layers are the basic building block of many modern neural network architectures. In a fully-connected layer, every unit in a layer receives as input the output of every unit in the previous layer. A well-known architecture that consists of L 'hidden' layers and an output layer that are fully-connected is the *Multi-Layer Perceptron* (MLP). In an MLP the j^{th} unit in layer l computes:

$$\boldsymbol{z}_j^{(l)} = \sum_{i=0}^{N-1} \boldsymbol{W}_{i,j}^{(l)} \cdot \boldsymbol{a}_i^{(l-1)} + \boldsymbol{b}_j^{(l)}, \tag{2.21}$$

$$\boldsymbol{a}_j^{(l)} = \sigma(\boldsymbol{z}_j^{(l)}), \tag{2.22}$$

where $\boldsymbol{W}^{(l)} \in \mathbb{R}^{N \times M}$ contains the weights on the edges from units i in layer $l-1$ to units j in layer l, and $\boldsymbol{b}^{(l)} \in \mathbb{R}^M$ contains the bias for units j in layer l. Here $\boldsymbol{a}^{(l-1)} \in \mathbb{R}^N$ is the output of the previous layer $l-1$, where $\boldsymbol{a}^{(0)} = \boldsymbol{x}$ acts as the input layer, and $\hat{\boldsymbol{y}} = \boldsymbol{a}^{(L+1)}$ as the output. Together, the weights and biases of each unit in each layer form the parameters of the MLP.

Activation Functions The quantities $\boldsymbol{z}_j^{(l)}$ and $\boldsymbol{a}_j^{(l)}$ are sometimes referred to as the *pre-activation* and the *activation* of a unit. The latter is obtained by applying an *activation function* σ to each unit. Standard activation functions we will make use of include the *Sigmoid*: $1/(1-e^z)$, the *Hyperbolic Tangent*: $tanh(z)$, the *Rectified Linear Unit* (ReLU): $\max(0, z)$ [von der Malsburg, 1973], and variations on the ReLU, such as the *Leaky ReLU* [Maas et al., 2013] and the *Exponential Linear Unit* (ELU) [Clevert et al., 2015].

It is important to note that when using a *linear* activation function, i.e. $\sigma(z) = z$, the discriminative power of an MLP with multiple hidden layers is equivalent to that of an MLP with a single hidden layer[7]. Hence, σ usually takes the form of a non-linearity in order to make effective use of the depth of the network.

[7]There may still be an advantage to using *multiple* layers in the linear case due to possible

Output Layers The dimensionality of the output layer and its activation function typically depends on the task itself, especially since we will often treat neural network outputs as implementing probability distributions. In a binary classification setting, where inputs x are said to either belong to class $Y = 0$ or $Y = 1$, the output layer consists of a fully-connected layer with a single unit and a Sigmoid activation function. The output of the network $\hat{y}$ can then be interpreted as the *probability* that the input belongs to the first class:

$$p(Y = 0 \mid x) = \hat{y} \quad \text{and} \quad p(Y = 1 \mid x) = 1 - \hat{y}. \tag{2.23}$$

Alternatively, this can be viewed as the network f_θ parametrizing a Bernoulli random variable $Y \sim \text{Bernoulli}(p = f_\theta(x))$. Notice also the similarity between an MLP that computes the output for a given input in this way, and the well-known *logistic regression* model [Bishop, 2006].

In a multi-class classification setting where $\mathcal{Y} = \{0, 1, \ldots, C-1\}$, which we will explore in Chapter 7, we will let the network implement a categorical distribution in a similar fashion. In this case, the activation function needs to consider the pre-activation of all output neurons simultaneously in order to obtain a valid probability mass function. More specifically, for a classification problem having C classes, the output layer consists of a (fully-connected) layer with C units and the following *softmax* activation function [Bridle, 1990]:

$$p(Y = j \mid x) = \text{softmax}(\boldsymbol{z}_j) = \frac{e^{\boldsymbol{z}_j}}{\sum_{j'=0}^{C-1} e^{\boldsymbol{z}_{j'}}}. \tag{2.24}$$

In this case, the possible values Y can take are essentially one-hot encoded by the output neurons.

In other settings, like when using a neural network to implement a conditional Gaussian distribution $\mathcal{N}(x \mid f_\theta(z))$, we will take a similar approach. We will let the output of the neural network correspond to the parameters of the corresponding distribution (in this case μ, σ^2), which will then dictate the choice of activation function or output dimensionality.

Convolutional Neural Networks

When modeling images we will often make use of *Convolutional Neural Networks* (CNNs) [Fukushima, 1979; Waibel et al., 1989; LeCun et al., 1989, 1998]. The basic building block of a CNN is the convolutional layer whose activation $\boldsymbol{A}^{(l)} \in$

differences in error back-propagation and the resulting credit assignment [Baldi and Hornik, 1995].

$\mathbb{R}^{h_l \times w_l \times c_l}$ is given by a three-dimensional volume. Here h_l and w_l are the spatial dimensions whose size is derived from the corresponding spatial dimensions of the input image, and c_l is the *feature* dimension.

The computations of a convolutional layer are best described at the level of two-dimensional activation maps $\boldsymbol{A}_c^{(l)} \in \mathbb{R}^{h_l \times w_l}$. Each map contains the activations of the neurons that are co-located in the feature dimension. Neurons that belong to this same *feature map* share their weights, hence the naming of this dimension as the feature dimension. The activation of a neuron at spatial location (i, j) in $\boldsymbol{A}_k^{(l)}$ is computed as a weighted sum of the activations of the neurons in the previous layer at the same spatial locations (i, j) and the surrounding neighborhood. In other words, the activations in feature map k are obtained by first *convolving* a kernel $\boldsymbol{W}_k^{(l)} \in \mathbb{R}^{k_h, k_w, c_{l-1}}$ with $\boldsymbol{A}^{(l-1)}$ across the spatial dimensions, followed by a sum across the feature dimension (and adding a bias term for each feature). Here k_h, k_w are the width and height of the kernel (hyperparameters) that determine the size of the neighborhood, and c_{l-1} is the number of feature maps in the previous layer.

It is desirable in a CNN to gradually decrease the size of the spatial dimensions with each consecutive layer as a means of increasing their receptive field and encourage more global (hierarchical) features to be learned [Fukushima, 1979; LeCun et al., 1989]. The approach that we will make use of in this case is to use *strided* convolutions, where the kernel is applied only at regular intervals along the spatial dimensions.

Note that it is trivial to combine convolutional layers with fully-connected layers via reshaping. In that sense, convolutional layers are also compatible with the fully-connected output layers discussed previously. Similarly, a convolutional layer may itself act as the output layer by appropriately choosing the output dimensionality and activation function as before.

Weight-sharing across the spatial dimensions as encountered in CNNs provides an inductive bias that is particularly helpful when processing images. In particular, this spatial invariance in lower layers makes it easy to learn features that capture local image statistics, such as edges or other local patterns, that can be re-used across a variety of image locations [Lee et al., 2009; Zeiler and Fergus, 2014]. Meanwhile, higher layers that act on the image more globally (i.e. due to having a larger receptive field) are able to learn more specialized (hierarchical) features, such as a 'face feature' [Lee et al., 2009], that use low-level features as building blocks.

Recurrent Neural Networks

The neural network architectures discussed previously only propagate information forward. A Recurrent Neural Network (RNN) also incorporates cyclic dependencies that are resolved through time [McCulloch and Pitts, 1943; Kleene, 1956; Werbos, 1988; Hochreiter and Schmidhuber, 1997; Gers et al., 2000; Chung et al., 2014]. From this perspective, an RNN can also be viewed as a dynamical system whose state evolves over time and which can function as a type of memory when processing sequential inputs [Schmidhuber, 1992b; Hochreiter and Schmidhuber, 1997].

Although an RNN can incorporate a number of fully-connected, convolutional, or other layers, it includes at least one *recurrent layer*. The most basic RNN consists of a single fully-connected recurrent layer that computes:

$$\boldsymbol{z}_j[t] = \sum_{i=0}^{N-1} \boldsymbol{W}_{i,j} \cdot \boldsymbol{x}_i[t] + \sum_{i=0}^{M-1} \boldsymbol{R}_{i,j} \cdot \boldsymbol{a}_j[t-1] + \boldsymbol{b}_j, \tag{2.25}$$

$$\boldsymbol{a}_j[t] = \sigma(\boldsymbol{z}_j), \tag{2.26}$$

where $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ contains the weights on the edges from units i in the input layer to the hidden units j, $\boldsymbol{R} \in \mathbb{R}^{M \times M}$ contains the weights on the edges from the hidden units i at time-step $t-1$ to hidden units j at time-step t, and $\boldsymbol{b} \in \mathbb{R}^M$ contains the bias for the hidden units j. Here $\boldsymbol{x}[t] \in \mathbb{R}^N$ is the input (layer), $\boldsymbol{z}[t] \in \mathbb{R}^M$ the pre-activations, and $\boldsymbol{a}[t] \in \mathbb{R}^M$ the activations at timestep t. The output of the RNN at timestep t are given by the activations $\boldsymbol{a}[t]$, although more commonly a fully-connected output layer is used to decouple the dimensionality of the hidden state (the memory) from the task under consideration.

The weights of the RNN are shared across time, which is an architectural inductive bias that simplifies learning solutions that are invariant across time. Moreover, this makes it possible to apply and train RNNs on sequences of varying time-length. A common choice of activation function is the Sigmoid, which squashes the pre-activation to $[0, 1]$ and prevents the activations from 'exploding' when increasing the sequence length.

2.3.2 Learning

In order to train neural networks, we will make use of a loss function $L(\boldsymbol{\theta}, \mathscr{D})$ that relates the neural network weights and the observed data. The goal of learning is then to find the neural network weights $\boldsymbol{\theta}$ that minimizes $L(\boldsymbol{\theta}, \mathscr{D})$. Due to the non-linearity of neural networks, it is usually not possible to solve for $\boldsymbol{\theta}$

analytically, which leaves gradient-based optimization or search as alternatives. If $L(\boldsymbol{\theta}, \mathscr{D})$ is differentiable with respect to $\boldsymbol{\theta}$ it is often more efficient to train a neural network using gradients, which is the approach that we will adopt in this work.

In order to derive a suitable loss function, we will normally treat neural networks as part of probabilistic models and train their weights via gradient-based maximum likelihood estimation. In the unsupervised case this means $L(\boldsymbol{\theta}, \mathscr{D}) = -\log p(\mathscr{D} \mid m, \boldsymbol{\theta})$, where the model m is (based on) the neural network. Analogously, in the supervised case we will have $L(\boldsymbol{\theta}, \mathscr{D}) = -\log p(\mathscr{D}_y \mid \mathscr{D}_x, m, \boldsymbol{\theta})$. Depending on the exact model implemented by the neural network, e.g. a Gaussian, or a Categorical distribution, a precise functional form can be derived.

In certain cases, we will additionally incorporate some form of *regularization*. Standard approaches to regularization discourage a NN to learn a solution that is complex, based on some a priori definition of complexity (e.g. a lack of sparsity) that relates to generalization [Krogh and Hertz, 1992; Srivastava et al., 2014]. For other optimization-based regularization techniques that we will make use of it is less clear how the structure they impose relates to generalization and their utility is verified empirically [Ioffe and Szegedy, 2015; Ba et al., 2016b].

Throughout this work, we will split the available data in separate training, validation, and test sets to protect against overfitting. The training set will be used to train the network and normally contains the largest fraction of the available data. The validation set contains a small fraction of the data and is used to evaluate a trained model during the development stage (i.e. during architectural design, or hyperparameter tuning). To report the final results, we will evaluate trained models on the test set.

Backpropagation

To compute gradients for neural network weights, we will make use of the *backpropagation* algorithm [Linnainmaa, 1970; Werbos, 1982]. In this case, gradients are computed by proceeding backward through the network, which is efficient due to the recursive application of the chain-rule. In the following, we will briefly discuss backpropagation for fully-connected feedforward networks and RNNs. A similar derivation follows for other layer types, such as convolutional layers, and we refer to Goodfellow et al. [2016] for a more detailed overview.

Feedforward Networks For a fully-connected feedforward neural network, backpropagation proceeds as follows. First, let us define the error of neuron i seperately

for each datapoint in $\mathscr{D}$ as:[8]

$$\boldsymbol{\delta}_i^{(l)} := \frac{\partial L(\boldsymbol{\theta}, \mathscr{D})}{\partial \boldsymbol{z}_i^{(l)}}. \tag{2.27}$$

Although this term is easy to compute via the chain rule for neurons in the last layer, it is more difficult to compute for lower layers. In particular, there exist many different ways (involving different 'credit assignment paths' [Schmidhuber, 2015b] through the network) in which a neuron at a lower level can influence the neurons at the higher levels, which all must be accounted for. This is where backpropagation comes in, namely through computing $\boldsymbol{\delta}_i^{(l)}$ recursively via repeated application of the chain-rule for intermediate layers:

$$\begin{aligned}
\boldsymbol{\delta}_i^{(l)} &= \frac{\partial \boldsymbol{a}_i^{(l)}}{\partial \boldsymbol{z}_i^{(l)}} \frac{\partial L(\boldsymbol{\theta}, \mathscr{D})}{\partial \boldsymbol{a}_i^{(l)}} \\
&= \sigma'(\boldsymbol{z}_i^{(l)}) \sum_{j=0}^{N-1} \frac{\partial L(\boldsymbol{\theta}, \mathscr{D})}{\partial \boldsymbol{z}_j^{(l+1)}} \frac{\partial \boldsymbol{z}_j^{(l+1)}}{\partial \boldsymbol{a}_i^{(l)}} \\
&= \sigma'(\boldsymbol{z}_i^{(l)}) \sum_{j=0}^{N-1} \boldsymbol{\delta}_j^{(l+1)} \cdot \boldsymbol{W}_{i,j}^{(l+1)}.
\end{aligned} \tag{2.28}$$

Notice how, when proceeding *backwards* through the network to compute (2.28) starting from $\boldsymbol{\delta}_i^{(L+1)}$ (which can easily be computed for the output units, e.g. as in (2.7)) we are able to compute $\boldsymbol{\delta}_j^{(l)}$ for all other layers efficiently. In this case, backpropagation requires only a single computation for each neuron when additionally storing the pre-activations $\boldsymbol{z}$ that were computed in the 'forward pass' (i.e. when computing the output of the network) in memory. Once we have obtained the errors for each neuron it is straightforward to compute the partial derivatives:

$$\frac{\partial L(\boldsymbol{\theta}, \mathscr{D})}{\partial \boldsymbol{W}_{i,j}^{(l)}} = \boldsymbol{\delta}_j^{(l)} \cdot \boldsymbol{a}_i^{(l-1)} \quad \text{and} \quad \frac{\partial L(\boldsymbol{\theta}, \mathscr{D})}{\partial \boldsymbol{b}_j^{(l)}} = \boldsymbol{\delta}_j^{(l)}. \tag{2.29}$$

Recurrent Neural Networks The backpropagation algorithm can be extended to efficiently compute gradients for RNNs, although in this case, we need to consider credit assignment paths that flow both through depth and through time.

[8] Since $\mathscr{D}$ is an i.i.d. sample, $p(\mathscr{D} \mid m, \theta)$ factorizes and $\log p(\mathscr{D} \mid m, \theta)$ becomes a sum, such that gradients can be computed for each individual datapoint separately.

The Backpropagation Through Time (BPTT) algorithm [Werbos, 1988; Williams, 1989] resolves this issue by essentially 'unrolling' the computational graph of an RNN for a fixed number of time steps. Like that, it can be treated similar to a feedforward network of depth T, but that receives an additional input at each 'layer'.

Similar to backpropagation, BPTT proceeds *backwards* in depth (and in time) to compute errors $\boldsymbol{\delta}_i^{(l)}[t]$ from which one can efficiently compute gradients:

$$\boldsymbol{\delta}_i^{(l)}[t] = \sigma'(\boldsymbol{z}_i^{(l)}[t]) \left[\sum_{j=0}^{N-1} \boldsymbol{\delta}_j^{(l+1)}[t] \cdot \boldsymbol{W}_{i,j}^{(l+1)} + \sum_{j=0}^{M-1} \boldsymbol{\delta}_j^{(l)}[t+1] \cdot \boldsymbol{R}_{i,j}^{(l)} \right]. \tag{2.30}$$

where $\boldsymbol{\delta}_i^{(l)}[t] := \partial L(\boldsymbol{\theta}, \mathscr{D}) / \partial \boldsymbol{z}_i^{(l)}[t]$. Notice the similarity to (2.28), where the main difference in (2.30) is due to the addition of a second term that accounts for the temporal dependencies in the recurrent case. Finally, we note how additional care must be taken when computing the partial derivatives in this case, since weights are shared across time.

Optimization

The gradients obtained previously can be used to improve the current parameter estimate (weights) in several ways. A straightforward choice is to make use of gradient descent, which is a simple iterative optimization algorithm that updates the parameters in the direction of the gradient:

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \alpha \frac{\partial L(\boldsymbol{\theta}^{\text{old}}, \mathscr{D})}{\partial \boldsymbol{\theta}^{\text{old}}}. \tag{2.31}$$

The step-size α, also known as the *learning rate*, determines the magnitude of the increment. Rather than using the full gradient, computed across the entire data set, we will normally make use of *stochastic* gradients that are obtained by sampling a batch of data from $\mathscr{D}$ to compute the gradient. The resulting Stochastic Gradient Descent (SGD) algorithm is known to find good solutions of $\boldsymbol{\theta}$ when using deep neural networks, even though the loss landscape is typically highly non-convex.

An obvious direction to improve upon SGD is by computing second-order gradients, however, the incurred variance usually outweighs performance, especially in high-dimensional settings. A more common approach is to incorporate some form of *momentum* to accelerate convergence speed [Polyak, 1964]. In our work, we will often make use of ADAM [Kingma and Ba, 2015], which computes

individual adaptive learning rates for different weights based on estimates of the first and second moments of the gradients.

Chapter 3

Challenges & Related Work

The techniques discussed in Chapter 2 provide a method for unsupervised representation learning using neural networks. By using neural networks to define statistical models, it becomes natural to optimize their weights via maximum likelihood estimation to model the observed data and learn representations. This combination of statistical modeling and deep neural networks is particularly fruitful for learning representations of high-dimensional input data, such as images, that are otherwise difficult to model using a simpler choice of (latent variable) model.

Previously, we have discussed how the utility of representations learned in this way is contingent on having a suitable *inductive bias* [Mitchell, 1980]. In deep learning, the inductive bias is determined by a combination of factors, including the choice of neural network architecture (i.e. the function class) and the learning objective. Incorporating the right inductive bias for representation learning is challenging since it requires considering what properties make a particular representational format desirable and how they can be translated into a particular choice of neural network architecture or learning objective. In the unsupervised case, one must additionally ensure that the learned representations are sufficiently general and thereby useful for solving a variety of potential 'down-steam' tasks.

In this chapter, we will discuss these challenges in detail and review related work. Since the focus of this dissertation is on representation learning for visual reasoning tasks, which was motivated in Chapter 1, we will center our discussion around *object representations* and the notion of *disentanglement*.

Sections 3.1 to 3.4 of this chapter are based on Greff et al. [2020], which was first presented as a workshop paper at *ICML 2019* [van Steenkiste* and Greff* et al., 2019]. These works were done in close collaboration and both authors made substantial contributions.

Object Representations act as basic 'building blocks' for neural processing to behave symbolically. Like symbols, they are self-contained and separate from one another such that they can be related and assembled into structures without losing their integrity. But unlike symbols, they retain the advantages of distributed feature-based internal structure of connectionist representations. Object representations are motivated by the fact that human perception is structured around objects [Spelke and Kinzler, 2007], which serve as compositional building blocks for many aspects of higher-level cognition, including reasoning.

Evidence indicates that existing neural networks lack a suitable inductive bias to acquire the ability to process information symbolically in this way (e.g. [Lake and Baroni, 2018; Santoro et al., 2018b]). We propose that this is because of the *binding problem*: The inability to dynamically and flexibly bind information that is distributed throughout the network. The binding problem affects their ability to form meaningful entities from unstructured sensory inputs (*segregation*), to maintain this separation of information at a representational level (*representation*), and to use these entities to construct new inferences, predictions, and behaviors (*composition*).

The primary contribution of Sections 3.1 to 3.4 in this chapter is to introduce the binding problem in neural networks and work towards a solution. We will identify several important challenges and requirements that must be addressed (some of which we will address in later chapters) and review relevant mechanisms from the machine learning literature. Together, these provide a starting point for identifying the right inductive bias to enable neural networks to incorporate *object representations* and generalize more systematically.

Representations that are Disentangled adopt a particular representational format where there is a local correspondence between informative 'factors of variation' of the input and the activation of (individual) neurons at a representational level. Disentangled representations exploit the insight that complex high-dimensional data, such as the visual appearance of an object, can be compactly described using a number of informative features that are general and that focus on separate properties. A representational format that separates information about factors of variation is here viewed as *complementary* to incorporating object representations in this regard, and is expected to offer additional benefits in terms of sample efficiency and generalization to unknown feature combinations. In Section 3.5, we will review the challenge of learning representations that are disentangled and include a brief overview of existing approaches.

3.1 The Binding Problem

We begin our discussion of object representations by reviewing the importance of symbols as units of computation and highlight several symptoms that point to the lack of emergent symbolic processing in existing neural networks. We argue that this is a major obstacle for achieving human-level generalization and posit that the binding problem in neural networks is the underlying cause of this weakness. This section serves as an introduction of the binding problem and provides the necessary context for the subsequent in-depth discussion of its aspects in Sections 3.2 to 3.4.

3.1.1 Importance of Symbols

Human capacity to comprehend reaches far beyond their direct experiences. We can reason causally about unfamiliar scenes, understand novel sentences with ease, and use models and analogies to make predictions about entities far outside the scope of everyday reality, like atoms, and galaxies. This seemingly infinite expressiveness and flexibility of human cognition have long fascinated philosophers, psychologists, and AI researchers alike. The best explanation for this remarkable cognitive capacity revolves around symbolic thought: the ability to form, manipulate, and relate mental entities that can be processed like symbols. By decomposing the world in terms of abstract and reusable 'building blocks', humans are able to understand novel contexts in terms of known concepts, and thereby leverage their existing knowledge in nearly infinite ways. This *compositionality* underlies many high-level cognitive abilities such as language, causal reasoning, mathematics, planning, analogical thinking, etc.

Human understanding of the world in terms of objects develops at an early age [Spelke and Kinzler, 2007] and infants as young as five months old seem to understand that objects continue to exist in the absence of visual stimuli (object permanence) [Baillargeon et al., 1985]. Arguably, this decoupling of mental representation from direct perception is the first step towards a compositional description of the world in terms of more abstract entities. By the age of eighteen months, young children have acquired the ability to use gestures symbolically to refer to objects or events [Acredolo and Goodwyn, 1988]. This ability to relate sensory entities is then key to the subsequent development of language. As the child grows up, entities become increasingly more general and start to include categories, concepts, events, behaviors, and other abstractions, together with a growing number of universal relations such as "same", "greater than", "causes", etc. This growing set of composable building blocks yields an increasingly more

powerful toolkit for constructing structured mental models of the world [Johnson-Laird, 2010].

The underlying compositionality of such symbols is equally potent for AI, and numerous methods that model intelligence as a symbol manipulation process have been explored. Early examples included tree-search over abstract state spaces (e.g. General Problem Solver [Newell et al., 1959]) for theorem proving, or chess [Campbell et al., 2002]; Expert systems that made use of decision trees to perform narrow problem solving for hardware design [Sollow, 1987] and medical diagnosis [Shortliffe et al., 1975]; Natural language parsers that used a dictionary and a fixed set of grammatical rules to interpret written English; And knowledge bases such as semantic networks (networks of concepts and relations) that could be used to answer basic questions [Weizenbaum, 1966], solve basic algebra word problems [Bobrow, 1964], or control simple virtual block worlds [Winograd, 1971]. All of these examples of *symbolic AI* relied on manually designed symbols and rules of manipulation, which allowed them to generalize in *predictable* and *systematic* ways. Since then, many of these approaches have become part of the standard computer-science toolbox[1].

3.1.2 Symbolic Processing in Connectionist Methods

Connectionism takes a different, brain-inspired, approach to Artificial Intelligence that stands in contrast to symbolic AI and its focus on the conscious mind [Fodor, 1975; Newell and Simon, 1981]. Rather than relying on hand-crafted symbols and rules, connectionist approaches such as neural networks focus on *learning* suitable distributed representations directly from low-level sensory data. In this way, neural networks have resolved many of the problems that haunted symbolic AI, including their brittleness when confronted with inconsistencies or noise, and the prohibitive amount of human engineering and interpretation that would be required to apply these techniques on more low-level perceptual tasks. Importantly, the distributed representations learned by neural networks are directly grounded in their input data, unlike symbols whose connection to real-world concepts is entirely subject to human interpretation (see *symbol grounding problem* [Harnad, 1990]). Modern neural networks have proven highly successful and superior to symbolic approaches in perceptual domains, such as in visual object recognition [Ciresan et al., 2011, 2012; Krizhevsky et al., 2012] or speech recognition [Fernández et al., 2007; Hinton et al., 2012], and even in some

[1] They are hardly called AI anymore since it is now well understood how to solve the problems that they address. This redefinition of what constitutes AI is sometimes called the *AI effect*, summarized by Douglas Hofstaedter as "AI is whatever hasn't been done yet".

inherently symbolic domains such as language modelling (e.g. BERT [Devlin et al., 2019], GPT2 [Radford et al., 2019]), translation [Wu et al., 2016], board games [Silver et al., 2017], and symbolic integration [Lample and Charton, 2020].

On the other hand, it has become increasingly evident that neural networks fall short in many aspects of human-level generalization, including those that symbolic approaches exhibit by design. For example, it is difficult for neural language models to generalize syntactic rules such as verb tenses or embedded clauses in a systematic manner [Lake and Baroni, 2018; Loula et al., 2018; Hupkes et al., 2019; Keysers et al., 2020]. Similarly, in vision, neural approaches often learn overly specialized features that do not easily transfer to different data sets or held-out combinations of attributes [Yosinski et al., 2014; Atzmon et al., 2016; Santoro et al., 2018b]. In reinforcement learning, where the use of neural networks has led to superhuman performance in gameplay [Mnih et al., 2015; Silver et al., 2017; Berner et al., 2019], it is found that agents are fragile under distributional shift [Kansky et al., 2017; Zhang et al., 2018b; Gamrian and Goldberg, 2019] and require substantially more training data than humans [Tsividis et al., 2017]. These failures at systematically reusing knowledge suggest that neural networks do not learn a compositional knowledge representation. In some cases, such as in vision, it may appear that object-level abstractions can emerge naturally as a byproduct of learning [Zhou et al., 2015]. However, it has repeatedly been shown that such features are best understood as "a texture detector highly correlated with an object" [Sundararajan et al., 2017; Ancona et al., 2017; Geirhos et al., 2019; Brendel and Bethge, 2019; Olah et al., 2020]. In general, evidence indicates that neural networks learn mostly about surface statistics (eg. between textures and object categories in images) in place of the underlying concepts [Karpathy et al., 2015; Jo and Bengio, 2017; Lake and Baroni, 2018].

A hybrid approach that combines the seemingly complementary strengths of neural networks and symbolic approaches may help address these issues, and several variations have been explored [Bader and Hitzler, 2005]. A common variant uses a neural network as a perceptual interface (or pre-processor) tasked with learning symbols from raw data, which then serve as input to a symbolic reasoning system (e.g. Mao et al. [2019]). Similarly, 'bottom-up' neural networks have been used to make inference more tractable in probabilistic generative models that contain the desired symbolic structure (eg. in the form a symbolic graphics renderer [Kulkarni et al., 2015a]). Neural networks have also been combined with search-based methods to improve their efficiency [Silver et al., 2016]. Countless other variations that vary in terms of the division of labor between the symbolic and neural components, and the choice of the mechanism used to couple them, are possible [McGarry et al., 1999; Bader and Hitzler, 2005].

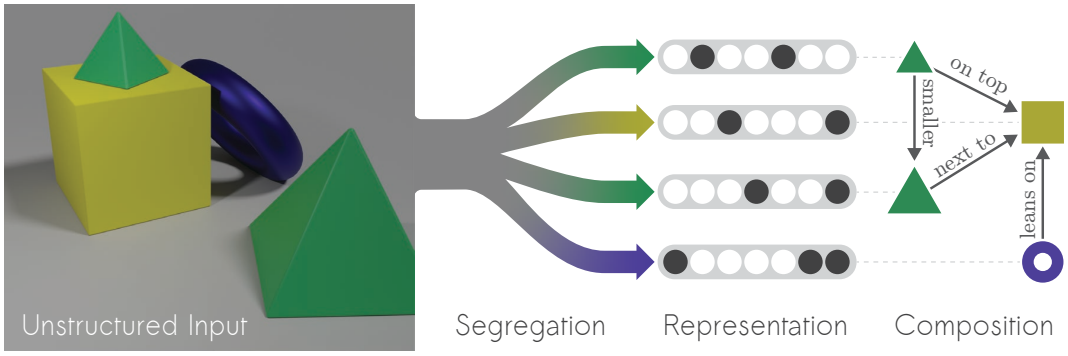


Figure 3.1. The binding problem in neural networks can be understood in terms of *segregation*, *representation*, and *composition*, which each focus on a different functional aspect of binding neurally processed information in the context of facilitating more symbolic information processing.

In this dissertation, we will adopt a more unified approach that addresses these problems from within the framework of connectionism. It is concerned with incorporating a suitable inductive bias in neural networks that enable them to efficiently learn about symbols and the processes for manipulating them. We believe that this is advantageous for several reasons. Firstly, it reduces the required amount of task-specific engineering[2], and helps generalize to domains where expert knowledge is not available. Secondly, by tightly integrating multiple different layers of abstraction they can continuously co-adapt, which avoids the need for rigid interfaces. We also note that, as is evident from the brain, it is sufficient to simply *behave* as an emergent symbol manipulator, and therefore explicit (discrete) symbolic structure is not a requirement.

3.1.3 The Binding Problem in Connectionist Methods

We claim that there exists an underlying cause for the lack of emergent symbolic processing in neural networks, which we refer to as the binding problem. The binding problem is about the inability to dynamically and flexibly combine information that is distributed throughout the network, which is required to effectively form, represent, and relate symbol-like entities. In regular neural networks, information routing is largely determined by the architecture and weights, both of which are fixed at training time. This limits their ability to dynamically route

[2] This leaves the question of the innateness of aspects like causality or three-dimensional space open. Such priors might be helpful or eventually even necessary, however, an intelligent system must also be capable of independently discovering and using novel concepts and structures.

information based on a particular context and thereby to provide for certain patterns of generalization.

The binding problem originates from neuroscience, where it is about the explanatory gap in our understanding of information processing in the brain. It includes perceptual binding problems such as visual binding (color, shape, texture), auditory binding (a voice from a crowd), binding across time (motion), cross-modal binding (sound and vision into joint events), motor-behavior (an action) and sensorimotor binding (hand-eye coordination) [Treisman, 1996; Roskies, 1999; Feldman, 2013]. Another class sometimes referred to as cognitive binding problems, includes binding semantic knowledge to a percept, memory reconstruction, and variable binding in language and reasoning[3].

In the case of neural networks, the binding problem is not just a gap in understanding but rather characterizes a limitation of existing neural networks. Hence, it poses a concrete implementation challenge to address the need for binding neurally processed information, which is common to all of the above subproblems. To tackle this challenge for the purpose of faciliating more symbolic information processing, we propose a *functional division* of the binding problem (inspired by Treisman [1999]) into three aspects: *representation*, *segregation*, and *composition* (Figure 3.1).

The Representation Problem is about encoding relevant information in a way that combines the richness of neural representations with the compositionality of symbols. It revolves around *object representations*, which act as basic building blocks for neural processing to behave symbolically. We chose the term "object" representation because it is evocative of physical objects, which are processed as symbols in many important cognitive tasks, such as visual reasoning. However, we emphasize that object representations are in principle also meant to encode non-visual entities such as spoken words, imagined or remembered entities, and even more abstract entities such as categories, concepts, behaviors, and goals[4].

Interestingly, even the seemingly basic task of incorporating object representations in neural networks faces several problems, such as the "superposition catastrophe" [Von Der Malsburg, 1986] portrayed in Figure 3.2. It suggests that fully-

[3]The term binding problem has also been used in the context of consciousness, namely as the problem of how a single unitary experience arises from the distributed sensory impressions and processing in the brain [Singer, 2001].

[4] We have considered a number of other terms for "object" representations, including entity, gestalt, icon, and concept, which perhaps better reflect their abstract nature but are also less accessible at an intuitive level. The fact that objects are more established in the relevant literature gave them the final edge.

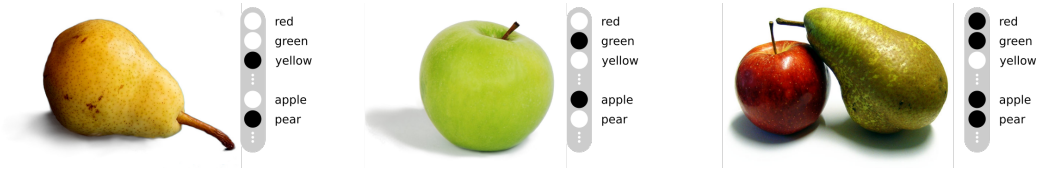


Figure 3.2. Illustration of the *superposition catastrophe*: A distributed representation in terms of disentangled features like color and shape (a, b) leads to ambiguity when confronted with multiple objects (c): The representation in (c) could equally stand for a red apple and green pear, or a green apple and red pear. It leads to an indiscriminate bag of features because there is no association of features to objects. A simple form of this problem in neural networks was first pointed out in Rosenblatt [1961] and has been debated in the context of neuroscience since [Milner, 1974; von der Malsburg, 1981].

connected neural networks suffer from an "inherent tradeoff between distributed representations and systematic bindings among units of knowledge" [Hummel and Holyoak, 1993]. A general treatment of object representation in neural networks involves addressing the superposition catastrophe and other challenges, which we discuss in Section 3.2.

The Segregation Problem is about the process of forming grounded object representations from raw unstructured inputs. It relates to the problem of instance segmentation in that it also produces a division of the input into meaningful parts, but it is complicated by the fact that it is concerned with objects in their most general form. Indeed, the notion of an object is context- and task-dependent, and difficult to formalize even for concrete objects like a tree, a mirror, or a house, which are self-evident to humans. The incredible variability among objects makes it intractable to resolve this issue through supervision. Hence, the segregation problem (Section 3.3) is about enabling neural networks to acquire an appropriate, context- and task-dependent, notion of objects in an unsupervised fashion.

The Composition Problem is about leveraging the modularity of object representations to build structured models for inference, prediction and behavior that generalize in predictable and systematic ways. This relies on the ability to learn abstract relations that can be arbitrarily and recursively applied to object representations, not unlike the way variables can be bound to placeholder symbols in a mathematical expression. Often the desired structure is not known in advance and has to be inferred and adapted to a given context or task. To

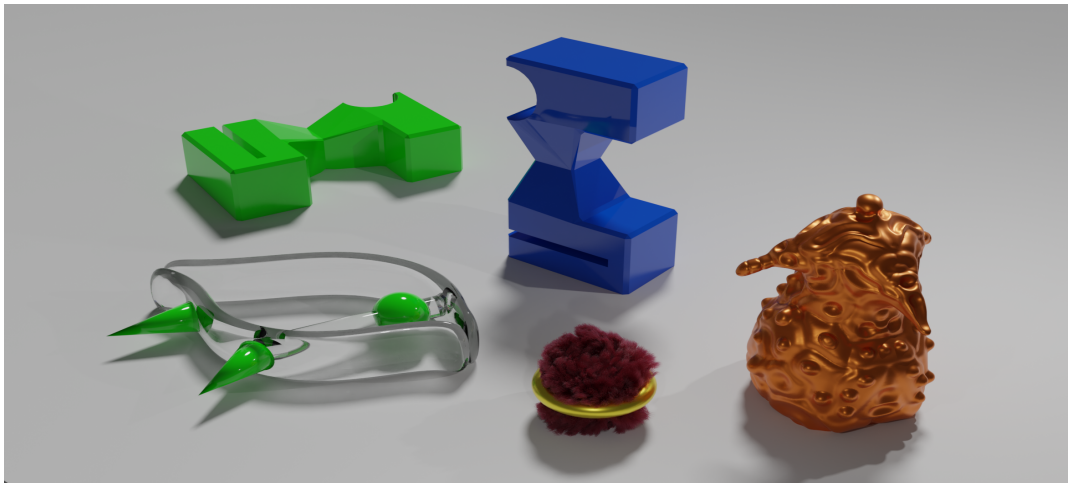


Figure 3.3. A visual scene composed of various unseen objects.

address the composition problem (Section 3.4), a neural network thus requires a mechanism that provides the flexibility to quickly restructure its information flow and that ultimately enables it to leverage object representations to generalize systematically.

3.2 Representation

We have argued that exploiting the benefits of more symbolic information processing using neural networks, requires some form of object representations that combine the richness of neural representations with the compositionality of symbols. These object representations are intended as modular 'building blocks' from which to efficiently compose structured models of the world. This provides several requirements for the representational format and underlying dynamics.

Consider for example Figure 3.3, where you are clearly able to distinguish between five different objects. You can readily describe each object in terms of their shape, color, material, and other properties, despite most likely never having encountered them before. Notice also how these properties relate to individual objects as opposed to the entire scene, which is also evident from the fact that you can tell that the color green occurs multiple times for different objects. Finally, notice how you are readily able to perform comparisons, for example, to tell that the shape of the blue object is the same as that of the green one in the back, but that they differ in color.

In this section we take a closer look at the *format* of object representations

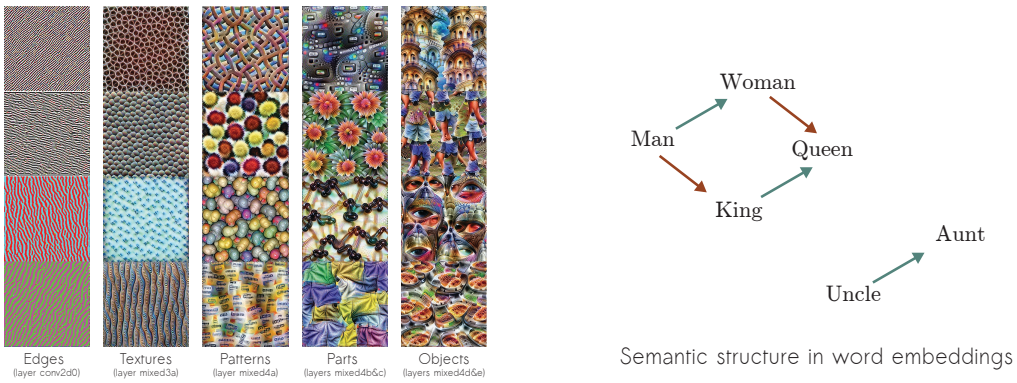


Figure 3.4. **Left:** Interpretable features learned on ImageNet as observed in Olah et al. [2017]. **Right:** Semantic structure of text emerges among learned word embeddings [Mikolov et al., 2013] (although to lesser extent than initially reported [Nissim et al., 2019]).

(Section 3.2.1). We work towards a format that separates information about objects and is general enough to accommodate unfamiliar objects in a meaningful way so that they can readily be compared. Additionally, we will consider the representational *dynamics* that are required to support stable and coherent object representations over time (Section 3.2.2). Towards the end, we review relevant approaches from the literature that may help incorporate these aspects of object representations into neural networks (Section 3.2.3).

3.2.1 Representational Format

We seek a representational format that distinguishes objects, while retaining the advantages of learned distributed representations by modern deep neural networks. These representations have proven highly successful and are known to partially capture the semantic structure of a task (Figure 3.4), such as interpretable image features [Zeiler and Fergus, 2014; Olah et al., 2020], or the semantic structure of text [Mikolov et al., 2013] (but compare Nissim et al. [2019]). In this way, learned object representations can also benefit from existing techniques that focus on feature hierarchies [Lee et al., 2009], equivariance to symmetry transformations [Cohen et al., 2019], spatio-temporal coherence [Becker and Hinton, 1992], sparsity [Olshausen and Field, 1996], non-euclidean feature spaces [Nickel and Kiela, 2017], and in particular *disentanglement* as we will discuss in Section 3.5.

Separation

In order to support the construction of structured models, object representations need to act as modular building blocks. This requires information about individual objects to remain separated at a representational level, such that their features do not interfere with one another, even when composed. Additionally, the features that belong to an object must be able to act as a unit, which implies strong dependencies between its features. For example, when an object representation appears or ceases to exist, all of its features are equally affected.

The separation of information has to be flexible enough to ensure that objects can be formed from novel (unseen) feature combinations. Hence, it is important that it is not purely determined by the representational content of the objects but rather acts as an independent degree of freedom. Regarding the capacity of this working memory, it may suffice to represent only a few objects simultaneously, even though a typical scene potentially contains a large number of objects. Indeed, the capacity of the human working memory is normally believed to only be around 3–9 objects [Miller, 1956; Fukuda et al., 2010], where the observed variability is due to differences in the amount of mental resources required to track various objects [Alvarez and Franconeri, 2007; Frank et al., 2008].

Common Format

To be able to efficiently relate and compare a wide variety of object representations they must be described in a common format. Recall how in Figure 3.3 you were able to freely compare several unfamiliar objects in terms of their properties, such as their size, shape, and location. On the one hand, this is possible because you have acquired a number of general relationships, such as bigger than, left of, etc., which we will discuss in detail in Section 3.4. What is more important here is that such relations can only be applied if object representations provide a shared interface. More generally, a common format helps to ensure that *any* learned relation, transformation, or skill (like grasping) transfers between similar objects independent of context. Similarly, a common set of features helps carry over experiences between objects during learning.

3.2.2 Representational Dynamics

When interacting with the real world, the input stream of sensory information continuously evolves. It is therefore important not only to consider instantaneous representations, but also their *dynamics* over time.

Temporal Dynamics

An object representation requires ongoing updates across time in several cases. Firstly, with objects constantly moving and transforming in the real world, their corresponding representations need adjustments to remain accurate. Secondly, certain temporal attributes such as movement or behavior can only be estimated when considering the history of information. Finally, with the limited amount of information that can be observed about an object at any given time, accumulating information over multiple partial views can help produce more informative object representations.

An important aspect of all these cases is the need for an object representation to consider not only the input but also its history (recurrence). This requires a stable identity to help ensure that information across time-steps is associated with the correct object representation. Note that the identity of an object cannot be tied exclusively to its visible properties, as illustrated by the extreme example of a fairy-tale prince that is transformed into a frog [Marcus, 2003; BamBini et al., 2012].

Reliability

Structured mental models critically depend on object representations (iconic representations) to provide a stable foundation for reasoning and other types of information processing [Johnson-Laird, 2010]. The reliability of this foundation is especially important for more abstract computations to whom such representations provide the only connection to the world. However, perfect reliability is unattainable since sensory information about the world is noisy and incomplete, and the capacity of any model is inherently limited.

Explicitly quantifying uncertainty can help mitigate this issue and prevent noise and errors from accumulating undetectably. In addition, certain small amounts of noise in an object representation may be continually corrected by leveraging dependencies among its features (i.e. through the features of an object acting as a unit). An important source of uncertainty accumulation is due to objects that are temporarily not perceived (e.g. as a result of occlusion). In this case, a 'self-correcting' representation may help maintain a stable object representation, even in the absence of sensory input (object permanence).

Uncertainty about object representations may also arise due to ambiguous inputs that allow for several distinct but coherent interpretations. The ability to (at least implicitly) encode multi-modal uncertainty is crucial to effectively treat such cases. Top-down feedback may then help disambiguate different interpretations (see also Section 3.3.2).

3.2.3 Methods

In order to fulfill the desiderata outlined above, we require a number of specialized mechanisms to implement a corresponding inductive bias. Indeed, it should now also be clear that a simple MLP falls short at adequately representing multiple objects simultaneously: If it attempts to avoid the superposition catastrophe by learning features that are specific to each object, then they lack a common format and become difficult to compare[5]. Therefore, in the following, we will review several techniques for representing multiple objects in neural networks. Our focus is in particular on mechanisms for separation of information, which is critical to incorporating object representations in neural networks, yet often overlooked.

Slot-based Approaches

The simplest approach to separation is to provide a separate representational 'slot' for each object. This provides a (typically) fixed capacity working memory with independent object representations that can all be accessed simultaneously. Weight sharing can then be used to ensure a common format among the individual slots.

Instance Slots In their most general form, which we call instance slots, all slots share a common format and their information can be kept separate independent of their representational content[6]. Instance slots are very flexible and general in that they have no preference for content or ordering. However, this generality introduces a *routing problem* when a common format is obtained via weight sharing: with all slots being identical, bottom-up information processing needs to break this symmetry to avoid assigning the same content to each slot. Hence, in this case, the allocation of information to each slot must be determined by taking the other slots into account, which complicates the process of segregation. Instance slots have been used in several approaches to learning object representa-

[5] Others have suggested ways in which MLPs could *in principle* circumvent this problem [Pollack, 1990; O'reilly and Busby, 2002]. However, neither of these offer a solution that is able to convincingly fulfill all of the above desiderata simultaneously. For plain RNNs it was found that when they are trained to explicitly remember multiple objects internally, they resort to more localist representations [Bowers et al., 2014].

[6] The notion of a finite number of discrete 'instance slots' as working memory closely relates to the traditional notion of visual working memory used in psychology [Baddeley and Hitch, 1974; Luck and Vogel, 1997], although it was later shown that this format does not capture human capacity for object-tracking well [Alvarez and Franconeri, 2007; Frank et al., 2008] (but see Barton et al. [2009]).

tions, including Masked Restricted Boltzman Machines (M-RBMs [Le Roux et al., 2011]) and IODINE [Greff et al., 2019]. We will also make use of instance slots ourselves in Neural Expectation Maximization (N-EM [Greff* and van Steenkiste* et al., 2017a] and Chapter 4) and its relational extension (R-NEM [van Steenkiste et al., 2018a] and Chapter 5). Instance slots can also be found in the memory of memory-augmented neural networks [Graves et al., 2016; Joulin and Mikolov, 2015], in self-attention models [Vaswani et al., 2017; Dehghani et al., 2019], and in certain graph neural networks [Battaglia et al., 2018], where they are treated as internal representations that can be accessed simultaneously (via attention).

Sequential Slots Sequential slots break slot symmetries by imposing an order on the representational slots, typically across time. They are commonly found in RNNs and, when paired with an attention mechanism that attends to a different object at each step, can serve as object representations. With weights typically being shared across (time)steps, sequential slots naturally share a common format and unlike other slot-based representations can dynamically adjust their representational capacity. Sequential slots in RNNs have been used as object representations, for example in Attend Infer Repeat (AIR [Eslami et al., 2016]) and to a lesser degree in DRAW [Gregor et al., 2015]. However, due to recurrent connections, these slots may not always be fully independent, which impedes their function as modular building blocks. Recent approaches, such as Multi-Object Networks (MONet [Burgess et al., 2019]) and GENESIS [Engelcke et al., 2019], alleviate this by using recurrence only for information routing but not for the object representations themselves. In general, a potential limitation of sequential slots is that they are not simultaneously accessible at any given (time-)step for down-stream processing. This can be addressed via a set function over sequential slots, such as the attention mechanism in certain neural machine translation methods [Bahdanau et al., 2014] or pointer networks [Vinyals et al., 2015].

Spatial Slots In spatial slots, each slot is associated with a particular spatial coordinate (e.g. in an image), which helps to break slot symmetries and simplifies information routing. They can still accommodate a common format through weight-sharing, but lack generality because their content is tied to a specific spatial location. Because location and separation are entangled, changes to the location of an object potentially correspond to a change of slot, which complicates maintaining object identity across time. Spatial slots are commonly found in CNNs, where multiple convolutional layers share filter weights across the spatial dimensions to yield a spatial map of representational slots. Although they are not

usually advertised as object representations in this way, several recent approaches, such as Relation Networks [Santoro et al., 2017], the Multi-Entity VAE [Nash et al., 2017], or the work by Zambaldi et al. [2019] explicitly treat each spatial position in the filter-map of a CNN as a candidate object representation. Even more recent approaches, such as SPAIR [Crawford and Pineau, 2019], SPACE [Lin et al., 2020], and SCALOR [Jiang et al., 2020], expand on this by incorporating explicit features for the presence of an object and its bounding box into each spatial slot. Nonetheless, a current limitation of these approaches is that their spatial slots are only suitable for objects that are reasonably well separated, and whose size is compatible with the corresponding receptive field (or the bounding box) in the image.

Category Slots A related approach is to allocate slots according to some categorization of objects based on properties other than location. In this case, because now category and separation are entangled, it is then no longer possible to represent multiple objects of the same category[7]. The main example of category slots is capsules [Hinton et al., 2011, 2018], although other approaches such as Recurrent Entity Networks [Henaff et al., 2017] can also be viewed from this perspective.

Augmentation

Augmentation based approaches, unlike slot based ones, keep a single set of features shared among all object representations and instead augment each feature with additional grouping information. This grouping information is usually continuous, which may help to encode uncertainty about the separation. Object representations based on augmentation will trivially be in a common format, although extracting information about individual objects now requires first processing the grouping information. An important limitation of augmentation is that it requires substantial deviations from standard neural networks and is thus more difficult to integrate with state of the art systems. Due to features being shared, augmentation may also suffer from capacity and ambiguity problems when a feature is active in multiple object representations at the same time (e.g. two red objects), similar to when representing multiple objects of the same category using category slots.

[7]There is evidence that humans similarly struggle with representing multiple *tokens* of the same *type* [Mozer, 1989]. Hence, it could also suggest that this apparent weakness may be advantageous for other aspects of information processing.

Temporal Codes An early approach to object representation using augmentation in neural networks made use of the temporal structure of *spiking neurons* for separation (temporal codes). Here, the activation of a feature encoded by the firing rate is augmented with grouping information encoded by the temporal correlation between firing patterns [Singer, 2009]. In other words, the features that form an object are represented by neurons that fire in synchrony [Milner, 1974; von der Malsburg, 1981; Singer, 1999]. Rather than using unrestricted spiking networks, most work on object representation using temporal codes focuses on *oscillatory networks*, where the firing pattern takes the form of a regular frequency rhythm (for an overview see Wang [2005]). Because temporal codes rely on spiking neurons, they are non-differentiable and also require simulating the dynamics of each neuron even for static inputs. This makes them incompatible with gradient-based training and necessitates a completely different training framework typically based on Hebb's rule [Kempter et al., 1999], or Spike-Timing Dependent Plasticity (STDP [Caporale and Dan, 2008]).

Complex Valued Codes An alternative approach to augmentation uses *complex-valued* neurons (features) in place of oscillatory neurons. Hence, instead of explicitly simulating the temporal behavior of an oscillator, its activation and grouping information can now be described as the absolute value and angle of a complex-valued neuron. Similar to before, the grouping is implicit and continuous, with neurons that "fire at similar angles" being grouped together. Complex-valued neurons are differentiable and more compatible with existing gradient-based learning techniques. On the other hand, they require specialized activation functions that consider both real and imaginary parts, which tend to be difficult to integrate into existing methods. Successful integrations include complex-valued Boltzmann Machines [Zemel et al., 1995; Reichert and Serre, 2013] and complex-valued RNNs that could be trained either with backpropagation [Mozer et al., 1992] or via Hebbian learning [Rao et al., 2008].

Tensor Product Representations

A Tensor Product Representation (TPR) consists of a real-valued matrix (tensor) that is the result of combining distributed representations of *fillers* with distributed representations of *roles*. TPRs can be used for representing multiple objects by associating fillers with object representations and using roles to encode grouping information. A TPR is formed by combining each filler with a corresponding role via an outer product ("binding operation"), which are then composed to accommodate multiple object representations ("conjunction operation"). When

the role representations are linearly independent, then the object representations can be retrieved from the TPR via matrix multiplication ("unbinding operation"). Notice that, when the role-vectors are one-hot encodings, the TPR reduces to instance slots. However, the additional freedom afforded by a general *distributed* role vector can be used to encode structural information or uncertainty about the separation of objects[8]. TPRs always assume that the object representations are described in a common format but note that, similar to augmentation, extracting information about individual objects first requires processing the grouping information (in this case via the unbinding operation). TPRs were first introduced in Smolensky [1990] and several modifications have since been proposed that consider different binding, unbinding, and conjunction operations [Plate, 1995; Kanerva, 1996; Gayler, 1998] (see Kelly et al. [2013] for an overview). In recent literature, TPR-like mechanisms have been incorporated into neural networks using fast-weights [Schlag and Schmidhuber, 2018] or self-attention [Schlag et al., 2019] to perform reasoning in language.

Attractor Dynamics

Up until this point, we have focused on methods that address the representational format of object representations. Now we consider *attractor dynamics* as an approach for addressing their representational dynamics. Robust object representations are well described by a stable attractor state in a larger dynamical system (e.g. implemented by an RNN) that models its representational dynamics based on the input. In this case, inferring a coherent object representation corresponds to running the dynamical system forward until it converges to an attractor state. A stable attractor is naturally self-correcting, and multiple competing interpretations (from ambiguous inputs) can easily be described by separate attractor states. Top-down feedback can then be used to switch interpretations, namely by pushing the state of the system enough to cause it to cross over to a different basin of attraction. By adapting the system dynamics to changing inputs, they allow for moving attractors (changes of the object) or bifurcations (creation or vanishing of particular interpretations).

Attractor Networks incorporate attractor dynamics in neural networks and have a long history in connectionist research. Early work includes Hopfield networks [Hopfield, 1982], Boltzmann machines [Ackley et al., 1985] (see also Glauber [1963]; Sherrington and Kirkpatrick [1975]) and associative memory

[8] Alternatively, it can be used to implement a weaker inductive bias where a factorizable representation is only encouraged, i.e. by constraining the role representations to not be fully linearly independent.

Figure 3.5. Photo of two leaf-tailed geckos "young and old"©2015 by Paul Bertner.

[Kohonen, 1989]. Attractor states were also found to occur naturally in RNNs, especially when using symmetric recurrent weights [Almeida, 1987; Pineda, 1987]. In recent years, however, they have received little attention (but see [Mozer et al., 2018; Iuzzolino et al., 2019]), which might be in part because they can be difficult to train. In particular, the fact that each weight participates in the specification of many attractors can lead to spurious (unintended) attractors and ill-conditioned attraction basins [Neto and Fontanari, 1997]. Localist attractor networks [Zemel and Mozer, 2001] and flexible kernel memory [Nowicki and Siegelmann, 2010] are two approaches that address this issue by introducing a separate representation for each attractor. However, it should be noted that spurious attractors may also be advantageous for generalization, e.g. when they correspond to novel feature combinations.

3.3 Segregation

The problem of segregation is about forming object representations from raw (unstructured) sensory information, such as images. Humans effortlessly perceive objects, yet the process of perceptual organization is surprisingly intricate. Even for everyday objects like a mirror, a chair, or a house, it is difficult to formulate precise boundaries or a definition that generalizes across multiple different contexts. Nonetheless, we argue that an important aspect common to all objects is that they may act as stable and self-contained abstractions of the raw input. This then implies several requirements for the process of segregation.

Consider for example Figure 3.5, which demonstrates several challenges that must be overcome. To recognize the two geckos sitting on a branch you have to segment out two unfamiliar objects (zero-shot) even though they belong to the same class (instance segmentation) and their use of camouflage (texture similarity). Both the large gecko and the branch are visually disconnected due to occlusion, and yet you perceive them as independent wholes (amodal completion). Beyond separating these objects, you have also formed separate representations for them that enable you to efficiently relate, describe, and reason about them.

In this section we take a closer look at this process of segregation[9]. We first work towards a general *notion* of an object built around modularity and hierarchy (Section 3.3.1). Next, we focus on the process of *forming* object representations based on this notion (Section 3.3.2). Unlike segmentation, which is typically only concerned with a static split of information at the *input*-level, segregation is inherently task-dependent and aims to produce stable object *representations* that are grounded in the input and which maintain their identity over time. Towards the end, we review relevant approaches from the literature that may help neural networks perform segregation (Section 3.3.3).

3.3.1 Objects

The question of what constitutes a meaningful object (i.e. for the purpose of building structured models of the world) is central to segregation. However, despite long-standing debates in many fields including philosophy, linguistics, and psychology, there exists no general agreed-upon definition of objects [Cantwell-Smith, 1998; Green, 2018]. Here, we take a pragmatic stance that focuses on the *functional role* of objects as compositional building blocks. Hence, we are not interested in debating the "true" (i.e. metaphysical) nature of objects, but rather consider object representations as components of a useful representational "map" that refers to (but is not identical to) parts of the "territory" (world)[10].

Modularity

From a functional perspective, the defining quality of an object is that it is modular, i.e. it is self-contained and reusable independent of context. While this suggests choosing objects with minimal information content (to improve reusability), it

[9] Calling this process "segregation" as opposed to "grouping" emphasizes the fact that it typically requires a *separation* of the inputs and features into meaningful parts.

[10] *"A map is not the territory it represents, but, if correct, it has a similar structure to the territory, which accounts for its usefulness."* [Korzybski, 1958].

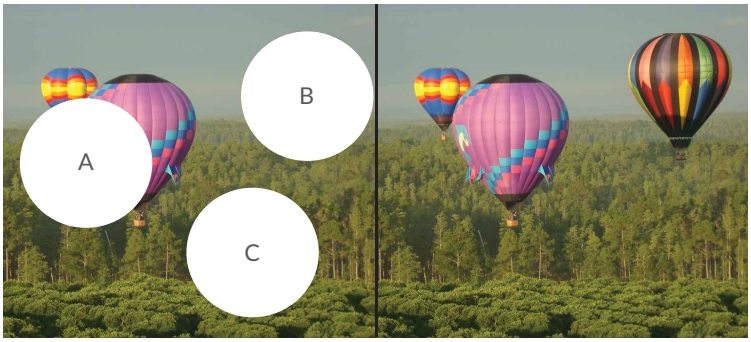


Figure 3.6. For partial objects (A) or only background (C), the occluded regions can be inpainted reasonably well, while in the case of full objects (B) that is impossible.

is equally important that objects can be represented efficiently based on their internal predictive structure. We argue that this trade-off induces a Pareto front of valid decompositions into objects that have both strong internal structure, yet remain largely independent of their surroundings. By organizing information in this way, objects are expected to capture information that is due to independent causes, which matches our own intuitive notion of objects in the real world [Chater, 1996; Green, 2018].

Consider the example of three balloons in front of a forest as depicted in Figure 3.6. When a balloon is partially occluded (as in A), you are still able to make a reasonable guess about the occluded part purely based on its internal predictive structure. On the other hand, when an entire balloon is occluded (as in B) it is impossible to infer its presence from the (unoccluded) context, and the most reasonable reconstruction is to fill in based on the background (as in C). Notice that each balloon is modular in the sense that it is possible to reuse them in many different contexts (e.g. when placed in a different scene). In contrast, this would not be possible if an object were to be formed from the background *and* the balloon. Hence, by carving up perception at the "borders of predictability", objects allow for an approximate divide and conquer (i.e. a compositional) approach to modeling the world.

Hierarchical

Objects are often hierarchical in the sense that they are composed of parts that can themselves be viewed as objects. Consider, for example, a house consisting of a roof and walls, which themselves may consist of several windows and a door, etc. Depending on the desired level of detail, a scene can, therefore, be decomposed in terms of coarser or finer scale objects, corresponding to different solutions on the Pareto front. In most cases, these decompositions relate to each other in the sense that they correspond to different levels in the *same* part-whole hierarchy.

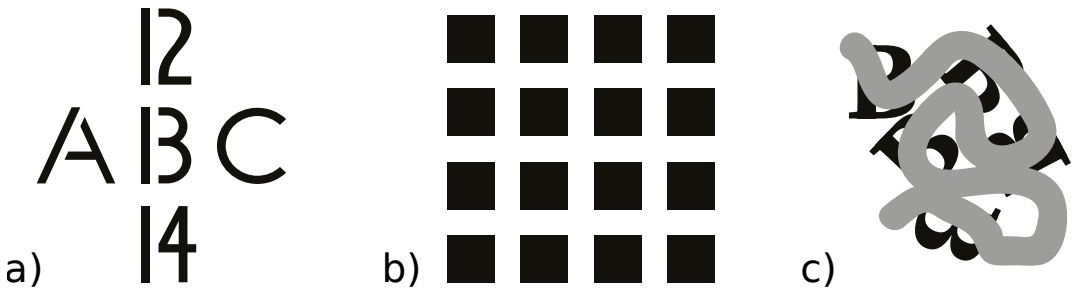


Figure 3.7. Examples of visual illusions that demonstrate multistable perception (a, b) and amodal completion (c).

However, in rare cases, two decompositions may also consider incompatible parts, for example in a page of text that can be decomposed either into lines or sentences[11]. Notice that there is a difference between this part-whole hierarchy and the feature hierarchy commonly found in deep neural networks. Here, parts are themselves objects, which are the result of dynamically separating information into object representations (segregation). Hence, a part-whole hierarchy can also be viewed in terms of a number of general "is-part-of" relations that can be reused between objects (see also Section 3.4).

3.3.2 Segregation Dynamics

Segregation needs not only infer a decomposition into objects, but also corresponding object representations. As is evident from our previous discussion, there is no universal choice of objects that is appropriate in all circumstances, which requires segregation to consider both context- and task-dependent information. Together with the need for a stable outcome, this has several consequences for the segregation dynamics, which we will consider next.

Multistability

Most scenes afford many different useful decompositions that either stem from choosing different levels of granularity (i.e. levels of hierarchy) or from ambiguous inputs that allow for multiple distinct but coherent interpretations (hence the need for multi-modal separation uncertainty in Section 3.2.2). Together, these result in a massive number of potential object representations (e.g. $\geq$ 3000

[11] A unique hierarchy is favored by modularity because in the case of incompatible decompositions (i.e. not corresponding to the same part-whole hierarchy) their objects *cross* the borders of predictability, which implies a weaker internal structure.

letters per page of text). Simultaneously representing all of them is not only intractable but also undesirable, as the majority of object representations will not be useful for any particular situation. A practical solution to this problem is a dynamical segregation process that has *multiple stable* equilibria that each correspond to a particular decomposition of a given scene. Indeed, humans resolve this problem via multistable perception, which allows us to seamlessly switch back and forth between different interpretations [Attneave, 1971]. This effect is often demonstrated with visual illusions as in Figure 3.7a), but is in fact much more common than these constructed examples suggest. For example, a simple tile pattern (as in Figure 3.7b) can easily be perceived in a number of ways, including rows or columns of tiles. Notice that it is possible to simultaneously perceive multiple objects from the same decomposition, but not from different decompositions (e.g. perceiving 13 and B simultaneously in Figure 3.7a). This inherent limitation of multistable segregation can also act as an advantage since it ensures a single coherent decomposition of the input and avoids mixing objects from different incompatible decompositions. Further, this implies that the process of segregation also has to be able to efficiently resolve conflicts from competing decompositions (explaining away).

Incorporating Top-Down Feedback

Certain decompositions lead to a set of building blocks (objects) that are more useful than others for a given task or situation. For example, when moving a stack of chairs to another room it is useful to group information about the individual chairs together as a single object. On the other hand, when the goal is to count each of the individual chairs, a more fine-grained decomposition is preferred (and perhaps when repairing a chair an even more fine-grained decomposition is needed). These building blocks underlie the structure of downstream models that can be used for inference, prediction, and behavior, and the choice of decomposition, therefore, affects the ability to generalize in predictable and systematic ways. Hence, it is important that the outcome of the segregation process can be steered towards the most useful decomposition, based on contextual information. One of the main sources of contextual information is *top-down feedback*, for example in the form of task-specific information (e.g. to guide visual search) or based on a measure of success at performing the given task. Another source of contextual information could be memory, for example by recalling a decomposition that has previously proven useful in the given situation.

Consistency

It is important that the grounding of object representations, as provided by the segregation process, is both stable and consistent across time (i.e. it maintains object identity). This helps to correctly accumulate partial information about objects, to infer temporal attributes from prior observations, and to ensure that the outcome of more abstract computations in terms of object representations remain valid in the environment (Section 3.2.2). It may also help to avoid "double-counting" of evidence (e.g. during learning)[12]. Object identity depends on a reliable mechanism for re-identification i.e. a mechanism for identifying an object as being the same despite changes in appearance, perspective, or temporary absence of sensory information. Consider, for example, a game of cups and balls, which involves tracking a ball hidden under one of three identical cups that are being moved around. In this case, a stable object identity requires maintaining separate identities for the cups despite their identical appearance, as well as re-identifying the ball as it reappears from under the cup. When an object is re-encountered after a prolonged period of time, re-identification may require interfacing with some form of long-term memory.

3.3.3 Methods

To succeed at segregation (in the sense outlined above) a neural network must acquire a comprehensive notion of objects and incorporate mechanisms to dynamically route their information. Due to the prohibitive amount of potentially useful objects, it is unlikely that an adequate notion can be engineered directly or taught purely through large-scale supervision. Therefore, in the following, we will review a wide range of approaches, including more classic non-neural approaches that have produced promising results despite incorporating domain-specific knowledge (albeit to a lesser degree). The latter serve as inspiration for the development of neural approaches that can learn about objects directly from raw data (e.g. by focusing on modularity) and indeed in Chapters 4 and 6 we will present examples of this.

[12] Consider the example from Marcus [2003] about owning a three-legged dog. Even though you will likely see your dog much more often than other dogs, this series of observations does not affect your overall belief about the number of legs that dogs typically have, since these observations are all associated with *the same* dog.

Clustering Approaches to Image Segmentation

Image segmentation is concerned with segmenting the pixels (or edges [Arbeláez et al., 2011]) belonging to an image into groups (e.g. objects) and therefore provides a good starting point for segregation. A common approach to image segmentation is to cluster the pixels of an image based on some similarity function [Jain et al., 1988]. One particularly successful approach is the spectral graph-theoretic framework of normalized cuts [Shi and Malik, 2000], which treats image segmentation as a graph partitioning problem in which nodes are given by pixels and weighted edges reflect the similarity between pairs of (neighboring) pixels. Partitioning is performed by trading-off the total dissimilarity between different groups with the total similarity within the groups. To the extent that the similarity function is able to capture the predictive structure of the data, this is then analogous to the trade-off inherent to modularity discussed before. It is straightforward to achieve a hierarchical segmentation in this graph clustering framework either via repeated top-down partitioning [Shi and Malik, 2000] or bottom-up agglomerative merging [Mobahi et al., 2011; Hoiem et al., 2011].

In the context of segregation, a central challenge is to define a good similarity function between pixels that leads to useful objects. As we have argued, a hardwired similarity function (e.g. as in Shi and Malik [2000]; Malik et al. [2001]) has little chance at facilitating this required flexibility, although different initial seedings of the clustering may still account for multiple different groupings (i.e. multistability). Labeled examples can be used to address this challenge in a multitude of ways, e.g. to learn a similarity function between segments [Ren and Malik, 2003; Endres and Hoiem, 2010; Kong and Fowlkes, 2018] or discrete graphical patterns [Lun et al., 2017], to learn boundary detection [Martin et al., 2004; Hoiem et al., 2011], or as a means of top-down feedback [Mobahi et al., 2011]. Unsupervised approaches (based on self-supervision) provide a more promising alternative. One approach is to learn a similarity function between pairs of pixels, e.g. based on their point-wise mutual information using kernel-density estimation [Isola et al., 2014] or based on a self-supervised prediction task using a neural network [Isola et al., 2015]. Alternatively, one can attempt to steer the clustering process based on the unsupervised principle of compressibility (minimum description length) [Mobahi et al., 2011].

We note that, since clustering-based approaches to image segmentation focus on low-level similarity structures, their understanding of objects at a more high-level is limited (i.e. at the level of object representations).

Neural Approaches to Image Segmentation

An alternative approach to image segmentation that leverages the success of end-to-end learning, is to directly output the segmentation with a deep neural network. Unlike clustering-based approaches, which focus on the similarity structure between pixels (or small segments), learning now takes place at the (global) image level, which allows objects to be modeled at multiple levels of abstraction. On the other hand, due to the one-to-one (feedforward) mapping from image to segmentation, it may now be more difficult to provide multiple different segmentations (multistability), or a hierarchical segmentation, for a given input.

Recent approaches based on supervised learning from ground-truth segmentations have produced high-quality instance segmentations of real-world images[13]. For example, approaches based on R-CNN [Girshick et al., 2014] decompose the instance segmentation problem into the discovery of bounding boxes using region-proposal networks [Ren et al., 2015] and mask prediction [Dai et al., 2016; He et al., 2017] to provide instance segmentations. Other approaches output an energy function from which the segmentation is easily derived, e.g. based on the Watershed transformation [Bai and Urtasun, 2017]. Instance segmentation has also been phrased as an image-to-image translation problem using conditional generative adversarial networks [Mo et al., 2018]. Moreover, approximate instance segments can be obtained as a by-product of performing some other task, such as learning to interpolate between multiple images [Zhang et al., 2020], or minimizing mutual information between image segments [Yang et al., 2020]. Unsupervised approaches that directly infer the segmentation (and that do not require large-scale supervision) are more relevant in the context of segregation but have received far less attention. One approach is to train a neural network to directly output the segment that an input belongs to by maximizing the mutual information between paired inputs in representational space [Ji et al., 2018] (although it operates at the level of patches as opposed to the global image). In the context of video, motion segmentation often produces segments that correspond to instances (provided that they move), which can be learned through unsupervised multi-task learning [Ranjan et al., 2019].

[13] We would like to emphasize the distinction between *instance* segmentation and *semantic* segmentation. In the context of segregation, we are more interested in the former, which is concerned with the more general notion of each segment being an object (instance). In contrast, semantic segmentation associates a particular semantic interpretation (in the form of a label) with each segment and therefore does not segregate multiple objects belonging to the same class.

Attention

In the context of segregation, attention mechanisms provide a means to selectively attend to different objects sequentially. Compared to image segmentation, this does not require exhaustively partitioning the image but instead allows one to focus only on the relevant locations in the image (e.g. as a result of top-down feedback). Here we focus mainly on *hard* attention mechanisms that attend to a strict (i.e. spatially delineated) subset of the available information in the form of an attention window (e.g. in the shape of a bounding-box [Stanley and Miikkulainen, 2004] or a fovea [Schmidhuber and Huber, 1991]). Their strong spatial bias (due to the shape of the attention window) makes them particularly relevant for the domain of images, but also more difficult to adapt to modalities in which meaningful objects are not characterized by spatial closeness. On the other hand, the rigid shape of the attention window may interfere with modularity due to potential difficulties in extracting information about objects with incompatible shapes or that are subject to occlusion.

The main challenge in incorporating attention mechanisms is to correctly place the window. Early approaches by-pass this problem by evaluating a fixed attention window exhaustively at each possible image location, or using several of many heuristics [Lampert et al., 2008; Alexe et al., 2010; Uijlings et al., 2013]. A classifier can then be trained to determine which window contains an object [Rowley et al., 1998; Viola and Jones, 2001; Harzallah et al., 2009]. Other approaches compute a two-dimensional topographical saliency map that reflects the presence of perceptually meaningful structures at a given location. This facilitates an efficient control strategy to direct an attention window in an image by visiting image locations in order of decreasing saliency [Itti et al., 1998]. Salient regions can be learned based on bottom-up information, such as self-information of local image patches [Bruce and Tsotsos, 2006]. Alternatively, they can be derived by also incorporating top-down information, e.g. by highlighting locations that are (maximally) informative for a discriminative task [Gao and Vasconcelos, 2005; Cao et al., 2015; Zhmoginov et al., 2019]. Recently, there has been renewed interest in saliency-based approaches through the discovery of keypoints [Jakab et al., 2018; Kulkarni et al., 2019; Minderer et al., 2019].

It is also possible to directly learn the control strategy for placing the window of attention, which naturally accommodates top-down feedback. For example, learning the control strategy can be viewed as a reinforcement learning problem, in which the actions of an "agent" determine the location of the window. A policy for the agent (frequently implemented by a neural network) can then be evolved [Stanley and Miikkulainen, 2004], trained with Q-learning [Paletta et al.,

2005], or via Policy Gradients [Butko and Movellan, 2009]. Alternatively, it can be incorporated as a separate action in an agent trained to perform some task (e.g. classification) or to interact with an environment [Mnih et al., 2014; Ba et al., 2014]. AIR [Eslami et al., 2016] and its sequential extension SQAIR [Kosiorek et al., 2018] deploy a similar strategy for an unsupervised learning task with the purpose of extracting object representations. They make use of an attention mechanism that is fully differentiable based on spatial transformer networks [Jaderberg et al., 2015], but see also DRAW [Gregor et al., 2015] for an alternative mechanism. Similarly, Tang et al. [2014] incorporates a window of attention in a deep belief network to extract object representations by performing (stochastic) inference over the window parameters alongside the belief states.

Soft attention mechanisms implement attention as a continuous weighing of the input (i.e. a mask) and can be seen as a generalization of hard attention. For example, in MONet [Burgess et al., 2019] and GENESIS [Engelcke et al., 2019] a recurrent neural network is trained to directly support the learning of object representations by outputting a mask that focuses on different objects at each step[14]. A similar soft-attention mechanism has also been used to facilitate supervised learning tasks, such as caption generation [Xu et al., 2015], image classification [Wang et al., 2017], instance segmentation [Ren and Zemel, 2017], or (multi-)object tracking [Kosiorek et al., 2017; Fuchs et al., 2019]. Soft attention mechanisms have also been applied *internally* (self-attention) to support segregation. For example, Mott et al. [2019] incorporates a form of dot-product attention [Vaswani et al., 2017] in an agent to attend to the internal feature maps of a bottom-up convolutional neural network that processes the input image. A similar self-attention mechanism was also used to support image classification [Zoran et al., 2020].

Probabilistic Generative Approaches

A probabilistic approach to segregation is via inference in a generative model that models the observed data in terms of multiple components (objects)[15]. An advantage of explicitly modeling the constituent objects is that it is easy to incorporate assumptions about their structure, including modularity and hierarchy. This then enables inference (segregation) to go beyond low-level similarities or

[14]Notice, however, that these specific methods enforce an *exhaustive* partition of the image similar to image segmentation methods.

[15]Human perception is also said to be generative in the sense that we can perceive objects as coherent wholes even when they are only partially observed (amodal completion), e.g. as in Figure 3.7c.

spatial proximity, and recover object representation based on their high-level structure as implied by the model. On the other hand, as we will see below, inference usually becomes more difficult as the complexity of the generative model increases, and especially when considering multi-modal distributions (e.g. for multistability).

The most basic assumption to incorporate in a generative model, for the purpose of segregation, is to assume that the input is *directly* composed of multiple parts (objects) that are each modeled individually. Inference in such models then allows one to recover a partitioning of the input in addition to a description of each part (object representation). Early approaches model images with a *mixture model* that treats the color values of individual pixels as independent data points that are identically distributed [Samadani, 1995; Friedman and Russell, 1997]. Alternatively, the decomposition can be based on other features such as optical flow [Jepson and Black, 1993] or the coefficients of a wavelet transform [Guerrero-Colón et al., 2008]. Mixture models can also be biased towards spatial coherence to explicitly account for the spatial structure of visual objects [Weiss and Adelson, 1996; Blekas et al., 2005]. Independent Component Analysis (ICA) models the observed data as linear combinations (mixtures) of unobserved random variables (sources) that are statistically independent [Hyvärinen and Oja, 2000]. This approach has been particularly successful at blind source separation (segregation) in the auditory domain (e.g. the cocktail party problem [Cherry, 1953]), although it has also seen application in the context of images [Lee and Lewicki, 2002].

In order to more accurately model complex data distributions, it is possible to incorporate domain-specific knowledge in the generative model (and thereby improve the result of inference). For example, a generative model that captures the geometry of 3D images of indoor scenes, as well as the objects that are in it "[...] integrates a camera model, an enclosing room 'box', frames (windows, doors, pictures), and objects (beds, tables, couches, cabinets), each with their own prior on size, relative dimensions, and locations" [Del Pero et al., 2012]. The results that can be obtained in this way are impressive [Tu and Zhu, 2002; Tu et al., 2005; Zhao and Zhu, 2011; Del Pero et al., 2012, 2013]. However, performing inference in highly complex generative models of this type is problematic and frequently relies on custom inference methods tailored to this particular task (e.g. Markov Chain Monte Carlo using jump moves to remove or add objects or specific initialization strategies [Del Pero et al., 2012]). In recent years, *probabilistic programming languages* have emerged as a general-purpose framework to simplify the design of complex generative models and the corresponding inference process. For example, they have enabled the use of symbolic graphic renderers as forward models [Mansinghka et al., 2013] and incorporated deep neural networks to

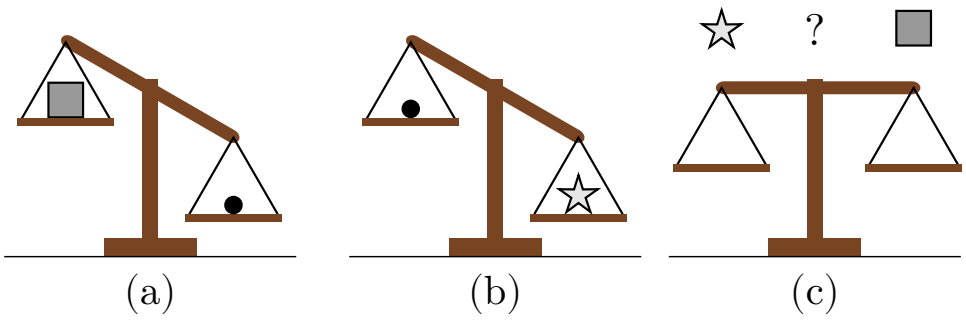


Figure 3.8. Three different objects (■, •, ★) appear in different pairings on a scale (a) and (b). By evaluating their relationships, it can be inferred how the scale will tip in (c).

help make inference more tractable [Kulkarni et al., 2015a]. Nonetheless, in the context of segregation, the amount of domain-specific engineering that is still required limits their generality and applicability to other domains (similar to when overly relying on supervised labels originating from a particular domain).

An alternative approach to more accurately modeling complex data distributions is to incorporate fewer assumptions, but rather parameterize the generative model with a neural network that can *learn* a suitable generative process from many different observations. For example, in Chapter 6 we will demonstrate how a (spatial) mixture model that combines the output of multiple deep neural networks is able to learn to generate images as compositions of individual objects and background [van Steenkiste et al., 2020] (see also Nguyen-Phuoc et al. [2020]). However, in order to perform segregation we must also be able to perform inference in these models, which can be very challenging. This has been addressed by simultaneously learning an amortized iterative inference process based on de-noising [Greff et al., 2016], generalized expectation maximization [Greff* and van Steenkiste* et al., 2017a] (and Chapter 4) or variational inference [Greff et al., 2019]. While these methods still struggle at modeling complex real-world images, they are capable of learning object representations that incorporate many of the previously mentioned desiderata (e.g. common format, modularity), in a completely unsupervised manner.

3.4 Composition

Composition is about leveraging the modularity of object representations to build structured models of the world that are *compositional*. Compositionality is an

important aspect of human cognition and underlies our ability to understand novel situations in terms of existing knowledge. It enables humans to generalize far beyond their direct experiences and is similarly important for systematic generalization in neural networks.

Consider the sequence of observations in Figure 3.8, which allows you to infer the relative weights of the three depicted objects (■, • and ★). Several interesting observations can be made. For example, from panel (a) you can tell that • is heavier than ■, and likewise, that ★ is heavier than • from panel (b). This information does not describe a property of any of the individual objects, but rather more generally applicable *relations* between them. On the other hand, it can still be used to update the properties of the participating objects in response to new information (e.g. the precise weight of ■) or to respond to generic queries, such as answering which of the objects is the heaviest. The latter, in this case, also requires comparing the weights of ■ and ★ (panel (c)). Notice how this is only possible through transitivity of the "heavier than" relation, which allows you to combine the relations from panels (a) and (b) to infer that ★ is heavier than ■. Finally, it is worth pointing out that you are able to apply this comparative reasoning to arbitrary (possibly unfamiliar) objects appearing in novel contexts.

In this section, we take a closer look at how to enable neural networks to dynamically implement *structured models* for a given task, with the ultimate goal of generalizing in a more systematic (human-like) fashion. First, we focus on incorporating compositional structure that combines relations and object representations without undermining their modularity (Section 3.4.1). Next, we consider how a neural network can dynamically infer the appropriate structure and leverage it for reasoning (Section 3.4.2). Towards the end, we survey relevant approaches from the literature that address these aspects of composition (Section 3.4.3).

3.4.1 Structure

To implement structured models, a neural network must organize its computations based on the desired *structure* in terms of objects and their relations. This structure can generally be described by a graph where nodes correspond to objects and edges to relations[16]. By representing relations separately (independent of object

[16]In our discussion, we focus mainly on binary relations (e.g. A is bigger than B) that are well represented by individual edges. However, keep in mind that it is also possible to represent higher-order relations (e.g. A divides B from C), either by using a higher-order graph (e.g. a factor graph) or with the help of auxiliary nodes (e.g. by adding a 'division node' with binary relations to A, B, and C).

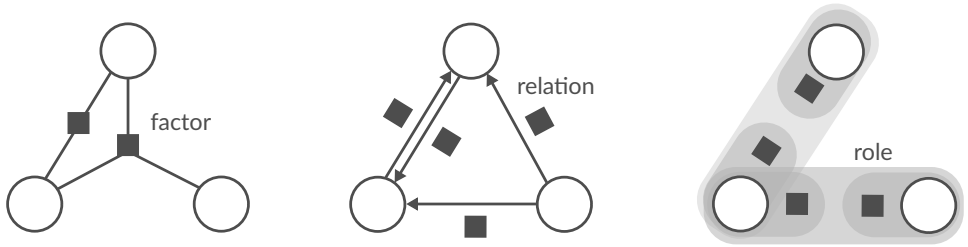


Figure 3.9. Three different ways in which structure can be defined in terms of relations between objects.

representations) it is possible to freely compose relations and objects to form arbitrary structures (i.e. corresponding to different graphs). However, certain types of relations may also impose constraints on the structure to ensure internal consistency between relations (e.g. symmetry or transitivity).

Relations

Relations encode the different computational interactions between the object representations in a structured model. Many different types of relations are possible, including causal relations (e.g. "collides with"), hierarchical relations ("is part of"), or comparative relations (e.g. "bigger than"). Moreover, these general relations can often be specialized to include the nature or strength of an interaction (e.g. "*elastic* collision", "*much* bigger than"). Arguably, to efficiently account for this variability and support learning, relations are best encoded using flexible (neural) representations. Similar to object representations, it may then also be desirable to use a common format that provides a measure of similarity between relations and ensures that they can be used interchangeably[17]. The way structure is defined in terms of relations may also have implications for their corresponding representations. When the structure is given by a regular (directed) graph or a factor graph (see Figure 3.9 a & b), then each relation is encoded by a single representation corresponding to either an edge or a factor. Alternatively, it is possible to encode a relation with multiple representations that each correspond to a possible *role* that an object plays in their interaction as determined by the relation (see Figure 3.9 c). Finally, it is important that relations are represented separately from and independent of the object representations (see also *role-filler-independence* [Hummel et al., 2004]). This enables relations and objects to be composed in arbitrary ways to form a wide variety of (potentially novel) structures.

[17]Doumas et al. [2008] even argues that objects and relations should use a *shared* 'feature pool' with which both can be described.

Variable Binding

To enable a single neural network to implement different structured models, it requires a suitable 'variable binding' mechanism[18] that can *dynamically* combine modular object representations and relations. Consider the classic example of Mary and John adapted from Fodor and Pylyshyn [1988]: Depending on a given task or context it may be more important to consider that "Mary loves John", that "John is taller than Mary", or that "Mary hit John". In general, the number of possible structures that can be considered is potentially very large, and it is, therefore, intractable to represent all of them simultaneously. Apart from being dynamic, a suitable variable binding mechanism should also preserve the modularity of individual objects representations. This is critical to implement structured models that are *compositional*, which enables the neural network to generalize systematically and predictably with respect to the underlying objects.

In many cases, only a single level of variable binding that directly combines individual object representations and relations is needed. However, in certain other cases (e.g. "Bob knows that Mary loves John") it may be required to first build composite structures that can themselves act as 'objects', and that can then be combined recursively. When using a role-based representation for relations, multiple levels of variable binding are also needed to avoid ambiguity when a low-level object representation plays the same role in multiple relations.

Relational Frames

Each *type* of relation focuses on a particular aspect of the broader interaction among objects, and thereby defines a particular *relational frame* that is internally consistent. Consider again the example in Figure 3.8, which was concerned with the "heavier than" relation. This corresponds to a relational frame of comparison that induces an ordering among the objects in terms of their weight. In this case, an internally consistent ordering requires the relation to be transitive (i.e. $A > B \cap B > C \Rightarrow A > C$) and anti-symmetric (i.e. $A > B \Rightarrow B \not> A$). More generally, a relational frame is characterized by a particular type of relation, and by the logical consequences (i.e. different entailments) that are implied by having (multiple) relations of this type within the structure. We adopted the term relational frame from *Relational Frame Theory* (RFT [Barnes-Holmes and Roche, 2001; Hughes and Barnes-Holmes, 2016]), which distinguishes two

[18]The term variable binding is adapted from mathematics, where it refers to binding the free variables in an expression to specific values. In our case, variables correspond to object representations that are bound to the structure determined by the relations.

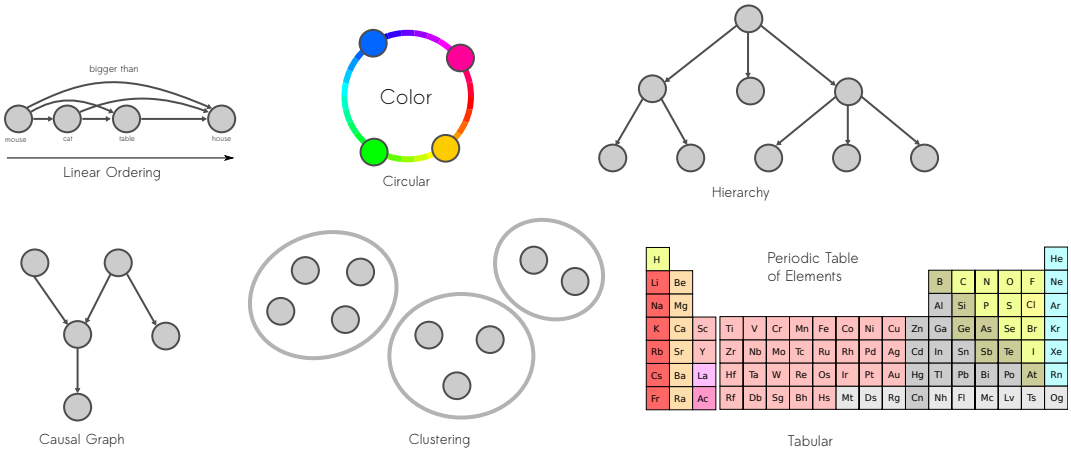


Figure 3.10. Examples of different structural forms [Kemp and Tenenbaum, 2008] that each can be used to define relations among objects but imply different patterns of generalization.

types of entailment that humans primarily use to derive (unobserved) relations: *mutual entailment* and *combinatorial entailment*. Mutual entailment is used to derive additional relations between two objects based on a given relation between them, e.g. anti-symmetry for a frame of comparison, or symmetry for a frame of coordination (i.e. deriving $B = A$ from $A = B$). Analogously, combinatorial entailment is used to derive new relations between two objects based on their relations with a shared third object, e.g. transitivity for a frame of coordination (i.e. deriving $A = C$ from $A = B$ and $B = C$).

Many different types of relational frames can be distinguished, which can be organized into a number of general classes, including 'coordination' (e.g. same as), 'comparison' (e.g. larger than), 'hierarchy' (e.g. part of), 'temporal' (e.g. after), or 'conditional' (e.g. if then) [Hughes and Barnes-Holmes, 2016]. Their corresponding rules for entailment give rise to different *structural forms* [Kemp and Tenenbaum, 2008] among their relations, such as trees, chains, rings, and cliques (see Figure 3.10). In this way, each relational frame can also be seen as encoding a particular (systematic) *pattern of generalization* among the objects. Multiple different relational frames may co-occur within the same structure, which allows for rules of entailment to interact across different frames to facilitate more complex generalization patterns (e.g. $A = B$ and $B > C$ implies $A > C$).

3.4.2 Reasoning

The appropriate structure for a model depends on the task and context, and should, therefore, be dynamically inferred by the neural network to focus only on relevant interactions between the objects. Likewise, it is important to consider the computational interactions between relations and object representations to make use of the inferred structure e.g. for prediction.

Relational Responding

In order to leverage a given structure in terms of relations between object representations, a neural network must be able to organize its computations accordingly. A common use case involves adjusting the (task-specific) response to an object based on its relation to other objects (relational responding). For example, if it is known that ■ is heavier than •, then learning that • is too heavy for a particular purpose (task) also changes your behavior concerning ■. More generally, relational responding of this kind may involve evaluating multiple (derived) relations between objects and combining information across different relational frames. Another use case is in implementing so-called *structure sensitive operations* [Fodor and Pylyshyn, 1988] that require responding *directly* to the structure given by the relations (independent of the object representations). This is especially important for solving abstract reasoning tasks, e.g. when applying the distributive law to a given mathematical expression (i.e. turning $a \cdot (b + c)$ into $a \cdot b + a \cdot c$).

To facilitate relational responding in a neural network, a natural choice is to organize its internal information flow (i.e. computations) in a way that reflects the graph structure of relations and objects. This ensures that newly available information affects the object representations in accordance with the dependency structure implied by the relations (and therefore also with the generalization patterns due to the relational frames). Most information processing of this kind can then be implemented in terms of only local interactions between object representations and relations, which maximally leverages their modularity. These local interactions, which can either be instantaneous (e.g. collides with) or persistent (e.g. is part of), can facilitate both directed (e.g. for causal relations) and bidirectional (e.g. for comparison) information flow. On the other hand, local interactions are ill-suited for implementing structure sensitive operations that require simultaneously considering multiple different parts of the larger structure.

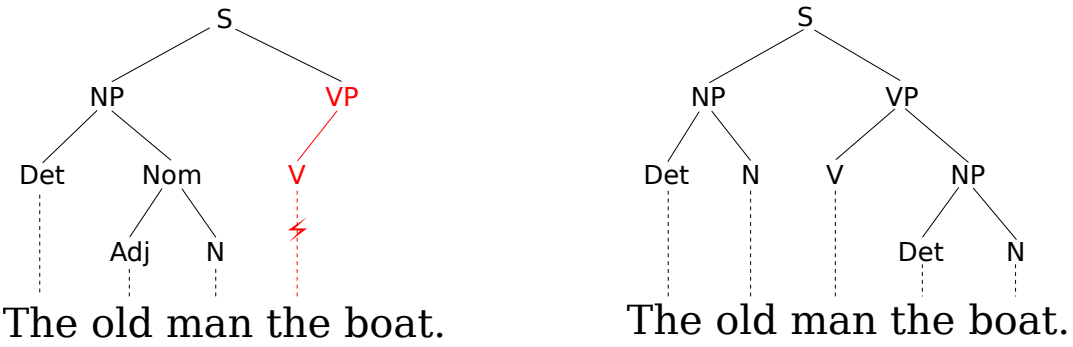


Figure 3.11. Two parse-trees of a garden-path sentence.

Inferring Structure

Inferring the most desirable structure is an inherently difficult task. It requires making many individual choices at the level of individual relations that all have to be coordinated to ensure that the structure as a whole is useful. One important guiding constraint is the internal consistency of the structure with respect to the rules of entailment as implied by the choice of relational frames. Inconsistencies between the observed information and predictions by the structured model are another indicator of a wrong or incomplete structure. The 'garden-path' sentence "The old man the boat." (see Figure 3.11) provides a good example of such a violation of expectations, which then triggers a revision of the structure. Upon first reading, "The old man" is likely parsed as the subject of the sentence, which implies a structure where the next word is expected to be a verb. However, since "the boat" is not a verb (and therefore does not match this expectation), the sentence cannot be parsed in this way. The problem is resolved by revising the structure so that it takes "The old" as the subject and "man" as the *verb* of the sentence. This example also illustrates the need for collaboration between composition and segregation: It was the initial grouping of "The old man" as a single object that gave rise to inconsistencies at the level of structure, which could only be resolved by also changing the outcome of the segregation process. Hence, the process of inferring structure must be able to provide (top-down) feedback to help guide the process of segregation.

Inferring structure at the level of individual relations between objects involves making choices about the type of relation, or about which of the properties of an object to relate. These decisions can be guided by *contextual cues* from the environment, such as the scales in Figure 3.9 that trigger a comparison of the objects in terms of their masses (as opposed to e.g. their relative position or shape). Inferring a relation between objects may also be triggered upon discovering their

relation to other objects (e.g. due to combinatorial entailment). However, for the sake of efficiency it may not always be desirable to explicitly represent such relations, but rather model their effect implicitly due to appropriately organizing the computations of the network (i.e. via relational responding). More generally, the process of inferring structure has to interface closely with the mechanism for variable binding (i.e. for dynamically composing modular object representations and relations).

3.4.3 Methods

To succeed at composition, a neural network requires a mechanism for organizing its internal computations in a way that facilitates relational responding based on the desired structure. A natural approach is to incorporate the structure at an *architectural level* by focusing directly on the local interactions between objects representations and relations. Alternatively, one can also use a more generic (recurrent) neural network "processor" that (sequentially) operates on a *representation* of the desired structure. In the following we will review both of these different approaches, focusing in particular on relational responding and the difficulty of inferring structure[19].

Graph Neural Networks

Graph Neural Networks (GNNs [Scarselli et al., 2009]) are a promising approach for composition that incorporates the desired structure for relational responding at an architectural level (see Wu et al. [2020] for an overview)[20]. At a high level, a GNN is a neural network that is structured according to a graph whose edges determine how information is exchanged among the nodes. In the context of composition, nodes correspond to object representations and edges to relations, which together form the structure, i.e. using (static) variable binding at the architectural level. A GNN fundamentally distinguishes two kinds of information processing, one that requires evaluating the relations between the object representations, and another that is concerned with combining (aggregating) the effect of the

[19]We note, however, that the problem of inferring structure has also received considerable attention in the causality literature, often specifically focusing on cause-effect discovery (e.g. see Hoyer et al. [2009]; Lopez-Paz et al. [2015]; Peters et al. [2016] or Peters et al. [2017] for an overview). More generally, we expect structural causal models to become highly relevant for composition, due to their robustness under intervention and utility for reasoning about hypothetical or unobserved scenarios [Pearl, 2019; Schölkopf, 2019].

[20]See also earlier work on NNs that can manipulate graphs e.g. [Pollack, 1990; Sperduti, 1994; Sperduti et al., 1995]

incoming relations to update the object representations. By implementing these in a general way that applies equally to different objects and relations, a GNN can accommodate many different structures. In general, the local information processing in a GNN ensures that information affects the object representations in accordance with the dependency structure implied by the relations (relational responding).

Graph Convolutional Networks Graph Convolutional Networks (GCNs) are types of GNNs based on a generalization of convolutional neural networks (which operate on grids) to non-euclidean geometries such as graphs [Bronstein et al., 2017]. A GCN consists of several layers that each produce an updated set of node representations by applying graph-convolutions to a local neighborhood in the graph. They have been successfully applied to a wide variety of graph-structured data including social networks [Hamilton et al., 2017], citation networks [Kipf and Welling, 2017], 3d surfaces [Litany et al., 2018], and bio-chemical modeling [Atwood and Towsley, 2016]. However, while they excel at modeling large-scale graphs, one disadvantage of GCNs in the context of composition is that they assume a given graph in the form of an adjacency matrix and node representations as input. For composition, scalability is less important since we are most interested in relatively small graphs (restricted by working memory) that are composed dynamically. On the other hand, some GCNs (e.g. Henaff et al. [2015]; Lee et al. [2019]) have used a mechanism for coarsening (down-sampling) the graph between layers to reduce computational complexity, which could possibly also provide a mechanism for refining the structure (i.e. structure inference).

Message Passing Neural Networks Message Passing Neural Networks (MPNNs [Gilmer et al., 2017]) iteratively update the node representations of a given graph by exchanging messages along its edges (until convergence)[21]. Compared to GCNs, both the graph structure and weights are shared across layers (iterations), and the messages (corresponding to the incoming relations) are typically implemented as a pairwise function of both adjacent node representations. Hence, edges play a more prominent role in information processing and by explicitly considering pairwise interactions it is easier to model comparative relations between objects. MPNNs were initially conceived as a generalization of RNNs to graph-structured inputs [Sperduti and Starita, 1997; Gori et al., 2005] and have since been adapted to consider modern deep neural networks [Li et al., 2016]. A more general

[21] Recently, MPNNs have also been extended to allow for continuous updates [Deng et al., 2019; Liu et al., 2019a].

framework that accommodates both MPNNs and GCNs was proposed in Battaglia et al. [2018].

MPNNs have been shown to generalize more systematically (e.g. compared to standard neural networks) on a number of different tasks that require relational responding in terms of objects, including common-sense physical reasoning [Chang et al., 2016; Battaglia et al., 2016; Mrowca et al., 2018; Janner et al., 2019; Ajay et al., 2019], visual question answering [Santoro et al., 2017; Palm et al., 2018], abstract visual reasoning [Andreas, 2019], natural language processing [Tai et al., 2015], physical construction [Hamrick et al., 2018], and multi-agent interactions [Sun et al., 2019]. In these cases, the desired structure is either specified directly or inferred dynamically based on some heuristic (e.g. based on proximity [Chang et al., 2016; Mrowca et al., 2018] or a language parser [Tai et al., 2015]). Alternatively, MPNNs have been used to implement a relational inductive bias based on a generic structure, e.g. by assuming it to be fixed and fully connected (as in Relation Networks [Santoro et al., 2017]). In this case, information can still be exchanged among all the nodes, although the generalization implied by having the correct structural dependencies is lost (e.g. for entailment).

A more desirable approach is to (dynamically) infer the desired structure, although this is challenging due to the discreteness of graphs and difficulties in comparing them efficiently. One approach is to first learn a continuous embedding for all possible graph structures and then optimize for the right structure in the corresponding space, e.g. using VAEs [Kusner et al., 2017; Zhang et al., 2019], or GANs [Yang et al., 2019]. Similarly, it is possible to search over meta-learned node and edge 'modules' to dynamically assemble the graph [Alet et al., 2019]. Another approach is to directly infer the connectivity between nodes in an iterative fashion based on message passing, e.g. for a fixed number of nodes as in Neural Relational Inference (NRI [Kipf et al., 2018]) or adaptively as in Graph Recurrent Attention Networks (GRANs [Liao et al., 2019]).

Approaches based on Self Attention Graph Neural Networks based on *self-attention* are closely related to MPNNs. The main difference to MPNNs is that they use self-attention to compute a *weighted* sum of the incoming messages (based on the relations) for updating the node representations. This provides a useful mechanism for dynamically adapting the information routing (here a kind of implicit variable binding) and thereby infer the desired structure for a fixed set of nodes. However, note that this may be computationally inefficient since it still requires computing all possible messages and only affects which of them end up being used in the final summation.

The approach by Wang et al. [2018] is based on a kind of (learned) dot-product attention to infer relations between spatial slots. In this case, the attention coefficients are computed for pairs of nodes while the messages are based only on a single node, which may make it more difficult to implement multiple different relations. The use of *multiple attention heads* (i.e. as in Vaswani et al. [2017]), may help mitigate this issue and has been successfully applied for the purposes of relational reasoning about objects [Santoro et al., 2018a; Zambaldi et al., 2019; Goyal et al., 2019; van Steenkiste et al., 2020] (see also Chapter 6) and question answering [Dehghani et al., 2019]. Alternatively, this can be addressed by also conditioning the message on the receiving object representation when using attention e.g. as we will explore in R-NEM [van Steenkiste et al., 2018a] (see also Chapter 5).

The idea of using (self-)attention as a mechanism for inferring structure (and dynamic information routing) has also been applied outside the scope of graph neural networks, e.g. in Pointer Networks [Vinyals et al., 2015], energy-based models [Mordatch, 2018], and Capsules [Sabour et al., 2017; Kosiorek et al., 2019].

Neural Computers

Neural computers offer an alternative approach to composition by learning to perform reasoning operations sequentially on some appropriate representation of the desired structure. In this case, the 'processor' is typically given by an RNN that interfaces with other components, such as a dedicated memory, via a prescribed set of differentiable operations. Compared to a GNN, the architecture of a neural processor is more generic and does not directly reflect the desired dependency structure in terms of relations between object representations. Instead, by considering structure at a representational level, it can more easily be adjusted depending on task or context. Similarly, by having a *central* processor that is responsible for relational responding (as opposed to a distributed GNN) it is easier to support operations that require *global* information (e.g. structure-sensitive operations). On the other hand, the ability of neural computers to learn more general algorithms comes at the cost of a weaker inductive bias for relational reasoning specifically. Hence, it is often necessary to incorporate more specialized mechanisms to efficiently learn algorithms for relational responding that generalize in accordance with the desired structure.

The most common type of neural computer consists of an RNN (the processor) that interfaces with an external differentiable memory component. The latter provides an interface for routing information content (now stored separately) to

the *variables* that take part in processing (i.e. the program executed by the RNN processor). Indeed, while an RNN can in principle perform any kind of computation using only its hidden state as memory [Siegelmann and Sontag, 1991], its dual purpose for representing structure and information processing makes it difficult to learn programs that generalize systematically [Lake and Baroni, 2018]. Early examples of memory-augmented RNNs [Das et al., 1992; Mozer and Das, 1993] use a continuous adaptation of stacks based on the differentiable push and pop operations introduced in Giles et al. [1990] (cf. Joulin and Mikolov [2015] for an alternative implementation). Although a stack-based memory has proven useful for learning about the grammatical structure of language (e.g. Das et al. [1992]), its utility for more general reasoning tasks is limited by the fact that only the top of the stack is accessible at each step.

The addressable memory used in the Neural Turing Machine (NTM [Graves et al., 2014]) offers a more powerful alternative, which can be accessed via generic read and write operations (but see memory networks for a read-only version [Weston et al., 2014; Sukhbaatar et al., 2015]). In this case, all memory slots (and thereby all parts of the structure) are simultaneously accessible through an attention mechanism (responsible for variable binding) that supports both content- and location-based addressing. Together, these operations have shown to provide a useful inductive bias for learning simple algorithms (e.g. copying or sorting) that generalize to longer input sentences (i.e. more systematically). Additional memory addressing operations, e.g. based on the order in which memory locations are accessed (DNC [Graves et al., 2016]), based on when they were last read [Munkhdalai and Yu, 2017], or based on a key-value addressing scheme [Csordas and Schmidhuber, 2019] may confer additional generalization capabilities that are especially relevant for relational reasoning. For example, the DNC has shown capable of learning traversal and shortest path algorithms for general graphs by writing an input sequence of triples ('from node', 'to node', 'edge') to memory, and iteratively traverse this structure using content-based addressing [Graves et al., 2016]. Moreover, given a family tree consisting of ancestral relations between family members, the DNC can successfully derive relationships between distant members, which demonstrates a form of combinatorial entailment.

Other memory-based approaches take a step towards GNNs by updating each memory location in parallel [Kaiser and Sutskever, 2016; Henaff et al., 2017] or incorporate specialized structure for reasoning into the processor, e.g. for visual question answering using a read-only memory (knowledge base) [Hudson and Manning, 2018]. Alternatively, certain (Hebbian) forms of fast weights [Schmidhuber, 1992a] can be viewed as a type of *internal* associative memory based on

previous hidden states [Ba et al., 2016a]. TPR-RNN [Schlag and Schmidhuber, 2018] extends this idea by equipping a fast-weight memory with specialized matrix operations inspired by Tensor Product Representations (TPRs [Smolensky, 1990]), which makes it easier to respond to relational queries. In contrast, Reed and de Freitas [2015] and Kurach et al. [2016] take a step towards modern computer architectures by, respectively, incorporating a call-stack with an explicit compositional structure or a mechanism for manipulating and dereferencing pointers to a differentiable memory tape.

3.5 Disentangling Factors of Variation

In the previous sections, we have focused on incorporating object representations in neural networks, which required addressing the challenges of representation, segregation and composition. In this section, we take another look at representation but focus on a particular representational format in which informative 'factors of variation' are *disentangled*. In Section 3.5.1 we argue for the utility of identifying these factors, while Section 3.5.2 is concerned with (learning) disentangled representations.

3.5.1 Informative Factors of Variation

The idea that complex high-dimensional data, such as the visual appearance of an object, can be efficiently described in terms of a number of informative factors of variation plays a central role in disentanglement. Their existence can be motivated by observing that the semantic features that humans use to describe concepts, such as visual objects, can be organized according to a relatively small number of comparable dimensions that focus on particular properties [Hong et al., 2014; Zheng et al., 2019]. These include more general perceptual dimensions focusing on color and shape, but also functional dimensions that often assume a particular context [Devereux et al., 2014]. By describing the concept that underlies the raw visual representation of an object in terms of these more *general* features (i.e. that equally apply to many other objects) it becomes easier to relate and compare novel instances [Zheng et al., 2019]. Indeed, recall Figure 3.3, which allowed you to experience your own capabilities in describing and comparing unseen objects.

In machine learning, informative factors of variation are typically viewed as a consequence of the generative process that underlies the observed data [Bengio et al., 2013; Higgins et al., 2018a]. They are in part characterized by (1) accounting for a significant fraction of the observed variation among the data, and (2)

corresponding to separate degrees of freedom that can be varied independently. These properties closely relate to the principle of *Independent Causal Mechanisms*, which states that the causal generative process of a system's variables is composed of autonomous modules that do not inform or influence each other [Schölkopf et al., 2012; Peters et al., 2017]. Hence, it may be helpful to think of factors of variation as belonging to the causal mechanisms (or *elementary ingredients* [Suter et al., 2019]) that define the structural causal model according to which the observed data was generated. This relationship then also highlights their significance for machine learning. Firstly, since they can be separately intervened upon, they may act as an *efficient* (i.e. compact) 'basis' for the space of all possible combinations of feature values according to which a significant fraction of the observed variability can be described. For example, by assuming independent mechanisms responsible for generating the color and shape of an object, one can efficiently explain a large chunk of their variability encountered in image space. Secondly, when a continuous stream of observations is due to sparse changes to the underlying causal mechanisms (i.e. as in the real world), they offer a degree of *invariance* that makes it easier to relate high-dimensional observations [Bengio et al., 2013]. Finally, due to their modularity, the same causal mechanism may play a part in different generative processes, which makes it easier to reuse knowledge [Schölkopf et al., 2012].

3.5.2 Learning Disentangled Representations

A *disentangled* representation adopts a particular representational format for encoding information about informative factors of variation. Although the precise properties of the format that make a representation disentangled remains a topic of debate, the core concept of a local correspondence between factors of variation and learned (latent) codes is generally agreed upon [Eastwood and Williams, 2018; Higgins et al., 2018a; Locatello et al., 2018; Ridgeway and Mozer, 2018; Suter et al., 2019].

In prior work, Eastwood and Williams [2018] and Ridgeway and Mozer [2018] put forth three criteria of disentangled representations:

- **Modularity:** The degree to which each neuron in a learned representation encodes information about only a single factor of variation.

- **Compactness:** The degree to which information about each factor of variation is encoded by only a single neuron in a learned representation.

- **Explicitness:** To what extent the mapping between factors and learned codes can be implemented with a simple (i.e. linear) model.

Together, modularity and compactness imply that a disentangled representation implements a one-to-one mapping between informative factors of variation and the learned codes. However, while modularity is commonly agreed upon, compactness is a point of contention. Ridgeway and Mozer [2018] argue that some features (eg. the rotation of an object) are best described with multiple codes, although this is essentially less compact. Other formulations of disentanglement, including the frameworks based on symmetry transformations [Higgins et al., 2018a] or based on causal mechanisms [Suter et al., 2019] decouple the notion of compactness from the number of latent codes.

It is easy to see how a representation that is more disentangled (according to the notion of modularity, compactness, and explicitness) is better able to harness the power of learning about informative factors of variation. Modularity partitions the representation according to factors of variation, which ensures that each part is robust (or invariant) to changes in the input due to other factors (or due to noise). Meanwhile, compactness and explicitness ensure that information about a specific factor can readily be accessed. Hence, learning to solve a downstream task from a disentangled representation has generally been found to be easier [Achille et al., 2018; Lopez et al., 2018] and more efficient [Hsu et al., 2017; Higgins et al., 2017b; Steenbrugge et al., 2018]. We offer our own evidence from a large-scale study regarding their sample-efficiency in Chapter 7 and in van Steenkiste et al. [2019]. Moreover, since different factors of variation focus on independent properties, this factorization captures a product space that has been found to make it easier to generalize to unseen feature combinations [Esmaeili et al., 2019; Greff et al., 2019].

Learning disentangled representations requires incorporating a suitable inductive bias, especially in the *unsupervised* case, where normally the auxiliary task of statistical modeling offers no guarantees about disentanglement of the underlying representation [Locatello et al., 2018; Caselles-Dupré et al., 2019]. Many recent unsupervised approaches to learning disentangled representations take an information bottleneck approach based on (Variational) Auto-Encoders [Tschannen et al., 2018]. In this case, the capacity of the bottleneck (i.e. the output of the encoder) is regularized to contain features that are maximally informative about the observed data and that factorize (e.g. Schmidhuber et al. [1996]; Makhzani et al. [2015]; Higgins et al. [2017a]; Kumar et al. [2018]; Kim and Mnih [2018]). Similarly, purely generative approaches such as InfoGAN [Chen et al., 2016] can learn disentangled features by maximizing mutual information between a number

of factored latent variables and the observed image.

Stronger guarantees can be obtained by incorporating a degree of supervision in the form of data selection. For example, Kulkarni et al. [2015b] propose an approach based on VAEs that assumes access to an oracle that is able to organize the data according to the value of a particular factor, while Karaletsos et al. [2016] requires the oracle to group the data according to a factor-specific similarity score. In this setting, it is also possible to learn disentangled representations via *deep-embedding* methods, which learn to embed high-dimensional observations such that their distance in this space reflects their semantic similarity [Ustinova and Lempitsky, 2016]. In this context, Veit et al. [2017] incorporates a masking mechanisms that makes it possible to learn disentangled embeddings that can be compared across multiple different subspaces (corresponding to a particular factor), but which requires a factor-specific oracle. Alternatively, Ridgeway and Mozer [2018] evaluates their dissimilarity objective only on the dimensions that are most separated, and does not require the oracle to communicate which of the factors is being varied. An advantage of this more weakly-supervised framework is that it is easier to obtain the required amount of supervision. Indeed, Locatello et al. [2020] demonstrates how, by departing from the i.i.d. setting, they can obtain paired observations as consecutive frames from an input stream of images in a self-supervised manner.

Chapter 4

Neural Expectation Maximization

The focus of this chapter is on learning object representations, and in particular on the challenges of *representation* and *segregation* that were discussed in Section 3.2 and Section 3.3 respectively. Learning object representations requires learning about the notion of an object (in this case directly from raw images) and incorporating suitable mechanisms for extracting and representing this information.

In order to learn about objects, we will focus on their functional role as abstract computational units that are modular and reusable across many different contexts. Hence, we adopt the intuitive notion of an object as being a common cause for multiple observations (the pixels that depict the object), which induces a dependency structure among the affected pixels, but leaves other pixels (largely) independent. It implies that we assume that objects are *self-contained*, in that knowledge about some pixels of an object helps in predicting its remainder, whereas it does not improve the predictions for pixels of other objects.

These assumptions allow us to treat segregation as a pixel-level clustering problem, where pixels are related through belonging to the same cluster, and where each cluster corresponds to a particular object. In order to attempt to solve this clustering problem, we require a *trainable* clustering procedure that can adapt to accommodate various definitions of an object, based solely on the observed statistical regularities in the data. When separately representing the information content associated with each cluster in a common format, it can then also address the superposition problem. Taken together, such an approach is expected to be capable of learning a representation of an image that is composed

This chapter is based on Greff* and van Steenkiste* et al. [2017a], which was published as a conference paper at *NIPS 2017*. A preliminary version of this work [Greff* and van Steenkiste* et al., 2017b] appeared as a workshop paper at *ICLR 2017*. These works were done in close collaboration and both authors contributed equally.

of multiple object representations, unlike standard approaches that are only capable of learning a single monolithic representation of an image [Kingma and Welling, 2014; Chen et al., 2016; Higgins et al., 2017a].

The primary contribution in this chapter is to formalize this approach as maximum likelihood estimation (using generalized EM [Dempster et al., 1977]) in a spatial mixture model, where each component is parameterized by a shared neural network. The weights of the neural network act as the similarity function according to which to cluster pixels, and we demonstrate how one can back-propagate gradients through the generalized EM procedure so that they can be trained to learn about objects. Empirically it is shown how the resulting trainable clustering algorithm can be applied to images and videos to learn object representations.

4.1 Method

We will treat segregation as a pixel-level clustering problem and assume that images are composed of K objects, such that each pixel is determined by exactly one object. Which objects are present, as well as the corresponding assignment of pixels to objects, varies from image to image. To acquire object representations, we seek to describe each object $k = 1 \dots K$ with a lower-dimensional distributed representation $\boldsymbol{\theta}_k$ that captures the information content of the affected pixels but carries no information about the remainder of the image.

The key insight underlying our approach is that if we assume access to the family of distributions $\{p(\boldsymbol{x} \mid \boldsymbol{\theta}_k) : \boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k\}$, then we can model each image with a mixture model. In this case, the parameter vectors $\boldsymbol{\theta}_k$ (object representations) and the corresponding assignment of pixels to components (objects), for a given image, can be obtained via maximum likelihood estimation using *Expectation Maximization* (EM). The central problem we therefore address in this section is how to learn this family of distributions in a completely unsupervised fashion.

At a high level, we overcome this problem by formulating a spatial mixture model that parametrizes $\{p(\boldsymbol{x} \mid \boldsymbol{\theta}_k) : \boldsymbol{\theta}_k \in \boldsymbol{\Theta}_k\}$ with a neural network $f_\phi(\boldsymbol{\theta}_k)$ (Section 4.1.1). In this case, the corresponding generalized EM procedure to perform maximum likelihood estimation is fully differentiable (Section 4.1.2), which allows us to obtain a differentiable clustering procedure (Section 4.1.3) that can be trained to learn about objects based on an appropriate 'outer' loss (Section 4.1.4). In the remainder of this section, we formalize and derive this trainable clustering algorithm, which we call *Neural Expectation Maximization* (N-EM).

4.1.1 Neural Spatial Mixture Model

We propose to model each image $\boldsymbol{x} \in \mathbb{R}^D$ as a spatial mixture of K components parameterized by vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K \in \mathbb{R}^M$. A differentiable non-linear function f_ϕ (a neural network) is used to transform these representations $\boldsymbol{\theta}_k$ into parameters $\psi_{i,k} := f_\phi(\boldsymbol{\theta}_k)_i$ for separate pixel-wise distributions. These are typically assumed to be either Bernoulli or Gaussian, in which case $\psi_{i,k}$ corresponds to the mean parameter or the mean and variance parameters respectively. This parametrization assumes that given the representation, the pixels are independent but *not* identically distributed (unlike in standard mixture models). A set of binary latent variables $\boldsymbol{Z}_i \in [0,1]^K$ encodes the unknown true pixel assignments, such that $\boldsymbol{Z}_{i,k} = 1$ *iff* pixel i was generated by component k, and $\sum_k \boldsymbol{z}_{i,k} = 1$. A graphical representation of this model can be seen in Figure 4.1, where $\boldsymbol{\pi} = (\pi_1, \ldots \pi_K)$ are the mixing coefficients that determine the categorical prior $p(\boldsymbol{z}_i)$. The full likelihood for $\boldsymbol{x}$ given $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ is given by:

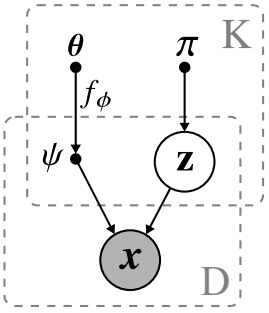


Figure 4.1. The probabilistic graphical model that underlies N-EM.

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \prod_{i=1}^{D} \sum_{\boldsymbol{z}_i \in \mathcal{Z}_i} p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\psi}_i) = \prod_{i=1}^{D} \sum_{k=1}^{K} \underbrace{p(\boldsymbol{Z}_{i,k} = 1)}_{\pi_k} p(\boldsymbol{x}_i \mid \boldsymbol{Z}_{i,k} = 1, \psi_{i,k}). \quad (4.1)$$

4.1.2 Expectation Maximization

Directly optimizing $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is difficult due to marginalization over $\mathbf{z}$, while for many standard choices of distributions optimizing $\log p(\boldsymbol{x}, \mathbf{z} \mid \boldsymbol{\theta})$ is much easier. Expectation Maximization (EM [Dempster et al., 1977])[1] takes advantage of this and instead optimizes a lower bound given by the expected log-likelihood (see Section 2.2.1 for a derivation):

$$\log p(\boldsymbol{x} \mid \boldsymbol{\theta}) \geq \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) := \sum_{i=1}^{D} \sum_{\boldsymbol{z}_i \in \mathcal{Z}_i} p(\mathbf{z}_i \mid \boldsymbol{x}_i, \boldsymbol{\psi}_i^{\text{old}}) \log p(\boldsymbol{x}_i, \mathbf{z}_i \mid \boldsymbol{\psi}_i). \quad (4.2)$$

[1] See von der Malsburg [1973] for an early application of EM to NNs.

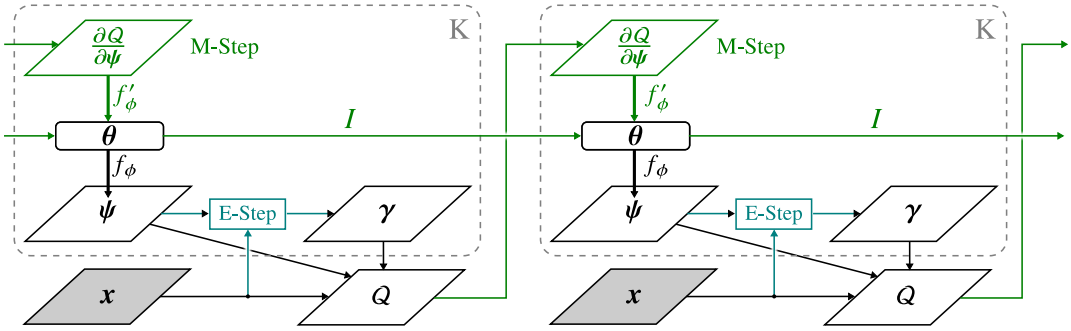


Figure 4.2. Illustration of the unrolled computational graph of generalized EM for the neural spatial mixture model under consideration. By backpropagating gradients through this graph (using backpropagation through time) it becomes possible to train the neural network parameters ϕ based on an appropriate "outer" loss. Two steps of this trainable clustering procedure, which we call Neural Expectation Maximization (N-EM), are shown.

Iterative optimization of this bound alternates between two steps. In the *E-step* we compute a new estimate of the posterior probability distribution over the latent variables given $\boldsymbol{\theta}^{\text{old}}$ from the previous iteration. In the case of a (spatial) mixture model, this re-estimation essentially corresponds to an update of the *soft assignment* of the pixels to the components (clusters) based on the old model parameters:

$$\gamma_{i,k} := p(\boldsymbol{Z}_{i,k} = 1 \mid \boldsymbol{x}_i, \psi_{i,k}^{\text{old}}) = \frac{\pi_k \cdot p(\boldsymbol{x}_i \mid \boldsymbol{Z}_{i,k} = 1, \psi_{i,k}^{\text{old}})}{\sum_{k'} \pi_{k'} \cdot p(\boldsymbol{x}_i \mid \boldsymbol{Z}_{i,k'} = 1, \psi_{i,k'}^{\text{old}})}. \tag{4.3}$$

In the *M-step* we then perform maximum likelihood estimation by choosing $\boldsymbol{\theta}$ as to maximally increase the expected log-likelihood using the posteriors computed in the E-step. Due to the non-linearity of f_ϕ it is difficult to formulate an analytical solution to $\arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$. However, since f_ϕ is differentiable, we can improve $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ and increase the likelihood by taking a gradient ascent step:[2]

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} + \eta \frac{\partial \mathcal{Q}}{\partial \boldsymbol{\theta}} \qquad \text{where} \qquad \frac{\partial \mathcal{Q}}{\partial \boldsymbol{\theta}_k} \propto \sum_{i=1}^{D} \gamma_{i,k} (\psi_{i,k} - \boldsymbol{x}_i) \frac{\partial \psi_{i,k}}{\partial \boldsymbol{\theta}_k}. \tag{4.4}$$

The resulting algorithm belongs to the class of *generalized EM algorithms* and is guaranteed (for a sufficiently small learning rate η) to converge to a (local) optimum of the data log-likelihood [Wu, 1983].

[2] Here we assume that $p(\boldsymbol{x}_i \mid \boldsymbol{Z}_{i,k} = 1, \psi_{i,k})$ is given by $\mathcal{N}(\boldsymbol{x}_i \mid \mu = \psi_{i,k}, \sigma^2)$ for some fixed σ^2 as in (2.7), yet a similar update arises for many typical parametrizations of pixel distributions.

4.1.3 Trainable Clustering Procedure

The similarity function that determines what pixels end up clustered together (after applying generalized EM) is largely determined by the neural network f_ϕ, and in particular by its weights ϕ. So far, we have considered f_ϕ to be fixed and have shown how we can compute a maximum likelihood estimate for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ (which acts as the representation) alongside the appropriate clustering. We now observe that by *unrolling* the iterations of the presented generalized EM algorithm, we obtain an end-to-end differentiable clustering procedure based on the spatial mixture model parameterized by f_ϕ (see Figure 4.2). This enables us to train f_ϕ to capture the statistical regularities corresponding to objects, namely by considering the outcome of clustering on an entire *data set* of images. In particular, using backpropagation through time (BPTT [Werbos, 1988; Williams, 1989]), we are now able to compute gradients of an appropriate 'outer' loss (see Section 4.1.4) with respect to ϕ and train f_ϕ via (stochastic) gradient descent. We refer to this trainable procedure as *Neural Expectation Maximization* (N-EM).

RNN-EM Upon inspection of the structure of N-EM (Figure 4.2) we find that it resembles K copies of a recurrent neural network with hidden states $\boldsymbol{\theta}_k$ that at each timestep receive $\boldsymbol{\gamma}_{:,k} \odot (\boldsymbol{\psi}_{:,k} - \boldsymbol{x})$ as their input (due to application of the chain-rule in (4.4)). Each copy computes a new $\boldsymbol{\psi}_{:,k}$, which is then used by the E-step to re-estimate the soft-assignments $\boldsymbol{\gamma}$. In order to accurately mimic the M-Step (4.4) with an RNN, we must impose several restrictions on its weights and structure: the 'encoder' (that processes the input before it is combined with the hidden state) must correspond to the Jacobian $\partial \boldsymbol{\psi}_{:,k} / \partial \boldsymbol{\theta}_k$, and the recurrent update must linearly combine the output of the encoder with $\boldsymbol{\theta}_k$ from the previous step. Instead, we introduce a variation of N-EM called RNN-EM, which substitutes that part of the computational graph with an actual RNN (without imposing any restrictions and using a standard neural network encoder). Although RNN-EM can no longer guarantee convergence of the data log-likelihood, its recurrent weights, and the use of an encoder increase the overall flexibility of the clustering procedure. In particular, by essentially *amortizing* the approximate M-step[3] across the entire data set it is foreseeable that RNN-EM is better able to leverage the top-down

[3] One way of understanding this modification is through the lens of amortized variational inference [Kingma and Welling, 2014]. If one were to treat $\boldsymbol{\theta}$ as a random variable, then *learning* would additionally involve computing the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{x})$, which can be approximated using an inference model such as a neural network encoder, similar to the one considered here. In fact, such a formalization but based on iterative amortized inference [Marino et al., 2018] was later studied in Greff et al. [2019].

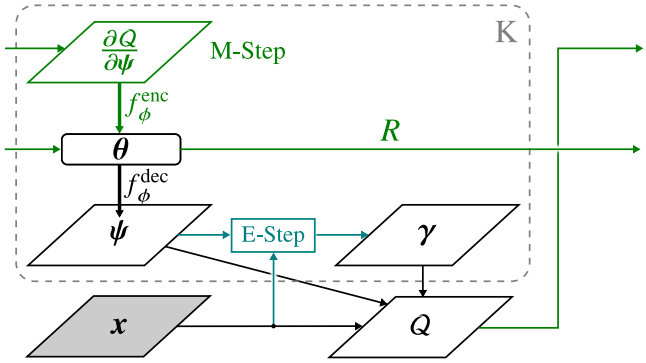


Figure 4.3. Illustration of a single step of RNN-EM. Compared to N-EM (Figure 4.2), the gradient in the M-step is approximated by the output of an encoder that receives the top-down error $\boldsymbol{\gamma}_{:,k} \odot (\boldsymbol{\psi}_{:,k} - \boldsymbol{x})$ as input. The parameters $\boldsymbol{\theta}_k$ now act as the hidden state of a recurrent neural network, which is updated at each (time-)step based on the output of the encoder and the previous estimate of the parameters. By training the encoder across a data set of images we essentially amortize the M-step.

error $\boldsymbol{\gamma}_{:,k} \odot (\boldsymbol{\psi}_{:,k} - \boldsymbol{x})$ to improve $\boldsymbol{\theta}_k$. Moreover, by using a fully parameterized recurrent weight matrix, RNN-EM naturally extends to sequential data, as we will see next. Figure 4.3 presents the computational graph of a single RNN-EM timestep.

4.1.4 Training Objective

N-EM is a trainable clustering procedure that clusters the pixels in an image and computes a distributed representation (that has a common format) for each cluster. Its outcome relies on the neural network f_ϕ, which can be trained across an entire data set of images to learn about statistical regularities corresponding to *objects*. This necessitates a suitable training objective that teaches f_ϕ to map representations $\boldsymbol{\theta}$ to parameters $\boldsymbol{\psi}$ that correspond to pixel-level distributions for such objects.

We accomplish this with a two-term loss function that guides the neural network to focus on regularities in the images that are modular and self-contained. More specifically, each of the K neural network copies is encouraged to model the pixels belonging to a cluster independently of any other information in the image by minimizing:

$$L(\boldsymbol{x} \mid \boldsymbol{\phi}) = -\sum_{i=1}^{D}\sum_{k=1}^{K} \underbrace{\gamma_{i,k} \log p(\boldsymbol{x}_i, \boldsymbol{Z}_{i,k} = 1 \mid \psi_{i,k})}_{\text{intra-cluster loss}} + \sum_{i=1}^{D}\sum_{k=1}^{K} \underbrace{(1-\gamma_{i,k}) D_{KL}[p(\boldsymbol{x}_i) \,||\, p(\boldsymbol{x}_i \mid \boldsymbol{Z}_{i,k} = 1, \psi_{i,k})]}_{\text{inter-cluster loss}}. \tag{4.5}$$

The *intra-cluster* loss corresponds to the same expected data log-likelihood $\mathcal{Q}$ that is optimized during the M-step in EM. It encourages each neural network copy to increase the likelihood for pixels that have been assigned to it, and in the case of RNN-EM, it will encourage the encoder to perform a similar role as the M-step. From the perspective of RNN-EM, the intra-cluster loss is analogous to a standard reconstruction loss used for training autoencoders but weighted by the soft assignment of pixels to clusters. Similar to autoencoders, this objective is prone to trivial solutions in case of overcapacity, which would prevent the network from modeling the statistical regularities that we are interested in. Standard techniques can be used to overcome this problem, such as making $\boldsymbol{\theta}$ a bottleneck (i.e. to put emphasis on information that can be compactly encoded) or by feeding a noisy version of $\boldsymbol{x}$ to the encoder (i.e. to put emphasis on information that has internal predictive structure). It can also be seen how this loss can be adapted to become a *next-step prediction* loss to extend RNN-EM to sequential data $\boldsymbol{x}[1], \ldots, \boldsymbol{x}[T]$, namely by evaluating (4.5) using $\boldsymbol{x}[t+1]$ and treating each EM step as a single time-step[4].

Weighing the loss pixel-wise is crucial since it allows each network to specialize its predictions to an individual object[5]. However, it also introduces a problem: the loss for out-of-cluster pixels ($\gamma_{i,k} = 0$) vanishes. This leaves the network free to predict anything for pixels that have not been assigned to it, which corrupts the underlying representation. Therefore, we add a second term (the *inter-cluster loss*), for regularization, which penalizes the KL divergence between out-of-cluster predictions and the pixel-wise prior of the data $p(\boldsymbol{x}_i)$ (hyperparameter).

[4]This way of applying RNN-EM can essentially be thought of as solving a moving target problem, where the target $\boldsymbol{x}$ updates after applying an approximate EM step. Alternatively, it is possible to consider *multiple* approximate EM steps per time-step. However, this would necessitate a separate update rule (i.e. a separate weight matrix) to distinguish between "refining" the representation for a given time-step, and "updating" it to capture information about the next time-step.

[5]A potential disadvantage of the interaction between $\boldsymbol{\gamma}$ and $\boldsymbol{\psi}$ in (4.5) is that it could yield conflicting gradients. For any $\boldsymbol{\theta}_k$ the loss for a given pixel i can be reduced by better predicting $\psi_{i,k}$, or by decreasing $\boldsymbol{\gamma}_{i,k}$ (i.e. taking less responsibility) which is, due to the form of the E-step, essentially equivalent to doing a worse job at predicting $\psi_{i,k}$. A practical solution to this problem is obtained by stopping the $\boldsymbol{\gamma}$ gradients, i.e. by setting $\partial L / \partial \boldsymbol{\gamma}$ to 0 during backpropagation.

Intuitively this prevents each representation $\boldsymbol{\theta}_k$ from containing information about non-assigned pixels, i.e. in this case $p(\boldsymbol{x}_i \mid \boldsymbol{Z}_{i,k} = 1, \psi_{i,k}) = p(\boldsymbol{x}_i)$.

4.2 Related work

A closely related method to our approach is Tagger [Greff et al., 2016], which similarly learns perceptual grouping in an unsupervised fashion using K copies of a neural network that work together by reconstructing different parts of the input. Unlike in the case of N-EM, these copies additionally learn to output the grouping, which gives Tagger more direct control over the segmentation and supports its use on complex texture segmentation tasks. Our work maintains a close connection to EM and relies on exact posterior inference using the E-step as a grouping mechanism. This facilitates theoretical analysis and simplifies the task for the resulting networks, which we find can be markedly smaller than in Tagger. While N-EM produces modular *object representations*, Tagger critically relies on Ladder Networks [Rasmus et al., 2015], which spread the representational content of objects across many different layers. Finally, Tagger does not include any recurrent connections at the level of hidden states, precluding it from next-step prediction on sequential tasks. RTagger [Prémont-Schwarz et al., 2017]: a recurrent extension of Tagger that does support sequential data, was developed concurrently to this work.

N-EM is also related to Attend Infer Repeat (AIR) [Eslami et al., 2016], which models an image in a sequential fashion by attending to different regions (corresponding to objects) at each step. Unlike AIR, which uses sequential slots, N-EM makes use of instance slots that are described in a common format and further decouples the number of iterations (EM steps) from the number of objects. Moreover, the use of attention in AIR introduces a stronger spatial bias, which potentially makes it more difficult to model objects consisting of parts that are not spatially near. However, unlike N-EM, AIR learns a *generative* model of the data distribution and is in principle also able to synthesize new images. While AIR could only be applied to static images, the later work by Kosiorek et al. [2018] was able to extend AIR to sequential data.

A broader connection of N-EM is to unsupervised segmentation, which has been studied in several different contexts [Schmidhuber, 1992c]: from random vectors [Hyvärinen and Perkiö, 2006] and textures [Guerrero-Colón et al., 2008] to images [Kannan et al., 2007; Isola et al., 2015]. Early work in unsupervised video segmentation [Jojic and Frey, 2001] used generalized EM to infer how to split frames of moving sprites. More recently, optical flow has been used to train convolutional networks to do figure/ground segmentation [Pathak et al.,

2017; Vijayanarasimhan et al., 2017]. A related line of work under the term of multi-causal modeling [Saund, 1995] has formalized perceptual grouping as inference in a generative compositional model of images. For example, Masked RBMs [Le Roux et al., 2011] extend Restricted Boltzmann Machines with a latent mask inferred through Block-Gibbs sampling.

Gradient backpropagation through inference updates has previously been used in the context of sparse coding with (Fast) Iterative Shrinkage/Thresholding Algorithms ((F)ISTA [Daubechies et al., 2004; Rozell et al., 2008; Beck and Teboulle, 2009]). Here, the unrolled graph of a fixed number of ISTA iterations is replaced by a recurrent neural network that parametrizes the gradient computations and is trained to predict the sparse codes directly [Gregor and LeCun, 2010]. We derive RNN-EM from N-EM in a similar fashion and likewise obtain a trainable procedure that has the structure of iterative pursuit built into the architecture while leaving tunable degrees of freedom that can improve their modeling capabilities [Sprechmann et al., 2015]. From this perspective RNN-EM can also be viewed as performing a kind of *meta-learning* (i.e. learning to learn) using RNNs [Hochreiter et al., 2001].

4.3 Experiments

We evaluate our approach on a perceptual grouping task for generated static images and video. By composing images out of simple shapes we have control over the statistical structure of the data, as well as access to the ground-truth clustering. This allows us to verify that the proposed method indeed recovers the intended grouping and learns representations corresponding to these objects. Additionally, we are interested in studying the role of next-step prediction as an unsupervised objective for perceptual grouping, the effect of the hyperparameter K, and the usefulness of the learned object representations.

In all experiments we use ADAM [Kingma and Ba, 2015] with default parameters and a batch size of 64 to train the neural networks. We use 50K observations for training, 10K for validation, and 10K for testing. For simplicity we choose to ignore the mixing prior $p(\boldsymbol{z} \mid \boldsymbol{\pi})$ in any of the proceeding computations by assuming fixed equal mixing probabilities for all components. We use a pixel prior with zero mean, which is either Bernoulli or Gaussian depending on the choice of likelihood. Consistent with earlier work [Greff et al., 2015, 2016], we evaluate the quality of the learned groupings with respect to the ground truth while ignoring the background and overlap regions. This comparison is done using the Adjusted Mutual Information (AMI [Vinh et al., 2010]) score, which provides a measure of

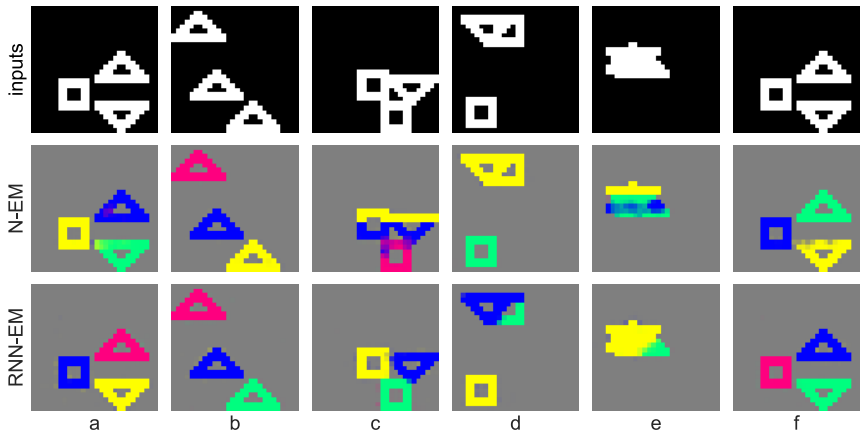


Figure 4.4. Groupings learned by RNN-EM (bottom row) and N-EM (middle row) for six input images (top row). Both methods recover the individual shapes accurately when they are separated (a, b, f), even when confronted with the same shape (b). RNN-EM is able to handle most occlusion (c, d) but sometimes fails (e). The exact assignments are permutation invariant and depend on the initialization as can be seen by comparing (a) and (f).

clustering similarity between 0 (random) and 1 (perfect match). We use early stopping when the validation loss has not improved for 10 epochs[6]. A detailed overview of the experimental setup can be found in Appendix A.1. All reported results are averages computed over five seeds[7].

4.3.1 Static Shapes

To validate that our approach yields the intended behavior, we consider a simple perceptual grouping task that involves grouping three randomly chosen regular shapes (△▽□) located in random positions of 28×28 binary images [Reichert and Serre, 2013]. This simple setup serves as a test-bed for comparing N-EM and RNN-EM, before moving on to more complex scenarios.

We implement f_ϕ using a single layer fully-connected neural network with a sigmoid output $\psi_{i,k}$ for each pixel, which corresponds to the mean of a Bernoulli distribution. The representation $\boldsymbol{\theta}_k$ is a real-valued 250-dimensional vector squashed to the $(0, 1)$ range by a Sigmoid function before being fed into the network. Simi-

[6]Note that we do not use AMI for early stopping and neither is it part of the training objective. All learning is done *unsupervised* and AMI is only measured for evaluation purposes.

[7]Code to reproduce all experiments is available at `https://github.com/sjoerdvansteenkiste/Neural-EM`.

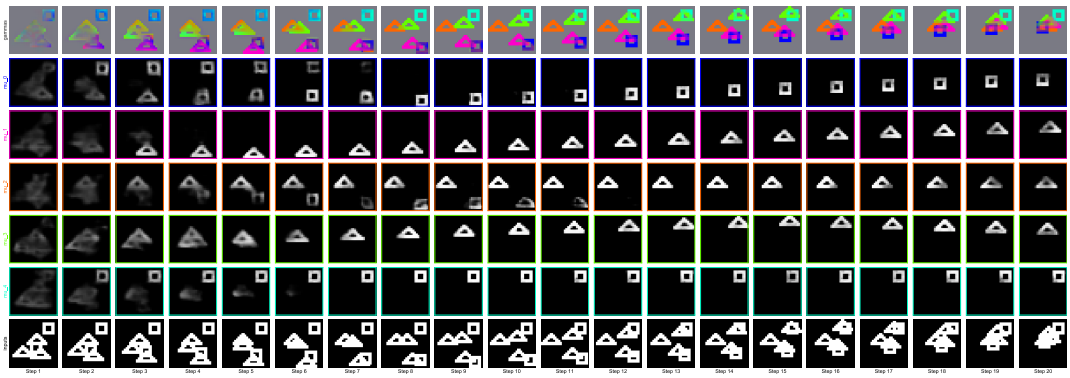


Figure 4.5. A sequence of 5 shapes flying along random trajectories (bottom row). The next-step prediction (i.e. the output) of each neural network copy (rows 2 to 5) and the soft-assignment of the pixels to each of the copies (top row). Notice how RNN-EM has learned to separate each of the individual shapes as a means to efficiently model the observed sequence when using the next-step prediction objective. Even when many of the shapes are overlapping, as can be seen in time-steps 18-20, the network is still able to separate the individual shapes from the clutter.

larly, for RNN-EM we use a recurrent neural network with 250 sigmoidal hidden units and an equivalent output layer. Both networks are trained with $K = 3$ and unrolled for 15 EM steps.

As shown in Figure 4.4, we observe that both approaches can recover the individual shapes as long as they are separated, even when confronted with identical shapes. N-EM performs worse if the image contains occlusion, and we find that RNN-EM is in general more stable and produces considerably better groupings. This observation is in line with findings for Sparse Coding [Gregor and LeCun, 2010] and we argue that the tunable degrees of freedom in RNN-EM similarly help speed-up the optimization process resulting in a more powerful approach that requires fewer iterations. This benefit is reflected in the large score difference between the two: 0.826 ± 0.005 AMI for RNN-EM compared to 0.475 ± 0.043 AMI for N-EM. In comparison, Tagger achieves an AMI score of 0.79 ± 0.034 (and 0.97 ± 0.009 with layer normalization [Ba et al., 2016b]), while using about twenty times more parameters [Greff et al., 2016].

4.3.2 Flying Shapes

We consider a sequential extension of the *static shapes* dataset in which the shapes (△▽□) are floating along random trajectories and bounce off walls. An

Training configuration			Different Test Configuration		
# obj.	K	AMI	# obj.	K	AMI
3	3	0.970 ± 0.005	3	5	0.972 ± 0.007
3	5	0.997 ± 0.002	3	3	0.914 ± 0.015
5	3	0.614 ± 0.003	3	3	0.886 ± 0.010
5	5	0.878 ± 0.003	3	5	0.981 ± 0.003

Table 4.1. AMI scores obtained by RNN-EM on the test-set of *flying shapes*. On the side, the same number of objects and components K is used during training and testing. On the right side, the number of objects or the number of components K is changed at test time.

example sequence with 5 shapes can be seen in the bottom row of Figure 4.5. We use a convolutional encoder and decoder inspired by the discriminator and generator networks of infoGAN [Chen et al., 2016], with a recurrent neural network of 100 sigmoidal units (for details see Appendix A.1.2)[8]. At each time-step t (corresponding to a single approximate EM step) the network receives $\boldsymbol{\gamma}_{:,k}[t-1] \odot (\boldsymbol{\psi}_{:,k}[t-1] - \tilde{\boldsymbol{x}}[t])$ as input, where $\tilde{\boldsymbol{x}}[t]$ is the current frame corrupted with additional bitflip noise ($p = 0.2$). The next-step prediction objective is implemented by replacing $\boldsymbol{x}$ with $\boldsymbol{x}[t+1]$ in (4.5), and is evaluated for each time-step.

Table 4.1 summarizes the results on flying shapes. For 3 shapes we observe that the produced groupings are close to perfect (AMI: 0.970 ± 0.005) and even in very cluttered scenes when using 5 shapes, RNN-EM is able to separate the individual objects and learn object representations in almost all cases (AMI: 0.878 ± 0.003). A representative example of applying a trained RNN-EM with $K = 5$ to a sequence of input observations can be seen in Figure 4.5.

These results demonstrate the adequacy of the next-step prediction objective for perceptual grouping. However, we find that the converse also holds: the factorization into separate object representations is useful for the prediction task. In Figure 4.6 we compare the next-step prediction error, i.e. the Binary Cross-Entropy (BCE), of RNN-EM with $K = 1$ (which reduces to a recurrent autoencoder that receives the difference between its previous prediction and the current frame as input) to RNN-EM with $K = 5$ on this task. To evaluate RNN-EM on next-step prediction we computed the BCE using the heuristic $p(\boldsymbol{x}_i[t+1] \mid \boldsymbol{\theta}[t]) \approx p(\boldsymbol{x}_i[t+$

[8] Using a *convolutional* encoder and decoder adds a spatial inductive bias in this case. It reflects that pixels belonging to the same object are often spatially connected.

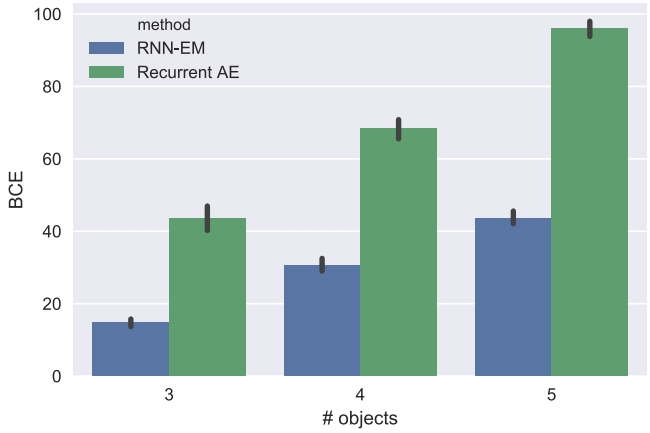


Figure 4.6. Binary Cross Entropy (BCE) error obtained by RNN-EM and a recurrent autoencoder (RNN-EM with $K = 1$) on the denoising and next-step prediction task (flying Shapes). RNN-EM produces significantly lower BCE across multiple different datasets that vary the numbers of objects.

$1] \mid \max_k \psi_{i,k}[t])$ as opposed to $p(\boldsymbol{x}_i[t+1] \mid \boldsymbol{\theta}[t]) = \sum_k \gamma_{i,k}[t] \cdot p(\boldsymbol{x}_i[t+1] \mid \boldsymbol{Z}_{i,k} = 1, \psi_{i,k}[t])$. This ensures that we do not include information from the next time-step through the E-step, while we expect this heuristic to perform only slightly worse as a replacement for evaluation. Regardless, from the figure, we observe that RNN-EM produces significantly lower errors, especially when the number of objects increases.

Finally, in Table 4.1 we also provide insight into the impact of choosing the hyperparameter K, which is unknown for many real-world scenarios. Surprisingly we observe that training with too large K is in fact favorable and that the network learns to leave the excess groups empty. When training with too few components we find that the network still learns about the individual shapes and we observe only a slight drop in score when correctly setting the number of components at test time. We conclude that RNN-EM is robust towards different choices of K, and specifically that choosing K to be too high is not detrimental.

4.3.3 Flying MNIST

To incorporate greater variability among the objects we consider a sequential extension of MNIST [LeCun et al., 1998]. Here, each sequence consists of gray-scale 24×24 images containing two down-sampled MNIST digits that start in random positions and float along randomly sampled trajectories within the image for T time-steps. An example sequence can be seen in the bottom row of Figure 4.7.

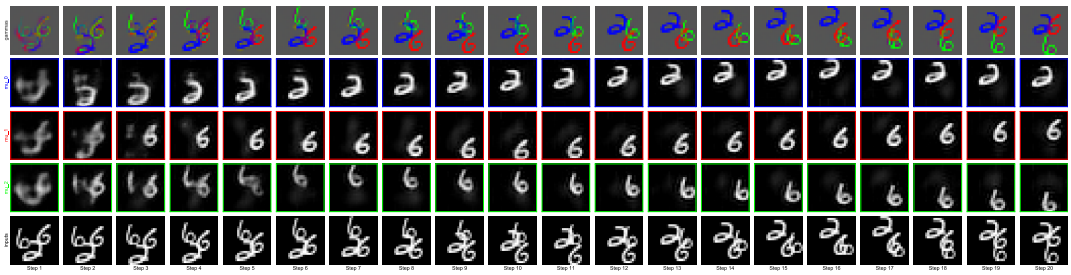


Figure 4.7. A sequence of 3 MNIST digits flying across random trajectories in the image (bottom row). The next-step prediction of each copy of the network (rows 2 to 4) and the soft-assignment of the pixels to each of the copies (top row). Although the network was trained (stage-wise) on sequences with two digits, it is accurately able to separate three digits.

We deploy a slightly deeper version of the architecture used on flying shapes and its precise configuration can be found in Appendix A.1.3. Since the images are gray-scale, we now use a Gaussian distribution for each pixel with fixed $\sigma^2 = 0.25$ and $\mu = \psi_{i,k}$ as computed by each neural network copy. The training procedure is identical to that for flying shapes except that we replace the bitflip noise with masked uniform noise: we first sample a binary mask from a multivariate Bernoulli distribution with $p = 0.2$ and then use this mask to interpolate between the original image and samples from a Uniform(0,1) distribution (range is the same as for the input data).

We train with $K = 2$ and $T = 20$ on flying MNIST having two digits and obtain an AMI score of 0.819 ± 0.022 on the test set. In earlier experiments, we also observed that, given the large variability among the 50 000 unique digits, we can boost the model performance by training in stages using a curriculum of $20, 500, 50\,000$ digits. Here we exploit the generalization capabilities of RNN-EM to quickly transfer knowledge from a less varying set of MNIST digits to unseen variations. We used the same hyperparameter configuration as before and obtain an AMI score of 0.917 ± 0.005 on the test set.

We study the generalization capabilities and robustness of these trained RNN-EM networks by means of three experiments. In the first experiment, we evaluate them on flying MNIST having three digits (one extra) and likewise set $K = 3$. Even without further training, RNN-EM is already able to obtain a high AMI score of 0.729 ± 0.019 (stage-wise: 0.838 ± 0.008) on the test-set. A test example can be seen in Figure 4.7. In the second experiment we test whether the grouping mechanism that has been learned can be transferred to static images. We find that when using 50 steps, RNN-EM is able to transfer a large part of the learned

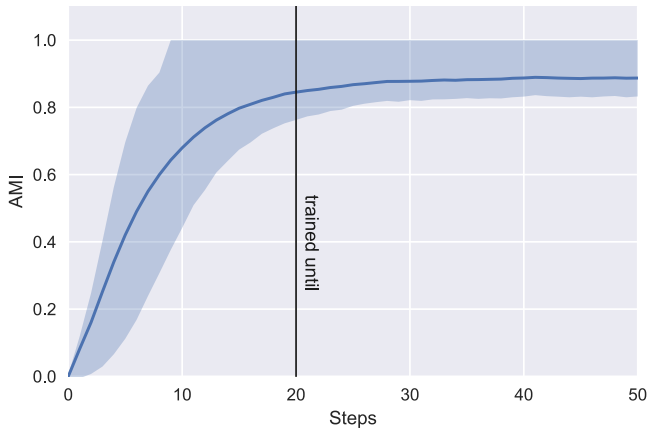


Figure 4.8. Average AMI score (blue line) measured for RNN-EM (trained for 20 steps) on the flying MNIST test-set and corresponding quartiles (shaded areas), computed for each of 50 time-steps. The learned grouping dynamics generalize to longer sequences and even further improve the AMI score.

grouping dynamics and obtains an AMI score of 0.619±0.023 (stage-wise: 0.772± 0.008) for two static digits. In the final experiment, we evaluate the previously trained network on the same dataset for a larger number of timesteps. Figure 4.8 displays the average AMI score across the test set as well as the range of the upper and lower quartile for each time-step.

The results of these experiments confirm our earlier observations for flying shapes, namely that the learned grouping dynamics are robust and generalize across a wide range of variations. Moreover, we find that the AMI score further improves at test time when increasing the sequence length.

4.4 Discussion

In this chapter, we introduced Neural Expectation Maximization (N-EM), a trainable clustering algorithm that combines neural spatial mixture models with generalized Expectation Maximization. N-EM closely resembles a recurrent autoencoder but incorporates several mechanisms to address key aspects of the segregation and representation problem. Regarding the latter, N-EM makes use of instance slots that are described in a common format and separate information about individual objects. It breaks slot symmetries by *iteratively* refining its predictions through competing components, which allows each slot to specialize to a specific object. To use N-EM to learn about objects, we proposed an unsupervised training objective that embraces the idea of objects being modular building blocks

that can be represented efficiently. In our experiments, we were able to confirm that N-EM is able to learn to group pixels according to objects and represent an image as a composition of object representations.

We also introduced RNN-EM, a variation of N-EM that treats each object representation as the hidden state of a recurrent neural network and adds a recurrent update rule. Together with a learned encoder, it parametrizes the M-step and this increased flexibility was shown to yield a significant improvement over N-EM. A key advantage of RNN-EM is that it can be applied to sequential data by modifying its learning objective to make predictions about the future. This makes it easier to learn about objects since pixels belonging to the same object usually share a *common fate* [Hatfield and Epstein, 1985]. Indeed, we were able to show how RNN-EM is able to learn about objects even in the presence of heavy occlusion, and that a representation that explicitly distinguishes individual objects makes it easier to make predictions about their future state in the environment.

As is typical in clustering methods, in (RN)N-EM there is no preferred assignment of objects to groups and so the grouping numbering is arbitrary and only depends on initialization. This property renders our results permutation invariant and naturally allows for *instance segmentation*, as opposed to semantic segmentation where groups correspond to pre-defined categories. (RN)N-EM learns to segment in an unsupervised fashion, which makes it applicable to settings with little or no labeled data. On the downside, this lack of supervision means that the resulting segmentation may not always match the intended outcome. This problem is inherent to this task since in real-world images the notion of an object is task- and context-dependent. A more severe limitation of (RN)N-EM is that the inter-cluster loss hinders in modeling more complex varying backgrounds since the background component would have to predict the pixel prior for predictions made by every other component. This could potentially be resolved by also learning the grouping, which would alleviate the need for this regularization term to discourage undesirable solutions that access information about the future through the E-step.

Another direction to improve (RN)N-EM is to consider a richer feature representation in place of only low-level pixels as input. For example, it is well known that humans extract 3D structure from monocular images very early on in the time course of visual information processing (e.g. Enns and Rensink [1990]). Similarly, RNN-EM may profit from an input representation that additionally considers optical flow, as was also explored in later work [Liu et al., 2019b].

Chapter 5

Relational Neural Expectation Maximization

The focus of this chapter is on the challenge of *composition* that was discussed in Section 3.4. Our aim is to build structured models for inference and prediction that take advantage of the underlying compositionality of object representations to generalize in more predictable and systematic ways. This requires learning about *relations* between object representations that are general, and incorporating a suitable variable binding mechanism so that they can be dynamically applied.

We work towards this goal in the context of common-sense physical reasoning, which is central to many physics-related tasks that humans solve on a daily basis [Lake et al., 2017]. For example, this ability enables humans to make predictions about the consequences of their actions (via simulation and planning) or to infer the state of parts of the world that are currently unobserved. Generally, such a *causal* understanding of the world is assumed to be an essential ingredient for any intelligent agent [Schmidhuber, 1990; Kaelbling, 1993; Ha and Schmidhuber, 2018].

The discovery and representation of objects play an important role in common-sense physical reasoning. They allow humans to decompose a complex visual scene into distinct parts, describe relations between them, and efficiently reason about their dynamics as well as the consequences of their interactions [Battaglia et al., 2013; Ullman et al., 2017]. The most successful machine learning approaches to common-sense physical reasoning incorporate such prior knowledge in their design. They maintain explicit object representations, which allow for

This chapter is based on van Steenkiste et al. [2018a], which was published as a conference paper at *ICLR 2018*. A preliminary version of this work [van Steenkiste et al., 2017] appeared as a workshop paper at *NIPS 2017*.

general physical dynamics to be learned between object pairs in a compositional manner [Chang et al., 2016; Battaglia et al., 2016; Watters et al., 2017]. However, in these approaches learning is *supervised*, as it relies on object representations from external sources (e.g. a physics simulator) that are typically unavailable in real-world scenarios. Neural approaches that learn to directly model motion or physical interactions in pixel space offer an alternative solution [Sutskever et al., 2009; Srivastava et al., 2015]. However, while unsupervised, these methods suffer from a lack of compositionality at the representational level of objects. This complicates learning about complex physical interactions and makes it difficult to generalize to different contexts.

The primary contribution in this chapter is *Relational N-EM* (R-NEM), a structured model for common-sense physical reasoning that learns about physical interactions between objects from raw visual images in a purely *unsupervised* fashion. At its core is *Neural Expectation Maximization* (N-EM) [Greff* and van Steenkiste* et al., 2017a], which learns compositional object representations, but is unable to model relations and interactions between objects. Therefore, we endow a variation of N-EM with a relational mechanism to enable it to model complex interactions between objects, in a way that leverages their underlying compositionality as to generalize more systematically.

5.1 Method

In order to build a structured model for common-sense physical reasoning, we require a method that is capable of learning object representations. One such approach is *Neural Expectation Maximization* (N-EM [Greff* and van Steenkiste* et al., 2017a]), which was described in Chapter 4. In the following, we first offer a computational description of N-EM that focuses on RNN-EM as a (world) model for making predictions about the future (Section 5.1.1). Next, we describe an interaction function that can be combined with RNN-EM to model relations between objects and perform dynamic variable binding (Section 5.1.2).

5.1.1 RNN-EM as a Predictive World Model

RNN-EM consists of K copies (i.e. shared weights) of a recurrent neural network with hidden state $\boldsymbol{\theta}_k$ that can be applied to a sequence of observations $\boldsymbol{x}[1], \ldots, \boldsymbol{x}[T]$ to make *predictions* about the future[1]. It is trained to divide the

[1] In Chapter 4 it was shown how RNN-EM can also be applied to a single static image. Here we will limit our discussion to the sequential setting, which is most relevant in the context of

pixels associated with each observation into groups, such that each group contains the pixels belonging to a single object. In this case, each RNN makes predictions only about the pixels belonging to a single object, such that its hidden state $\boldsymbol{\theta}_k$ yields an object representation.

To divide the input into groups and acquire corresponding representations, RNN-EM alternates between grouping and representation learning (see also Figure 5.1). Starting from the hidden state $\boldsymbol{\theta}_k[t]$, a neural network *decoder* outputs parameters $\psi_{i,k}[t]$ that are associated with pixel-level likelihoods at the *next* timestep $p(\boldsymbol{x}_i[t+1] \mid \psi_{i,k}[t])$. Next, each pixel is soft-assigned to each of the RNNs based on their relative success in modeling $p(\boldsymbol{x}_i[t+1] \mid \psi_{i,k}[t])$ using the E-step (4.3). The resulting grouping is stored in $\boldsymbol{\gamma}[t]$ and the output of RNN-EM at time-step t is used to compute $p(\boldsymbol{x}_i[t+1] \mid \boldsymbol{\theta}[t]) = \sum_k \gamma_{i,k}[t] \cdot p(\boldsymbol{x}_i[t+1] \mid \psi_{i,k}[t])$ for each pixel during training[2].

The representation $\boldsymbol{\theta}_k[t+1]$ at the *next* time-step is computed by first encoding the *top-down error* $\boldsymbol{\gamma}_{:,k}[t] \odot (\boldsymbol{\psi}_{:,k}[t] - \boldsymbol{x}[t+1])$ using a neural network *encoder* (this particular form assumes that $\boldsymbol{\psi}_{:,k}[t]$ are the mean parameters of a simple Bernoulli or Gaussian for each pixel). The top-down error tells each network about the mismatch between its predictions about the future that were made in the previous time-step $\boldsymbol{\psi}_{:,k}[t]$ and the actual future $\boldsymbol{x}[t+1]$ (which is now the present) filtered by the soft-assignment of pixels to networks $\boldsymbol{\gamma}[t]$. The latter is critical since it forces each neural network to focus its predictions on a particular subset of the information content. The representation $\boldsymbol{\theta}_k[t+1]$ is then computed by combining the output of the encoder with the representation at the previous time-step $\boldsymbol{\theta}_k[t]$ using a standard RNN update. Notice how the input (i.e. the top-down error) vanishes as each network becomes better at predicting the future, in which case it becomes possible to simulate future states of the environment purely in latent space [Ha and Schmidhuber, 2018].

RNN-EM is trained on a data set of sequences by backpropagating gradients through this iterative procedure. A two-term loss function (4.5) allows it to learn about statistical regularities in the data set corresponding to objects and acquire object representations. In our case, this loss function is evaluated using observations at the *next* time-step, and we perform additional regularization by adding pixel-level noise to the input. One intuitive interpretation of using denoising or next-step prediction as part of the training objective is to guide the network to learn about essential properties of objects, i.e. those that correspond to the Gestalt Principles of *prägnanz* and *common fate* [Hatfield and Epstein, 1985].

common-sense physical reasoning.

[2]Similar to before, during evaluation and testing we instead use $p(\boldsymbol{x}_i[t+1] \mid \max_k \psi_{i,k}[t])$ to prevent accessing information about the future via the E-step.

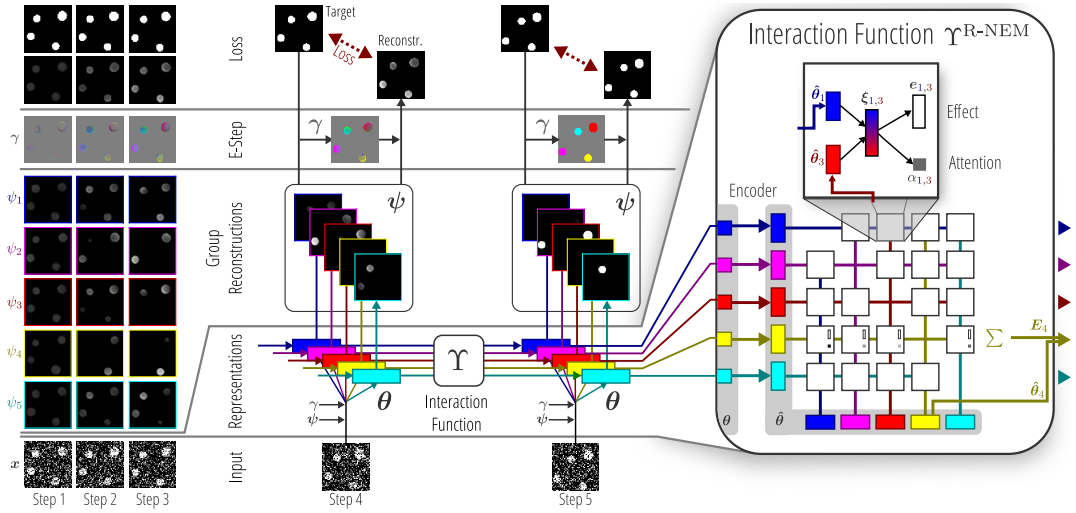


Figure 5.1. Illustration of the different computational aspects of R-NEM when applied to a sequence of images of bouncing balls (bottom row). In the middle, an overview of two iterations of RNN-EM is shown with different colors corresponding to different cluster components (object representations). Here $\boldsymbol{\gamma}, \boldsymbol{\psi}$ at the *Representations* level correspond to the same $\boldsymbol{\gamma}$ (*E-step*), $\boldsymbol{\psi}$ (*Group Reconstructions*) from the previous time-step. On the right side, a computational overview of $\Upsilon^{\text{R-NEM}}$ is shown, which performs physical reasoning by modeling pair-wise interactions between the object representations. The entire system is trained end-to-end in an unsupervised fashion to perform next-step prediction by adapting (4.5) to use observations at the *next* time-step.

5.1.2 Interaction Function

When applied to sequences of images, RNN-EM is able to capture the dynamics of individual objects through the parametrized recurrent connection that acts on the object representation $\boldsymbol{\theta_k}$ across consecutive time-steps. However, the relations and interactions that take place *between* objects can not be captured in this way. To overcome this shortcoming we propose *Relational N-EM* (R-NEM), which adds relational structure to the recurrent update to model interactions between objects without violating key properties of the learned object representations.

Consider a generalized form of the standard RNN-EM dynamics equation, which computes the object representation $\boldsymbol{\theta}_k$ at time t as a function of all object representations $\boldsymbol{\theta} := [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K]$ at the previous time-step through an *interaction function* Υ:

$$\boldsymbol{\theta}_k[t] = \text{RNN}(\hat{\boldsymbol{x}}[t], \Upsilon_k(\boldsymbol{\theta}[t-1])) := \sigma(\boldsymbol{W} \cdot \hat{\boldsymbol{x}}[t] + \boldsymbol{R} \cdot \Upsilon_k(\boldsymbol{\theta}[t-1])). \quad (5.1)$$

Here $\boldsymbol{W}, \boldsymbol{R}$ are weight matrices, σ is the Sigmoid activation function, and $\hat{\boldsymbol{x}}[t]$ is the input to the recurrent model at time t (possibly transformed by an encoder). When $\Upsilon_k(\boldsymbol{\theta}) = \Upsilon_k^{\text{RNN-EM}}(\boldsymbol{\theta}) := \boldsymbol{\theta}_k$, these dynamics coincide with a standard RNN update rule, thereby recovering the original RNN-EM formulation.

The inductive bias incorporated in Υ reflects the modeling assumptions about the interactions between objects in the environment, and therefore the nature of $\boldsymbol{\theta}_k$'s interdependence. If Υ is to incorporate the assumption that no interaction takes place between objects, then the $\boldsymbol{\theta}_k$'s should be fully independent and we recover $\Upsilon^{\text{RNN-EM}}$. On the other hand, if we do assume that interactions among objects take place, but assume very little about the structure of the interdependence between the $\boldsymbol{\theta}_k$'s, then we forfeit useful properties of $\boldsymbol{\theta}_k$ such as their compositionality. In particular, if we naively choose $\Upsilon_k(\boldsymbol{\theta}) = \text{MLP}(\boldsymbol{\theta})$ we can no longer extrapolate learned knowledge to environments with more or fewer than K objects and lose overall data efficiency [Santoro et al., 2017]. Instead, we can make efficient use of compositionality among the learned object representations $\boldsymbol{\theta}_k$ to incorporate general but guiding constraints on how these may influence one another [Battaglia et al., 2016; Chang et al., 2016]. In doing so we constrain Υ to capture the interdependence between the $\boldsymbol{\theta}_k$'s in a compositional manner that enables physical dynamics to be learned efficiently and allow for learned dynamics to be extrapolated to a variable number of objects.

Compositional Interaction Function We propose a parametrized interaction function $\Upsilon^{\text{R-NEM}}$ that incorporates these modeling assumptions and updates $\boldsymbol{\theta}_k$ based on the total effect of the pair-wise interactions between all other objects

$i \neq k$ and k. First, each $\boldsymbol{\theta}_i$ is transformed using an MLP to obtain $\hat{\boldsymbol{\theta}}_i$, which allows information that is relevant for the object dynamics to be made more explicit in the representation:

$$\hat{\boldsymbol{\theta}}_k = \text{MLP}^{enc}(\boldsymbol{\theta}_k). \tag{5.2}$$

Next, each pair $(\hat{\boldsymbol{\theta}}_k, \hat{\boldsymbol{\theta}}_i)_{i \neq k}$ is concatenated and processed by another MLP, which computes a shared embedding $\boldsymbol{\xi}_{k,i}$ that encodes the *causal* interaction between object k and object i:

$$\boldsymbol{\xi}_{k,i} = \text{MLP}^{emb}([\hat{\boldsymbol{\theta}}_k; \hat{\boldsymbol{\theta}}_i]). \tag{5.3}$$

Notice how this is achieved by explicitly distinguishing between the *focus* object k and the *context* object i as also done in prior work [Chang et al., 2016].

From $\boldsymbol{\xi}_{k,i}$ we compute the vector $\boldsymbol{e}_{k,i}$, which encodes the effect of the context object i on the focus object k, and we compute an attention coefficient $\alpha_{k,i}$ that encodes whether interaction between object i and object k takes place, using separate MLPs:

$$\boldsymbol{e}_{k,i} = \text{MLP}^{eff}(\boldsymbol{\xi}_{k,i}). \tag{5.4}$$

$$\alpha_{k,i} = \text{MLP}^{att}(\boldsymbol{\xi}_{k,i}). \tag{5.5}$$

The vector $\boldsymbol{E}_k$ captures the total effect of $\boldsymbol{\theta}_{i \neq k}$ on $\boldsymbol{\theta}_k$ and is computed as a weighted sum of the pair-wise effects multiplied by their attention coefficients:

$$\boldsymbol{E}_k = \sum_{i \neq k} \alpha_{k,i} \cdot \boldsymbol{e}_{k,i}. \tag{5.6}$$

The attention coefficients [Bahdanau et al., 2014; Xu et al., 2015] help select relevant context objects and they can be seen as a more flexible unsupervised replacement of the distance based heuristic that was used in previous work [Chang et al., 2016]. Alternatively, they can be viewed as an implicit mechanism for *dynamic variable binding*, since they determine what information is routed into MLP^{eff}, which essentially is a function of two variables that encodes their interaction[3].

The final output of the interaction function $\Upsilon^{\text{R-NEM}}$ for object k is obtained by concatenating $\hat{\boldsymbol{\theta}}_k$ and $\boldsymbol{E}_k$, which are then combined with the output of the encoder using (5.1). A visual overview of $\Upsilon^{\text{R-NEM}}$ can be seen on the right side of Figure 5.1, which clearly depicts its compositional structure. It can be viewed as a graph, whose nodes correspond to objects and edges reflect relations (interactions)

[3]It is implicit because it only affects which of all the possible ways of routing information into MLP^{eff} ends up being used in the *final* summation in (5.6). Because of this, it is computationally inefficient when the number of objects is large and interactions are sparse.

between the objects. From this perspective, the role of the attention coefficients can also be seen as performing *structure inference*, namely by masking out edges of the otherwise fully-connected interaction graph to infer the correct causal interactions. Finally, it is worth emphasizing that we learn about interactions between objects in the form of a *general* relation (encoded by MLP^{eff}) that is applicable to different pairs of object representations due to them being described in a common format. This makes it easier to apply the learned physical interactions to different contexts as we will find in our experiments.

5.2 Related Work

Machine learning approaches to common-sense physical reasoning can roughly be divided into two groups: symbolic approaches and approaches that perform state-to-state prediction. The former group performs inference over the parameters of a symbolic physics engine [Battaglia et al., 2013; Wu et al., 2015; Ullman et al., 2017], which restricts them to synthetic environments. The latter group employs machine learning methods to make state-to-state predictions, often describing the state of a system as a set of compact object-descriptions that are either used as an input to the system [Grzeszczuk et al., 1998; Fragkiadaki et al., 2015; Battaglia et al., 2016; Chang et al., 2016] or for training purposes [Watters et al., 2017]. By incorporating information (e.g. position, velocity) about objects these methods have achieved excellent generalization and simulation capabilities.

Purely unsupervised approaches for state-to-state prediction [Sutskever et al., 2009; Michalski et al., 2014; Agrawal et al., 2016; Lerer et al., 2016] that use raw visual inputs as state-descriptions have yet to rival these capabilities. Our method is the first purely unsupervised state-to-state prediction method that operates in pixel space while also leveraging the underlying compositionality of learned object representations. In that sense, it takes a first step towards common-sense physical reasoning under more realistic real-world assumptions.

The proposed interaction function $\Upsilon^{\text{R-NEM}}$ can be viewed as a type of Message Passing Neural Network (MPNN; Gilmer et al. [2017]) that incorporates a variant of neighborhood attention [Duan et al., 2017]. In that case, the 'effect vector' acts as the edge embedding (or message) that is passed to the receiving nodes (if a connection is present) to update the node embedding (object representation). It suggests that one could iterate these two stages to allow for higher-order interactions between the objects, but we did not explore this connection further in this work. In light of other recent work [Zaheer et al., 2017] $\Upsilon^{\text{R-NEM}}$ can also be seen as a permutation equivariant set function.

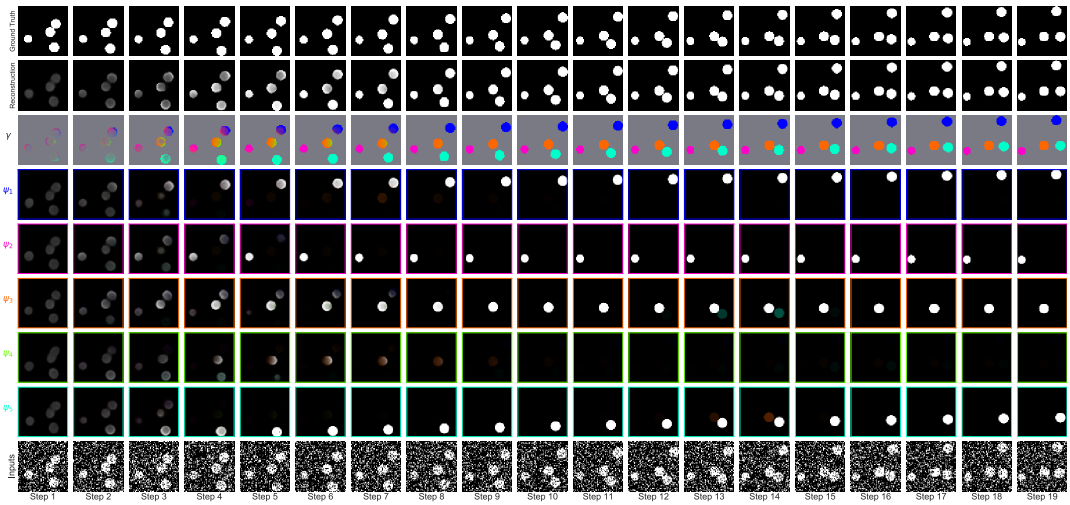


Figure 5.2. R-NEM applied to a sequence of 4 bouncing balls. Each column corresponds to a time-step, which coincides with an approximate EM step. At each time-step, R-NEM computes $K = 5$ new representations $\boldsymbol{\theta}_k$ according to (5.1) (see also *Representations* in Figure 5.1) from the input $\boldsymbol{x}$ with added noise (bottom row). From each new $\boldsymbol{\theta}_k$ the mean parameters $\boldsymbol{\psi}_{:,k}$ of the pixel-level likelihoods are produced (rows 2-6 from bottom) that model the state of the environment at the *next* time-step. Attention coefficients are visualized by overlaying a colored reconstruction of a context object on the focus object (see *Attention* in Section 5.3 for details). Based on the prediction accuracy of $\boldsymbol{\psi}$, the *E-step* (see Figure 5.1) computes new soft-assignments $\boldsymbol{\gamma}$ (row 7 from bottom), visualized by coloring each pixel i according to their distribution over components $\boldsymbol{\gamma}_i$. Row 8 visualizes the total prediction by the network ($\sum_k \boldsymbol{\psi}_{:,k} \cdot \boldsymbol{\gamma}_{:,k}$) and row 9 the ground-truth sequence at the next time-step.

Other related works have also taken steps towards combining the learnability of neural networks with the compositionality of symbols in modeling physics [Chang et al., 2016; Battaglia et al., 2016], playing games [Kansky et al., 2017; Denil et al., 2017], learning algorithms [Reed and de Freitas, 2015; Li et al., 2017; Cai et al., 2017; Bošnjak et al., 2017], visual understanding [Johnson et al., 2017b; Ellis et al., 2018], and natural language processing [Andreas et al., 2016; Hu et al., 2017].

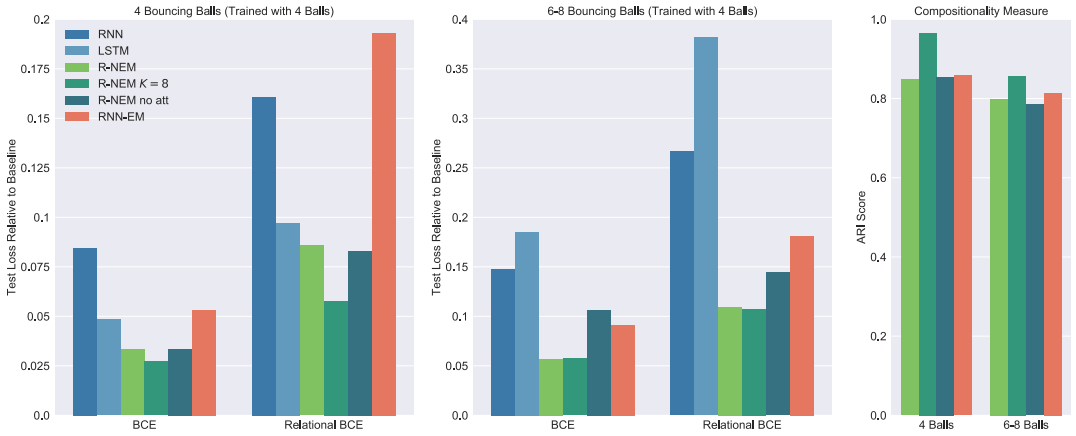


Figure 5.3. Performance of each method on the *bouncing balls* task. Each method was trained on a data set with 4 balls and evaluated on a test set with 4 balls (left), and on a test-set with 6-8 balls (middle). The losses are reported relative to the loss of a baseline for each data set that always predicts the current frame. The ARI score (right) is used to evaluate the degree of compositionality that is achieved.

5.3 Experiments

In this section, we evaluate R-NEM on three different physical reasoning tasks that each vary in their dynamical and visual complexity: bouncing balls with variable mass, bouncing balls with an invisible curtain, and the Arcade Learning Environment [Bellemare et al., 2013]. We compare R-NEM to other *unsupervised* neural methods but that lack an inductive bias aimed at modeling relations between learned object representations. Our results demonstrate that adding such structure is indeed beneficial, which validates our design choices, and highlight the usefulness of object representations for more systematic generalization[4].

All experiments use ADAM [Kingma and Ba, 2015] with default parameters, 50K train, 10K validation, and 10K test sequences, and early stopping with a patience of 10 epochs. For each of $\text{MLP}^{enc}, \text{MLP}^{emb}, \text{MLP}^{eff}$ we use a different single layer neural network with 250 *ReLU* units. For MLP^{att} we use a two-layer neural network: 100 *Tanh* units followed by a single *Sigmoid* unit. A detailed overview of the experimental setup can be found in Appendix A.2.

[4]Code is available at `https://github.com/sjoerdvansteenkiste/Relational-NEM`.

5.3.1 Bouncing Balls

We study the physical reasoning capabilities of R-NEM on the *bouncing balls* task, a standard environment to evaluate physical reasoning capabilities that exhibits low visual complexity but complex non-linear physical dynamics[5]. We train R-NEM on sequences of 64×64 binary images over 30 time-steps that contain four bouncing balls with different masses corresponding to their radii. The balls are initialized with random initial positions, masses, and velocities. Balls bounce elastically against each other and the image window.

Qualitative Evaluation Figure 5.2 presents a qualitative evaluation of R-NEM on the bouncing balls task. After 10 time-steps it can be observed that the pixels that belong to each of the balls are grouped together and assigned to a unique component (with a saturated color); and that the background (colored grey) has been divided among all components. This indicates that the representation $\boldsymbol{\theta}_k$ from which each component produces the predictions $\boldsymbol{\psi}_{:,k}$ does indeed only contain information about a unique object, such that together the $\boldsymbol{\theta}_k$'s yield a compositional object representation of the scene. The total prediction (that combines the predictions for each group and the soft-assignments) yields an accurate depiction of the input sequence at the next time-step, indicating that R-NEM has learned to model the dynamics of bouncing balls.

Comparison We compare the modeling capabilities of R-NEM to an RNN, LSTM [Hochreiter and Schmidhuber, 1997; Gers et al., 1999], and RNN-EM in terms of the Binary Cross-Entropy (BCE) loss between the predicted image and the ground-truth image of the last frame, as well as the *relational* BCE that only takes into account those objects that take part in a collision. Unless specified we use $K = 5$.

On a test set with sequences containing four balls, we observe that R-NEM produces markedly lower losses when compared to all other methods (left plot in Figure 5.3). Moreover, to validate that each component captures only a single ball (and thus compositionality is achieved), we report the Adjusted Rand Index (ARI [Hubert and Arabie, 1985]) score between the soft-assignments $\boldsymbol{\gamma}$ and the ground-truth assignment of pixels to objects. In the left column of the ARI plot (right side in Figure 5.3) we find that R-NEM achieves an ARI score of 0.8, which roughly indicates that in 80% of the cases each ball is modeled by a single component. This suggests that a compositional object representation is achieved for most of the sequences. Together these observations are in line with our qualitative

[5]Videos are available at `https://sites.google.com/view/r-nem-gifs/`.

evaluation and validate that incorporating real-world priors is greatly beneficial (comparing to RNN, LSTM) and that $\Upsilon^{\text{R-NEM}}$ enables interactions to be modeled well compared to RNN-EM in terms of the relational BCE.

Similar to the results presented in Chapter 4, we find that further increasing the number of components during training (leaving additional groups empty) increases the quality of the grouping, see R-NEM $K = 8$ in Figure 5.3. In addition, we observe that the loss (in particular the relational BCE) is reduced further, which is in line with our hypothesis that compositional object representations are greatly beneficial for modeling physical interactions.

Extrapolating Learned Knowledge We use a test set with sequences containing 6-8 balls to evaluate the ability of each method to *extrapolate* their learned knowledge about physical interactions between four balls to environments with more balls. We use $K = 8$ when evaluating R-NEM and RNN-EM on this test-set to accommodate the increased number of objects. As can be seen from the middle plot in Figure 5.3, R-NEM again greatly outperforms all other methods. Notice that, since we report the loss relative to a baseline, we roughly factor out the increased complexity of the task. Perfect extrapolation of the learned knowledge would, therefore, amount to *no change* in relative performance. In contrast, we observe far worse performance for the LSTM (relative to the baseline) when evaluated on this data set with extra balls. It suggests that the gating mechanism of the LSTM has allowed it to learn a sophisticated and overly specialized solution for sequences with four balls that does not generalize to a data set with 6-8 balls.

R-NEM and RNN-EM scale markedly better to this data set than LSTM, although the RNN also appears to suffer to a lesser extent from this type of "overfitting". We speculate that this is because of its inability to learn a reasonable solution on sequences of four balls to begin with. In general, we conclude that the superior extrapolation capabilities of RNN-EM and R-NEM are inherent to their ability to factor a scene in terms of compositional object representations.

Attention Further insight in the role of the attention mechanism can be gained by visualizing the attention coefficients, as is done in Figure 5.2. For each component k we draw $\alpha_{k,k'} \cdot \boldsymbol{\psi}_{k'}$ on top of the reconstruction $\boldsymbol{\psi}_{:,k}$, colored according to the color of component k'. These correspond to the colored balls (that are for example seen in time-steps 13, 14), which indicate whether component k took information about component k' into account when computing the new state (recall (5.6)). It can be observed that the attention coefficient $\alpha_{k,k'}$ becomes non-zero whenever collision takes place, as is evident from the colored ball being

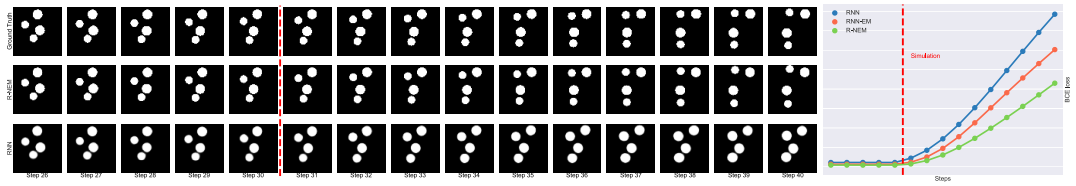


Figure 5.4. **Left**: Three sequences of 15 time-steps corresponding to ground-truth (top), R-NEM (middle), RNN (bottom). The last ten time-steps of the sequences produced by R-NEM and RNN are simulated. **Right**: The BCE loss on the entire test-set for these same time-steps.

visible in the *following* time-steps. The attention mechanism learned by R-NEM thus assumes the role of inferring the underlying interaction graph, matching our own intuitions of how this mechanism would best be utilized.

A quantitative evaluation of the attention mechanism is obtained by comparing R-NEM to a variant of itself that does not incorporate attention (*R-NEM no att*). Figure 5.3 shows that both methods perform equally well on the regular test set (4 balls), but that *R-NEM no att* performs worse at extrapolating from its learned knowledge (6-8 balls). A likely reason for this behavior is that the range of the sum in (5.6) changes with K. Thus, when extrapolating to an environment with more balls, the total sum may exceed previous boundaries and impede learned dynamics.

Simulation Once a scene has been accurately modeled, R-NEM can approximately simulate its dynamics through recursive application of (5.1) for each $\boldsymbol{\theta}_k$[6]. In Figure 5.4, we compare the simulation capabilities of R-NEM to RNN-EM and an RNN on the bouncing balls environment[5]. On the left, it displays a sequence with five normal steps followed by 10 simulation steps for R-NEM and an RNN, as well as the ground-truth sequence. From the last frame in the sequence, it can clearly be observed that R-NEM has managed to accurately simulate the environment. Each ball is approximately in the correct place, and the shape of each ball is preserved. The balls simulated by the RNN, on the other hand, deviate substantially from their ground-truth position and their size has increased. In general, we find that R-NEM produces mostly very accurate simulations, whereas the RNN consistently fails. Interestingly, we found that the cases in which R-NEM frequently fails are those for which a single component models more than one

[6]Note that in this case the input to the neural network encoder in component k corresponds to $\boldsymbol{\gamma}_{:,k}[t-1] \odot (\boldsymbol{x}[t] - \boldsymbol{\psi}_{:,k}[t-1])$, such that the output of the encoder vanishes when $\boldsymbol{\psi}_{:,k}[t-1] = \boldsymbol{x}[t]$.

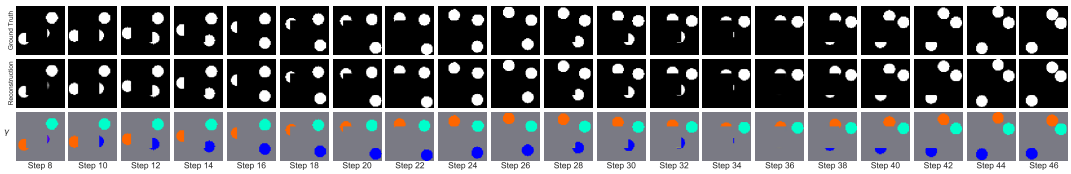


Figure 5.5. R-NEM applied to a sequence of bouncing balls with an invisible curtain. The ground truth sequence is displayed in the top row, followed by the prediction of R-NEM (middle) and the soft-assignments of pixels to components (bottom). R-NEM models objects, as well as their interactions, even when certain objects are completely occluded (step 36). Only a subset of the steps is shown.

ball. The right side of Figure 5.4 summarizes the BCE loss for these same time-steps across the *entire* test-set. Although this is a crude measure of simulation performance, since it does not take into account the identity of the balls, we still observe that R-NEM consistently outperforms RNN-EM and an RNN.

5.3.2 Hidden Factors

Occlusion is abundant in the real world, and the ability to model (temporarily) unobserved objects is crucial for any physical reasoning system. We therefore evaluate the capability of R-NEM to handle occlusion using a variant of bouncing balls that contains an invisible "curtain". Figure 5.5 shows that R-NEM accurately models the sequence and can maintain object states, even when confronted with occlusion[5]. For example, notice how in step 36 the "blue" ball is completely occluded and is about to collide with the "orange" ball. Nonetheless, in step 38 the ball is accurately predicted to re-appear at the bottom of the curtain (since collision took place) as opposed to the left side of the curtain. This demonstrates that R-NEM has a notion of *object permanence* and implies that it understands the scene on a level beyond pixels: it assigns persistence and identity to the objects as a result of its (architectural) inductive bias.

In terms of test-set performance we find that R-NEM (*BCE:* 46.22, *relational BCE:* 2.33) outperforms an RNN (*BCE:* 94.64, *relational BCE:* 4.14) and an LSTM (*BCE:* 59.32, *relational BCE:* 2.72).

5.3.3 Space Invaders

To test the performance of R-NEM in a visually more challenging environment, we train it on sequences of 84×84 binarized images over 25 time-steps of game-play on Space Invaders from the Arcade Learning Environment [Bellemare et al.,

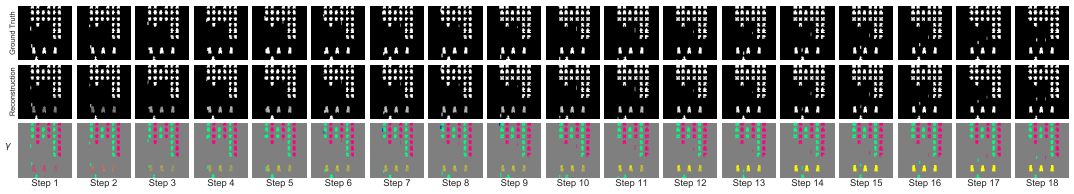


Figure 5.6. R-NEM accurately models a sequence of frames obtained by an agent playing Space Invaders. A group no longer corresponds to a single object, but instead assumes the role of high-level entities that engage in similar movement patterns.

2013][7]. We use $K = 4$ and also feed the action of the agent to the interaction function. Figure 5.6 confirms that R-NEM is able to accurately model the environment, even though the visual complexity has increased. Notice that these visual scenes comprise a large number of (small) primitive objects that behave similarly. Since we trained R-NEM with four components it is unable to group pixels according to individual objects and is forced to consider a different grouping. We find that R-NEM assigns different groups to every other column of aliens together with the space ship, and to the three large 'shields'. These groupings seem to be based on movement, which to some degree coincides with their semantic roles in the environment. In other examples (not shown) we also found that R-NEM frequently assigns different groups to every other column of the aliens, and to the three large 'shields'. Individual bullets and the space ship are less frequently separated, which may have to do with the action-noise of the environment (that controls the movement of the space ship) and the small size of the bullets at the current resolution that makes them less predictable.

5.4 Discussion

In this chapter, we have focused on the challenge of *composition*. We introduced Relational Neural Expectation Maximization (R-NEM), a structured model for common-sense physical reasoning that combines the learned object representations of RNN-EM with a compositional interaction function to model relations between objects. Due to the underlying compositionality of object representations and a mechanism for dynamic variable binding, it is able to learn about physical interactions between objects in a way that can be extrapolated to scenes consisting

[7]Binarization ensures that the color group of the entities on the screen does not give away their grouping.

of more or fewer objects.

In our experiments, we found that R-NEM indeed captures the (physical) dynamics of various environments more accurately than other methods and that it is able to generalize from its learned knowledge more systematically in this way. We also found that R-NEM can be used as an approximate simulator of the environment, and to predict movement and collisions of objects, even when they are completely occluded. This demonstrates a notion of object permanence and aligns with evidence that young infants seem to infer that occluded objects move in connected paths and continue to maintain object-specific properties [Spelke, 1990]. Moreover, young infants also appear to expect that objects only interact when they come into contact [Spelke, 1990], which is analogous to the behavior of R-NEM to only attend to other objects when a collision is imminent. In that sense, we argue that our method presents a step towards learning a more human-like model of the world in a completely unsupervised fashion.

Current limitations of our approach revolve around grouping and prediction. We found that the interaction in the E-step among the groups makes it difficult to increase the number of components above ten without causing harmful training instabilities. Similarly, the computational complexity of the proposed interaction function scales quadratically in the number of components due to the choice of variable binding mechanism. In terms of prediction, we have implicitly assumed that objects in the environment behave according to rules that can be inferred. This poses a challenge when objects deform in a manner that is difficult to predict (as is the case for objects in Space Invaders due to downsampling). However, in practice, we find that (once pixels have been grouped together) the masking of the input helps each component in quickly adapting its representation to any unforeseen behavior across consecutive time-steps.

Chapter 6

Object Compositionality in GANs

In this chapter we investigate an alternative approach to learning object representations based on more powerful implicit generative models. Using a *generative* approach, our goal is to recover the process according to which the observed data was generated, which in the case of images requires learning about objects, background, and their interactions. Using an appropriate inference procedure, this knowledge can then be extracted.

Our focus is on *deep* generative models, which leverage the expressiveness of deep neural networks to learn the generative process almost entirely from data [Goodfellow et al., 2014; Kingma and Welling, 2014; Van Oord et al., 2016]. In this case, the structure of the generative model (and thereby the interpretation of any latent variables) is mostly due to the *inductive bias* of the neural network, since this affects how it learns to generate the data and what abstractions are considered [Dinh et al., 2017; Donahue et al., 2017; Dumoulin et al., 2017]. For example, when a deep generative model is equipped with an inductive bias for disentanglement, it has shown capable of learning to generate images of human faces in a way that allows meaningful factors of variation, such as pose and lighting, to be recovered via inference [Chen et al., 2016; Higgins et al., 2017a].

In order to use a deep generative model to learn about objects, it should learn a generative process that explicitly considers such abstractions at a representational level. This requires incorporating a corresponding inductive bias that encourages the learned generative process to be *compositional*, in the sense that visual scenes are generated as a composition of reusable parts (i.e. objects). By 'inverting' this generative process, we can then attempt to recover knowledge about the

This chapter is based on van Steenkiste et al. [2020], which was published as a journal article in *Neural Networks*. A preliminary version of this work [van Steenkiste et al., 2018b] appeared as a workshop paper at *NeurIPS 2018*.

individual parts.

The primary contribution of this chapter is to formulate an implicit deep generative model based on *Generative Adversarial Networks* (GANs) that includes a corresponding inductive bias. We investigate a minimal modification to a standard neural network generator to allow for compositionality and demonstrate how it enables GANs to learn about objects, without prior access to this information. Using this general design as a backbone, we then propose two useful extensions that enable the generator to model dependencies between objects and to model background. This makes it possible to use this approach for segregation on more complex images that previous approaches, including N-EM (see Chapter 4), struggle at. To that extent, we demonstrate how one can leverage the learned structured generative process, which is now interpretable and semantically understood, to perform inference and recover information about individual objects without additional supervision. Finally, we demonstrate how our structured GAN is better at generating multi-object images that are more faithful to the reference distribution compared to standard GAN approaches.

6.1 Method

We investigate how we can leverage the power of implicit generative models [Mohamed and Lakshminarayanan, 2017] to learn about objects through the process of synthesizing images. If successful, then this offers an alternative approach to learning object representations that is expected to be more easily adaptable to complex datasets. In the following, we will first briefly introduce GANs, which are our choice of implicit generative models (Section 6.1.1). Next, we describe several architectural modifications that can be incorporated in the generator to guide it to consider objects at a representational level (Section 6.1.2).

6.1.1 Generative Adversarial Networks

We will make use of Generative Adversarial Networks (GANs) [Goodfellow et al., 2014][1], which are powerful implicit generative models that learn a stochastic procedure to generate samples from a distribution p_X. We chose GANs for two reasons. Firstly, despite their optimization difficulties, they have shown remarkably successful at generating complex (high-resolution) images [Brock et al., 2019]. Secondly, it has been shown that by incorporating structure in (the generator of)

[1] There exist interesting parallels between GANs and earlier approaches [Schmidhuber, 1990, 1992b], and we refer the reader to Schmidhuber [2020] for a comparison.

a GAN it is possible to exert considerable influence over the learned generative process, which is especially important for representation learning purposes [Chen et al., 2016; Lin et al., 2018; Nguyen-Phuoc et al., 2019].

Traditionally GANs consist of two deterministic functions: a generator $G(\boldsymbol{z})$ and a discriminator (or critic) $D(\boldsymbol{x})$. The goal is to find a generator that accurately transforms samples from a prior distribution $\boldsymbol{Z} \sim p_Z$ to match samples from the target distribution $\boldsymbol{X} \sim p_X$. This can be done by using the discriminator to implement a suitable objective for the generator, in which it should behave *adversarially* with respect to the goal of the discriminator in determining whether samples $\boldsymbol{x}$ were obtained from p_X or $p_{G(Z)}$ respectively. These objectives can be summarized as a *minimax* game with the following value function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{X\sim p_X}[\log D(\boldsymbol{x})] + \mathbb{E}_{Z\sim p_Z}[\log(1-D(G(\boldsymbol{z})))]\,. \tag{6.1}$$

When the generator and the discriminator are implemented with neural networks, optimization may proceed through alternating (stochastic) gradient descent updates of their parameters with respect to (6.1). However, in practice this procedure might be unstable and the minimax formulation is known to be hard to optimize. Many alternative formulations have been proposed and we refer the reader to Lucic et al. [2018] and Kurach et al. [2019] for a comparison.

Based on the findings of Kurach et al. [2019] we consider two practical reformulations of (6.1) in this work: Non-Saturating GAN (NS-GAN) [Goodfellow et al., 2014], in which the generator maximizes the probability of generated samples being real, and Wasserstein GAN (WGAN) [Arjovsky et al., 2017] in which the discriminator minimizes the Wasserstein distance between $p_{G(Z)}$ and p_X. In both cases we consider two additional techniques to improve optimization that have proven to work best on a variety of datasets and architectures: the gradient penalty from Gulrajani et al. [2017] to regularize the discriminator, and spectral normalization [Miyato et al., 2018] to normalize its gradients.

6.1.2 Incorporating Architectural Structure

We propose three architectural changes to the *generator* of a GAN so that it learns a generative process based on object representations. In the following, we first present a simple modification to a neural network generator to facilitate a generative process that is *compositional*, which is the main type of invariance that objects provide. Afterward, we present two useful extensions that allow it to additionally reason about relations between objects, and to explicitly model background and occlusion.

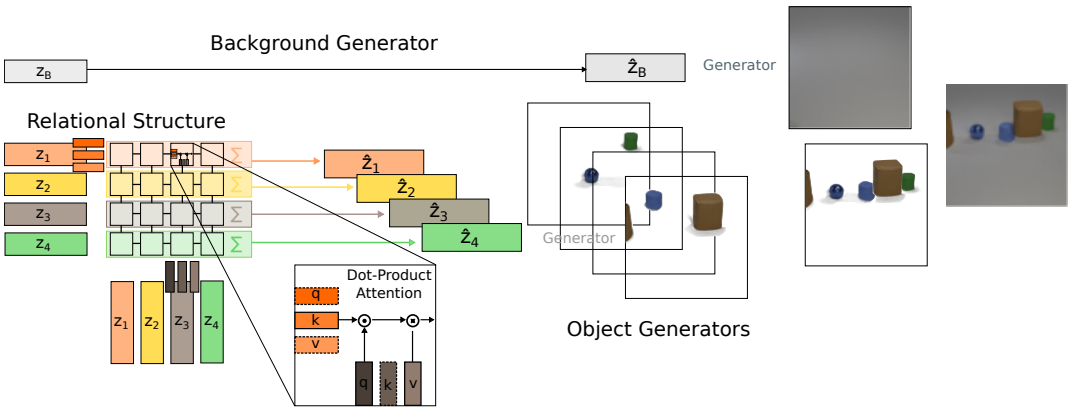


Figure 6.1. We propose three modifications to a standard neural network generator to generate images as a composition of individual objects and background. In this case, it consists of $K = 4$ *object generators* (shared weights) that each generate an image from separate latent vector $\boldsymbol{z}_i$, which serve as the backbone for compositionality. On the left side, *relational structure* is shown as one possible extension to model relations between objects by first computing $\hat{\boldsymbol{z}}_i$ from $\boldsymbol{z}_i$ that are then fed to the object generators. At the top, a second extension is shown that incorporates a *background generator* (unique weights) to generate a background image from a separate latent vector $\boldsymbol{z}_b$. The whole system is trained end-to-end as in the standard GAN framework, and the final image is obtained by composing (here using alpha compositing) the outputs of all generators.

Compositionality at the Representational Level of Objects

A minimal modification to a neural network generator is to assume that images $\boldsymbol{x}$ are composed of objects that are independent of one another. For images having K objects, we consider K i.i.d. vector-valued random variables $\boldsymbol{Z}_i \sim p_{\boldsymbol{Z}}$ that each describe an object at a representational level. K copies of a deterministic generator $G(\boldsymbol{z})$ transform samples from each $\boldsymbol{Z}_i$ into images, such that their superposition results in the corresponding output image:

$$\boldsymbol{x} = \sum_{i=1}^{K} G(\boldsymbol{z}_i). \tag{6.2}$$

When each copy of G generates an image of a single object, the resulting generative model efficiently generates images in a compositional manner (Figure 6.2). Each object in (6.2) is described in a common format (i.e. the $\boldsymbol{Z}_i$'s are i.i.d) and the weights among the generators are shared, such that any acquired knowledge in generating a specific object is transferred across all others. Hence,

rather than having to learn about all combinations of objects (including their own variations) that may appear in an image, it suffices to learn about the different variations of each individual object. This greatly simplifies the generative process without losing generality.

The superposition of generators in (6.2) can be viewed as a single generator of the form $G_K(\boldsymbol{z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_K])$ and trained as before using the objective in (6.1). The generators in (6.2) do not interact, which prevents degenerate solutions and encourages G to learn about modular parts that correspond to objects. On the other hand, this also implies that relations between objects *within* a scene cannot be modeled in this way. Finally, we note that the sum in (6.2) assumes that images only consist of objects and that their values can be summed in pixel-space. We will now present two extensions that focus on each of these aspects and incorporate additional structure besides the superposition of generators that serves as the basis for object compositionality.

Modeling Relations Between Objects

In the real world, objects are not strictly independent of one another. Certain objects may only occur in the presence of others or affect their visual appearance in subtle ways (e.g. shadows). To facilitate relationships of this kind, the first extension we propose consists of *relational structure*, in which the representation of an object is updated as a function of all others before each generator proceeds to generate its image.

The relational structure is implemented by a graph neural network which, in this case, consists of one or more self-attention blocks that compute these updates [Battaglia et al., 2018]. At the core of each attention block is Multi-Head Dot-Product Attention (MHDPA) [Vaswani et al., 2017] that performs message-passing when one associates each object representation with a node in a graph [Gilmer et al., 2017]. This is a natural choice since graph neural networks of this kind were previously found to excel at modeling interactions between objects at a representational level in a way that reflects their underlying relationships [Zambaldi et al., 2019].

Similar to Zambaldi et al. [2019], a single 'head' of an attention block first projects each latent vector $\boldsymbol{z}_i$ (associated with an object) into a query, key, and value vector:[2]

[2]The role of the key and value vectors are analogous to those in a key-value database. However, in this case, access takes place on the basis of similarity between key and query vectors (i.e. the attention weights), which allows this process to be differentiated.

$$\boldsymbol{q}_i = \text{MLP}^{query}(\boldsymbol{z}_i),\ \boldsymbol{k}_i = \text{MLP}^{key}(\boldsymbol{z}_i),\ \boldsymbol{v}_i = \text{MLP}^{value}(\boldsymbol{z}_i), \tag{6.3}$$

where $dim(\boldsymbol{q}_i) = dim(\boldsymbol{k}_i) = dim(\boldsymbol{v}_i) = d$.

Next, the interaction of an object k with all other objects $i = 1, \ldots, K$ (including itself) is computed as a weighted sum of their value vectors, where the weights (attention coefficients) are computed as dot-products between its query vector and all key vectors, followed by softmax normalization:

$$a_{k,i} = \frac{\exp\{\boldsymbol{q}_k^\intercal \cdot \boldsymbol{k}_i / \sqrt{d}\}}{\sum_{j=1}^{K} \exp\{\boldsymbol{q}_k^\intercal \cdot \boldsymbol{k}_j / \sqrt{d}\}}. \tag{6.4}$$

$$\boldsymbol{e}_k = \sum_{i=1}^{K} a_{k,i} \cdot \boldsymbol{v}_i. \tag{6.5}$$

The update vector $\boldsymbol{e}_k$ now contains the result of attending to all other object representations, or equivalently, a representation of the incoming 'messages' $\boldsymbol{v}_i$ that are used to update the representation belonging to each node (in this case node k) in the underlying interaction graph. Similar to the interaction function presented in Chapter 5, the role of the attention coefficients $a_{k,i}$ is then to mask out edges when no interaction should take place.

In order to compute the final update to the object representations (node representations) we project the update vector $\boldsymbol{e}_i$ back to the original size of $\boldsymbol{z}_i$:

$$\hat{\boldsymbol{z}}_i = \text{MLP}^{project}(\boldsymbol{e}_i) + \boldsymbol{z}_i. \tag{6.6}$$

Additional heads (capable of modeling interactions due to other relations) use different parameters, and their outputs are concatenated and fed to a separate MLP to arrive at a final $\hat{\boldsymbol{z}}_i$. Similarly, more complex interactions can be modeled by using multiple attention blocks to *iteratively* update $\boldsymbol{z}_i$ through repeated message passing. Details about the exact implementation of these computations can be found in Appendix A.3 and a schematic in Figure 6.1.

Background and Occlusion

The second extension that we propose explicitly distinguishes what we refer to as 'background' from objects. Background contains the remaining information content of an image that does not occur frequently enough in the data to be modeled separately as objects (or that lacks a regular visual appearance). This is often encountered when modeling more complex visual scenes, where the

observations a learner is given access to are typically biased towards a particular context, in the sense that only certain invariances can be observed.

One assumption that we have made is that objects can be compactly encoded in a representation $\boldsymbol{z}_i$ that is described in a *common format*. Treating background as an extra "object" violates this assumption as the latent representations $\boldsymbol{z}_i$ (and corresponding generator) now need to describe objects that assume a regular visual appearance, as well as the remaining background that is not regular in its visual appearance at all. Therefore, we consider an additional *background* generator (see also Figure 6.1) having its own set of weights to generate the background from samples from a separate vector-valued latent variable $\boldsymbol{Z}_b \sim p_{\boldsymbol{Z}_b}$. We will explore two different variations of this addition, one in which $\boldsymbol{z}_b$ participates in the relational structure, and one in which it does not.

A remaining challenge is then in *combining* objects with background and in modeling occlusion. A straightforward adaptation of the sum in (6.2) to incorporate pixel-level weights (e.g. as in N-EM) would require the background generator to assign a weight of zero to all pixel locations where objects appear, thereby increasing the complexity of generating the background exponentially. Instead, we require the object generators to generate an additional alpha channel for each pixel (while treating the output of the background generator as opaque), and use alpha compositing to combine the K different outputs of the object generators $(\boldsymbol{x}_i, \boldsymbol{\alpha}_i)$ and background generator $(\boldsymbol{x}_b, 1)$ as follows:

$$\boldsymbol{x} = \sum_{i=1}^{K} \left[\boldsymbol{x}_i \boldsymbol{\alpha}_i \prod_{j=1}^{i-1} (1 - \boldsymbol{\alpha}_j) \right] + \boldsymbol{x}_b \prod_{i=1}^{K} (1 - \boldsymbol{\alpha}_i). \tag{6.7}$$

Using alpha compositing as in (6.7) is a standard technique in computer graphics, and therefore a natural choice for synthesizing images. A potential disadvantage of alpha compositing is that it uses a fixed ordering, which assumes that the generated images (when overlapping) are ordered correctly. In principle, the relational structure can ensure that this is the case, although this may be difficult to learn in an adversarial setting. An alternative choice is to *learn* to composit, for example by using a conditional GAN that starts from images of individual objects [Lin et al., 2018; Azadi et al., 2019], but which is not explored here.

6.2 Related Work

Prior works on incorporating an inductive bias aimed at object compositionality also model an image as a *spatial mixture* of image patches, and utilize multiple copies of the same function to arrive at a compositional solution. Different

implementations use RBMs [Le Roux et al., 2011], VAEs [Nash et al., 2017], or (recurrent) auto-encoders inspired by EM(-like) inference procedures as in Chapter 4 [Greff* and van Steenkiste* et al., 2017a] (but see also Greff et al. [2016]) to model these patches. It was also shown that interactions between objects can be modeled efficiently in this framework [van Steenkiste et al., 2018a] (Chapter 5). In contrast, neither of these approaches have shown to be capable of modeling more complex visual scenes that incorporate unstructured background as well as relations between objects. By observing that GANs are often superior in terms of image generation capabilities, and by adding structure, we are able to improve upon these works in this regard. Indeed, compared to the later proposed IODINE [Greff et al., 2019] that also utilizes a spatial mixture formulation (but is based on iterative amortized variational inference [Marino et al., 2018]), we will demonstrate in Section 6.3.2 how our approach better succeeds at modeling complex image distributions.

A conceptually similar line of related work uses variational inference to learn recurrent neural networks to *iteratively* generate an image, one patch at a time [Gregor et al., 2015; Eslami et al., 2016; Kosiorek et al., 2018]. The work by Eslami et al. [2016] incorporates a strong inductive bias that associates each image patch with a single object. However, also their approach is limited to generating (sequences) of binary images without background [Eslami et al., 2016; Kosiorek et al., 2018]. Related settings have also been explored with GANs using a recurrent generator [Im et al., 2016; Kwak and Zhang, 2016], while other work [Yang et al., 2017] additionally considers a separate generator for the background that uses spatial transformations to integrate a foreground image. From these, only the work of Yang et al. [2017] briefly explores the problem of multi-object image generation on a dataset consisting of two non-overlapping MNIST digits. Their approach is only moderately successful while making extra assumptions about the size of the digits. Importantly, their approach does not provide a means to 'invert' the learned model and perform inference.

Other recent work in GANs has focused on *conditional* image generation to simplify the task of multi-object image generation. Johnson et al. [2018] generate multi-object images from explicit scene graphs, while Xu et al. [2018] condition on a stochastic and-or graph instead. Azadi et al. [2019] propose a framework to generate images composed of two objects by combining the images of each individual object. Similarly, Lin et al. [2018] present an iterative scheme to remove or add objects to a scene based on prior knowledge about the individual objects. Hinz et al. [2019] require object labels and bounding boxes to generate complex visual scenes consisting of multiple objects and background. While these works generate realistic multi-object scenes, they require prior information (scene

graphs, segmentations, etc.) about individual objects or scenes that is typically not available in many real-world settings. Our approach serves a complementary purpose in terms of image generation in that regard: while we consider visually simpler scenes, we do not require extra information about scenes or objects. In the context of representation learning, we are also able to 'invert' the learned generative model by leveraging its structure to learn about objects. This sets us firmly apart from these methods and also from standard unstructured GANs that do not require conditioning. Indeed, as we have argued, without explicitly considering objects at a representational level (e.g. in the form of architectural structure), one is unable to reliably recover object representations in this way.

In the context of unsupervised instance segmentation, three very recent works propose to structure the generator of a GAN to directly learn to perform instance segmentation. The copy-paste GAN in Arandjelović and Zisserman [2019] uses a generator to generate a mask that interpolates between two images and learns to discover objects. Chen et al. [2019] decomposes the generative process in a segmentation step that separates foreground and background, and a generation step that in-paints the foreground segment. Finally, Bielski and Favaro [2019] present a layered generator that distinguishes between the foreground and background by perturbing the foreground image relative to the background. Our approach is different insofar we present a structured generator that facilitates both generation and segmentation of visual scenes that are composed of *multiple* objects and background, and which can be trained through the process of synthesizing images alone.

6.3 Experiments

We investigate different aspects of the proposed architectural structure on several multi-object datasets[3]. We are particularly interested in verifying that images are generated as compositions of objects and that the relational and background structure is properly utilized. Moreover, we study the degree to which the incorporated structure helps model these more complex image distributions, and how we can perform inference to recover which image regions correspond to objects for unseen images.

Data Sets We consider five multi-object datasets[4]. The first three are different variations of *Multi-MNIST (MM)*, in which each image consists of three MNIST

[3]Code is available online at `https://git.io/JePuK`.

[4]Datasets are available online at `https://goo.gl/Eub81x`.

digits [LeCun et al., 1998] that were rescaled and drawn randomly onto a 64×64 canvas. In *Independent MM*, digits are chosen randomly and there is no relation among them. The *Triplet* variation imposes that all digits in an image are of the same type, requiring relations among the digits to be considered during the generative process. Similarly in *RGB Occluded MM* each image consists of exactly one red, green, and blue digit. The fourth dataset (*CIFAR10 + MM*) is a variation of CIFAR10 [Krizhevsky et al., 2009] in which the digits from *RGB Occluded MM* are drawn onto a randomly chosen (resized) CIFAR10 image. Our final dataset is CLEVR [Johnson et al., 2017a], which we downsample to 160×240 followed by center-cropping to obtain 128×128 images. Samples from each dataset can be seen in Appendix B.1.3.

Evaluation A popular evaluation metric to evaluate GANs is the *Fréchet Inception Distance* (FID) [Heusel et al., 2017]. It computes the distance between two empirical distributions of images as the Fréchet distance between two corresponding multivariate Gaussian distributions that were fit to each dataset using the features of a pre-trained Inception network for each image. Although prior work found that FID correlates well with perceived human quality of images on standard image data sets [Lucic et al., 2018], we find that FID is less useful when considering images consisting of *multiple* salient objects. Our results in Section 6.3.2 suggest that FID is not a good indicator of performance during later stages of training, and may easily be fooled by a GAN that focuses on image statistics rather than content (e.g. generating the correct number of objects). We hypothesize that this inability is due to the Inception network having been trained only for single object classification [Szegedy et al., 2015].

Therefore, in addition to FID, we conduct two different studies among humans, 1) to compare images generated by our models to a baseline, and 2) to answer questions about the content of generated images. The latter allows us to verify whether generated images are probable samples from the true image distribution. As conducting a human evaluation of this kind is not feasible for large-scale hyperparameter search, we will continue to rely on FID to select the "best" models during hyperparameter selection. Details of these human studies can be found in Appendix A.3.4.

Set-up Each model is optimized with ADAM [Kingma and Ba, 2015] using a learning rate of 10^{-4}, and batch size 64 for 1M steps. We compute the FID (using 10K samples) every 20K steps and select the best set of parameters accordingly. On each data set, we compare GANs that incorporate our proposed structure to a

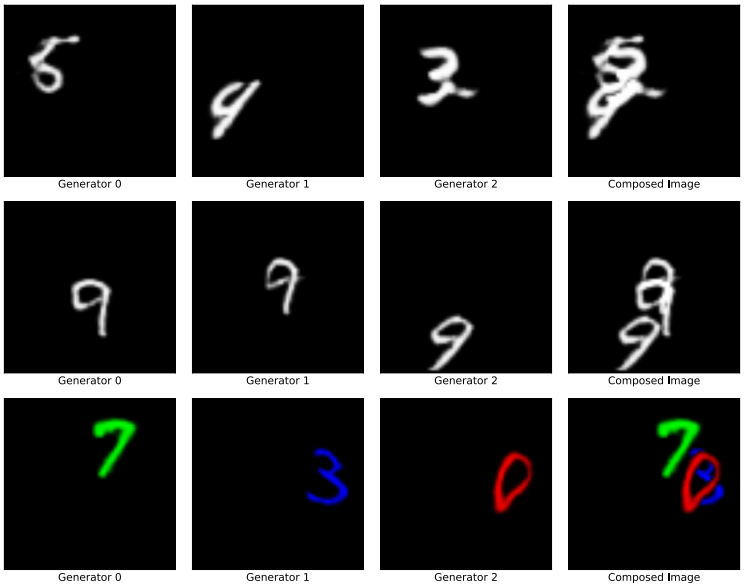


Figure 6.2. Generated images by *3-GAN* on Multi-MNIST: *Independent* (top), *Triplet* (middle), and *RGB Occluded* (bottom). The three columns on the left show the output of each object generator, and the right column the composed image.

strong baseline that does not. In both cases we conduct extensive grid searches, covering on the order of 40-50 hyperparameter configurations for each data set, using ranges that were previously found good for GANs [Lucic et al., 2018; Kurach et al., 2019]. Each configuration is run with 5 different seeds to be able to estimate its variance. A description of the hyperparameter search and samples of our best models can be seen in Appendix A.3.2 and B.1.3 respectively. For IODINE [Greff et al., 2019], we make use of the official trained model released by the authors.

Composing On the binary *Independent MM* and *Triplet MM* we sum the outputs of the object generators as in (6.2), followed by clipping to $(0, 1)$, since there is no need for compositing. On all other data sets, we use alpha compositing with a fixed order, i.e. using (6.7). In this case, the object generators output an additional alpha channel, except for *RGB Occluded MM* in which we obtain alpha values by thresholding the output of each object generator at 0.1 for simplicity since there is no background.

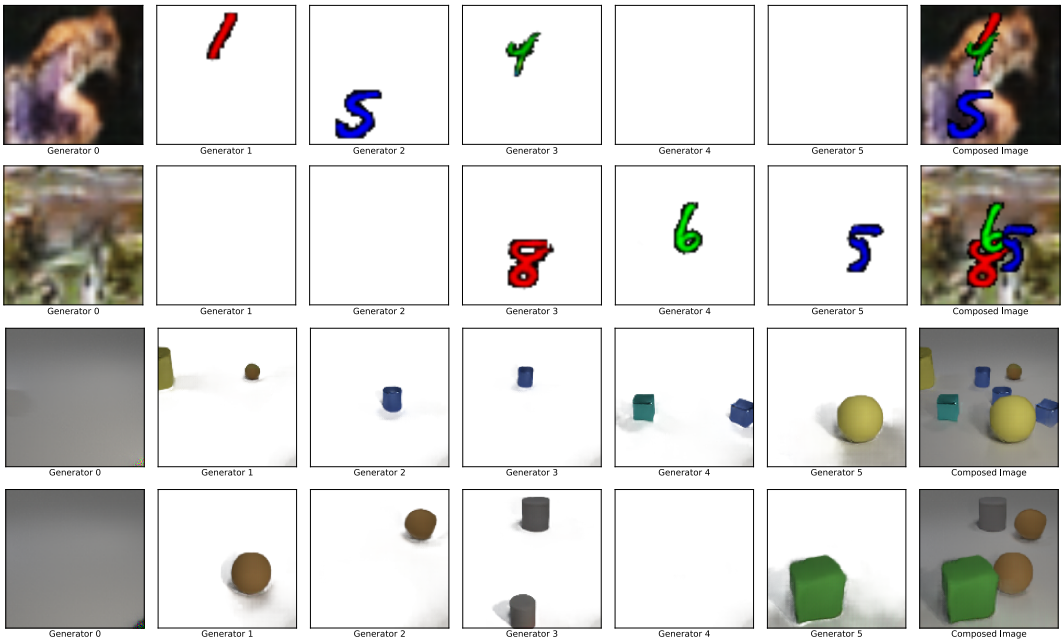


Figure 6.3. Generated samples by *5-GAN rel. bg.* on *CIFAR10 + MM* (top), and CLEVR (bottom). The left column corresponds to the output of the background generator. The next five columns are the outputs of each object generator, and the right column the composed image. Images are displayed as RGBA, with white denoting an alpha value of zero.

Notation In reporting our results we will break down the results obtained in terms of the structure that was incorporated in the generator. We will denote *k-GAN* to describe a generator consisting of $K = k$ components, *k-GAN rel.* if it incorporates relational structure and *k-GAN ind.* if it does not. Additionally, we will append "*bg.*" when the model includes a separate background generator. We will use *k-GAN* to refer more generally to GANs that incorporate any of the proposed structure, and *GAN* to refer to the collection of GANs with different hyperparameters in our baseline.

6.3.1 Qualitative Analysis

Utilizing Structure We begin by analyzing the output of each (object) generator for *k-GAN*. Among the best performing models in our search, we consistently find that the final image is generated as a composition of images consisting of individual objects and background. It can be seen that in the process of learning to generate images, *k-GAN* learns about what are individual objects, and what is

Figure 6.4. Generated images by *5-GAN rel. bg.* (top) and *GAN* (bottom), when traversing the latent space of a single (object) generator. For *k-GAN* only a single digit is transformed, while for *GAN* the entire scene changes.

background, without relying on prior knowledge or conditioning. This learned knowledge can then be extracted to learn unsupervised instance segmentation for the purpose of *segregation*, which we shall explore later. Examples generated by *k-GAN* for each data set can be seen in Figures 6.2 and 6.3.

On CLEVR, where images often have a greater number of objects than the number of components K that was used during training, we find that the generator continues to learn a factored solution (i.e. using visual primitives that consist of 1-3 objects)[5]. This is interesting, as it suggests that using compositionality is preferable even when the factorization is sub-optimal (compare also to our result for Space Invaders in Chapter 5). A similar tendency was found when analyzing generated images by *k-GAN ind.* when $k > 3$ on the Multi-MNIST data sets. The generator decodes some latents as "no digit" in attempting to generate the correct number of digits.

From the generated samples by *k-GAN rel.* we observe that relations among the objects are correctly captured in most cases and that the relational mechanism can be used to more reliably generate the correct number of digits when K is greater than the number of objects in the data set. We also observe that sometimes the background generator generates a *single* object together with the background. It rarely generates more than one object, which is further evidence that it is indeed more efficient to use the object generators.

[5]At the time of public release our results on $CLEVR$ were state of the art in the sense that there existed no other unconditional object-centric deep generative model that was capable of generating $CLEVR$ images (or more complicated multi-object images) at the level of quality as was presented in van Steenkiste et al. [2018b].

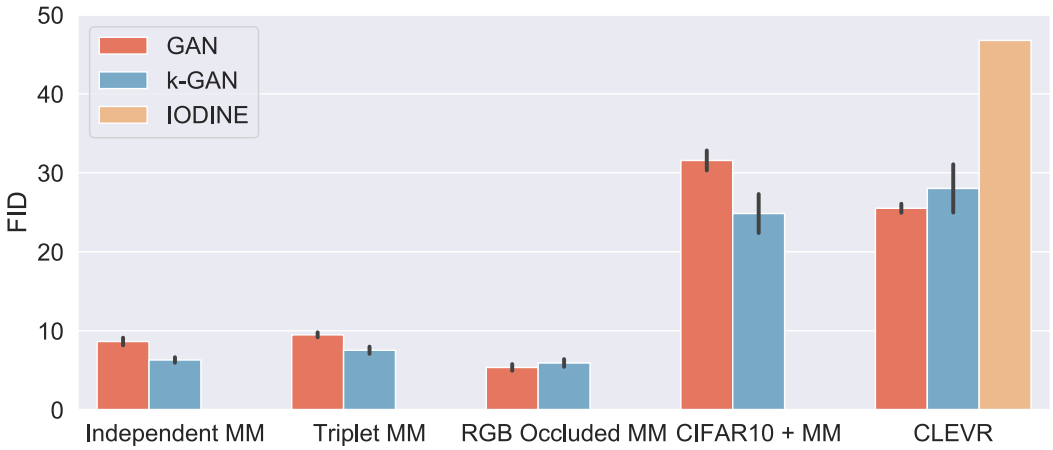


Figure 6.5. The best FID obtained by *GAN* and *k-GAN* on all data sets following our grid search. The best configurations were chosen based on the smallest average FID (across 5 seeds). Standard deviations across seeds are illustrated with error bars. For IODINE we made use of the official trained model for CLEVR released by the authors [Greff et al., 2019].

Latent Traversal We explore the degree to which the relational extension affects our initial independence assumption about objects. If it were to cause the latent representations to become fully dependent on one another then it could negate the benefits of compositionality. We conduct an experiment in which we traverse the latent space of a single latent vector in *k-GAN rel.* by adding a random vector to the original sample with fixed increments and generate images from the resulting latent vectors. An example can be seen in Figure 6.4 (top), where we find that traversing the latent space of a single component affects only the green digit, whereas the visual presentation of the others remains unaffected.

We observe this behavior (i.e. the relational mechanism does not unnecessarily interfere) for the majority of the generated samples, confirming to a large degree our own intuition of how the relational mechanism should be utilized. When traversing the latent space of *GAN*, for which information about different objects is entangled, it results in a completely different scene (see Figure 6.4 bottom). Hence, by decomposing the underlying representation it is more robust to common variations in image space.

(a) Independent MM *(b) Triplet MM*

Figure 6.6. Result of segmenting unseen test-images using a segmenter that was trained on images (and labels) generated by *3-GAN*. Note that this way of 'inverting' the learned generative model is only possible due to the added structure, wich makes the generative process interpretable and semantically understood.

6.3.2 Quantitative Analysis

Fréchet Inception Distance We train *k-GAN* and *GAN* on each data set, and compare the FID of the models with the lowest average FID across seeds (Figure 6.5). On all data sets but *CLEVR* we find that *k-GAN* compares better or similar to *GAN*, although typically by a small margin. Importantly, compared to the later proposed IODINE [Greff et al., 2019], which also incorporates architectural structure to facilitate object compositionality, we observe that k-GAN significantly outperforms.

An analysis, using different variations of *k-GAN*, leads to several interesting observations. On the relational data sets (Figure B.1) it can be observed that the relational extension is important to obtain good performance, while it does not harm performance when no relations are present. Likewise on the background data sets (Figures B.2 and B.3) we find that the background extension is important, although some mitigation is possible using the relational mechanism. Finally, we observe several small differences in FID when changing the number of object generators K. Surprisingly, we find that the lowest FID on *Independent MM* is obtained by *4-GAN* without relational structure, which by construction is unable to consistently generate 3 digits. It suggests that FID is unable to capture these properties of images (motivating our human study), and renders any subtle differences in terms of FID between *k-GAN* and *GAN* inconclusive.

Instance Segmentation We train a segmenter on data sampled from *3-GAN* by treating the output of each object generator as pixel-level segmentation labels for the generated image (a similar technique was explored in Spampinato et al. [2019] for motion segmentation in videos). Note that this labeled data is obtained

Data set	*Ground Truth*	*3-GAN*
Independent MM	0.890	0.886 ± 0.003
Triplet MM	0.911	0.903 ± 0.002
RGB Occluded MM	0.928	0.955 ± 0.003
CIFAR10 + MM	0.950	0.814 ± 0.131

Table 6.1. ARI scores obtained by training a segmenter on *ground truth* data, and on samples from *3-GAN*. Standard deviations are computed using generated data from the 5 best *3-GAN* models according to FID.

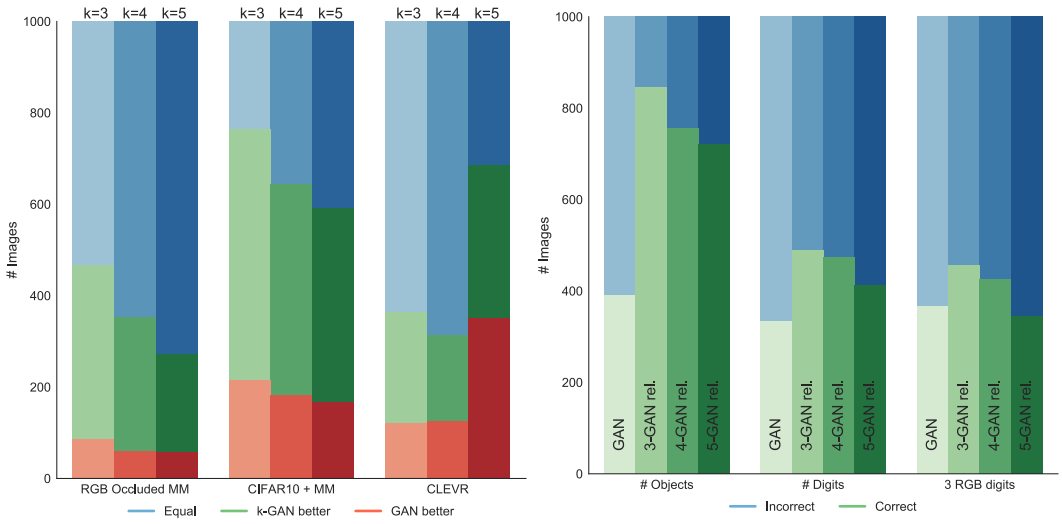
in a purely unsupervised fashion by exploiting the fact that, through incorporating structure, the learned generative process is now interpretable and semantically understood. We test how the segmenter generalizes to real images (Figure 6.6) for which we have ground-truth segmentations available and measure its accuracy using the Adjusted Rand Index (ARI [Hubert and Arabie, 1985]) score. As a comparison, we also train a segmenter in a purely supervised fashion using ground-truth segmentations (additional details are available in Appendix A.3.3).

In Table 6.1 we find that using unsupervised data from *3-GAN* is often as good as using ground-truth (and can even have a positive regularization effect). These results strengthen our initial findings that *k-GAN* reliably generates images as compositions of objects. Moreover, it presents a novel approach to unsupervised instance segmentation that does not rely on an "encoder", but rather leverages the structure of the learned generative process to perform inference.

Human Evaluation We asked humans to compare the images generated by *k-GAN rel.* to our baseline on *RGB Occluded MM*, *CIFAR10 + MM* and *CLEVR*, using the configuration with a background generator for the last two data sets[6]. For each model, we selected the 10 best hyperparameter configurations (lowest FID), from which we each generated 100 images. We asked up to three raters for each image and report the majority vote or "Equal" if no decision was reached.

Figure 6.7a reports the results when asking human raters to compare the visual quality of the generated images by *k-GAN* to those by *GAN*. It can be seen that *k-GAN* compares favorably across all data sets and in particular on *RGB Occluded MM* and *CIFAR10 + MM* we observe large differences. We find that *k-GAN* performs better even when $k > 3$, which can be attributed to the relational

[6]On *CLEVR* we instructed the raters to ignore visual implausibilities due to floating objects (for both *k-GAN* and *GAN*) that may arise due to the fixed order in (6.7), and measured this effect separately in Figure B.5.



(a) Comparing image quality

(b) Properties on RGB Occluded MM

Figure 6.7. Results of human evaluation a) comparing the quality of the generated images by *k-GAN* (k = 3,4,5) to *GAN* b) Properties of generated images by *k-GAN* (k = 3,4,5) and *GAN* on *RGB Occluded MM*. It can be seen that *k-GAN* generates better images that are more faithful to the reference distribution.

mechanism, allowing all components to agree on the correct number of digits.

In a second study, we asked humans to report specific properties of the generated images, a complete list of which can be found in Appendix A.3.4. Here our goal was to assess if the generated images by *k-GAN* are more faithful to the reference distribution compared to *GAN*, which is particularly important in the context of representation learning. The results on *RGB Occluded MM* are summarized in Figure 6.7b. It can be seen that *k-GAN* more frequently generates images that have the correct number of objects, number of digits, and that satisfy all properties simultaneously. The differences in the response to the correct number of digits and the correct number of objects suggest that the generated objects are often not recognizable as digits. This does not appear to be the case from the generated samples in Appendix B.1, suggesting that the raters may not have been familiar enough with the variety of MNIST digits.

On *CIFAR10 + MM* (Figure B.4), it appears that *GAN* is able to accurately generate the correct number of objects, although the addition of background makes it more difficult to interpret these results. Indeed, on the number of digits, *k-GAN* outperforms *GAN* by the same margin as one would expect compared to the results in Figure 6.7b.

Finally, in comparing the generated images by *k-GAN* and *GAN* on *CLEVR*, we noticed that the former generated more crowded scenes (containing multiple large objects in the center), and more frequently generated objects with distorted shapes or mixed colors. On the other hand, we found cases in which *k-GAN* generated scenes containing 'flying' objects, a by-product of the fixed order in which we apply (6.7). We asked humans to score images based on these properties, which appears to confirm these observations (see Figure B.5).

6.4 Discussion

In this chapter, we have investigated an alternative approach to learning about objects based on more powerful implicit generative models. We found that by incorporating architectural modifications to the generator of a GAN, it is able to learn to generate images as a composition of individual objects and background. It was also shown how it can resolve dependencies between objects at a representational level in this way.

The key motivation for using an approach based on GANs was that it can more easily be applied to more complex multi-object image distributions that were previously unattainable. Through extensive experiments, we empirically validated that this is the case, and our approach was the first of its kind that was able to scale to the challenging CLEVR data set [Johnson et al., 2017a]. Compared to a strong baseline of GANs, we were able to demonstrate how the proposed structure helps in learning a generative process that is more faithful to the observed reference distribution. On CLEVR, it was shown how our GAN-based approach is able to improve over the later proposed IODINE [Greff et al., 2019] in terms of generative capabilities.

For the purpose of segregation, it is important that the learned generative process can be 'inverted' to learn about objects. In the case of GANs, this requires incorporating suitable architectural structure in the generator that distinguishes objects at a representational level, and more generally, that the learned generative process utilizes this structure as intended. Empirically it was shown how our approach addresses both, which allowed us to formulate an inference procedure that leveraged the output of the object (and background) generator(s) to learn to perform instance segmentation. We demonstrated how the resulting segmentation model was able to generalize to unseen images for the purpose of *segregation*.

One area of improvement is in being able to correctly order objects according to their depth when combining the outputs of the object generators using alpha compositing. On CLEVR we observed cases in which objects appear to be flying,

which is the result of being unable to route the information content of a "foreground" object to the corresponding "foreground" generator as induced by the fixed order in which images are composed. Although in principle the relational mechanism may account for this distinction, in practice we found that this is not sufficient.

Another interesting avenue for improvement is to also incorporate structure in the discriminator, which implements the loss function for the generator. Similar to how we found that the usefulness of the pre-trained Inception embedding is limited for reasoning about the validity of multi-object images, the discriminator may also experience difficulties in accurately judging images from being real or fake without an appropriate inductive bias. Ideally, we would have the discriminator reason about the plausibility of the appearance of each object individually, as well as the image as a whole. Adding additional 'patch discriminators' [Isola et al., 2017], where patches correspond to objects, may serve a starting point for pursuing this direction.

Chapter 7

Evaluating Disentangled Representations

In this chapter we are concerned with *disentangled* representations (Section 3.5), which encode information about salient (or explanatory) factors of variation in the data using only a select few dimensions for each factor. We set out to conduct a large-scale study to evaluate the merits of this particular representational format as a way of encoding information about objects for the purpose of learning down-stream (abstract) visual reasoning tasks.

Following the success of learning distributed representations that efficiently encode the content of high-dimensional sensory data [Vincent et al., 2008; Kingma and Welling, 2014; Lotter et al., 2017], there has been an increasing interest in pursuing representations that disentangle informative factors of variation. In a disentangled representation, information about an individual factor value can be readily accessed and is robust to changes in the input that do not affect this factor. Hence, learning to solve a down-stream task from a disentangled representation is expected to require fewer samples and be easier in general [Schmidhuber et al., 1996; Bengio et al., 2013; Higgins et al., 2017b; Peters et al., 2017; Higgins et al., 2018b].

Several of these purported benefits can be traced back to empirical evidence presented in the recent literature. Learning from disentangled representations was found to be more sample-efficient [Higgins et al., 2018b], less sensitive to nuisance variables [Lopez et al., 2018], and yield better generalization performance [Higgins et al., 2017b; Hsu et al., 2017; Achille et al., 2018; Eastwood and Williams, 2018; Steenbrugge et al., 2018]. However, in other cases, it was less

This chapter is based on van Steenkiste et al. [2019], which was published as a conference paper at *NeurIPS 2019*.

clear whether the observed benefits were actually due to disentanglement [Kumar et al., 2018; Locatello et al., 2018]. Indeed, while these results are generally encouraging, a systematic evaluation on a complex down-stream task of a wide variety of disentangled representations obtained by training different models, using different hyperparameters, and data sets, is lacking.

The primary contribution of this chapter is a large-scale evaluation[1] of disentangled representations to systematically evaluate some of these purported benefits. Rather than focusing on a simple single factor classification task, we evaluate their usefulness on abstract visual reasoning tasks that challenge the current capabilities of state-of-the-art deep neural networks [Santoro et al., 2018b]. On these tasks, we are able to observe compelling evidence that more disentangled representations yield better sample-efficiency. This offers a more positive outlook for learning disentangled representations compared to a prior study, which did not find evidence of increased sample efficiency on a much simpler down-stream task [Locatello et al., 2018].

7.1 Methodology

The approach taken in our study is as follows. First, we create two new abstract visual reasoning tasks similar to Raven's Progressive Matrices [Raven, 1941] based on two disentanglement data sets: *dSprites* [Higgins et al., 2017a], and *3dshapes* [Kim and Mnih, 2018]. A key design property of these tasks is that they are hard to solve based on statistical co-occurrences and require reasoning about the relations between different objects. Second, we train a large number of unsupervised disentanglement models (spanning four different approaches from the literature) on the individual images of these two data sets and extract their representations. Finally, we train several reasoning models that use these disentangled representations to perform abstract reasoning and measure their accuracy at various stages of training. This then allows us to evaluate the usefulness of disentangled representations, namely by comparing the accuracy of these abstract reasoning models to the degree of disentanglement of the learned representations (measured according to five different disentanglement metrics).

In the remainder of this section, we will first describe the metrics used for evaluating disentanglement and the methods for learning disentangled representations (Section 7.1.1). Next, we introduce the abstract visual reasoning tasks that we will consider and the reasoning models that we will train to perform these tasks (Section 7.1.2).

[1] Reproducing these experiments requires approximately 2.73 GPU years (NVIDIA P100).

7.1.1 Disentanglement

Metrics for Measuring Disentanglement

Multiple metrics have been proposed that leverage the ground-truth generative factors of variation in the data to measure disentanglement in learned representations[2]. In a prior study [Locatello et al., 2018] several of these metrics were considered, which we will adopt for our purposes in this work: the *BetaVAE* score [Higgins et al., 2017a], the *FactorVAE* score [Kim and Mnih, 2018], the *Mutual Information Gap (MIG)* [Chen et al., 2018], the disentanglement score from Eastwood and Williams [2018] referred to as the *DCI Disentanglement* score, and the *Separated Attribute Predictability (SAP)* score [Kumar et al., 2018].

The *BetaVAE* score, *FactorVAE* score, and *DCI Disentanglement* score focus primarily on *modularity* (see Section 3.5.2 for a description of modularity, compactness, and explicitness). The former assess this property through *interventions*, i.e. by keeping one factor fixed and varying all others, while the *DCI Disentanglement* score estimates this property from the relative importance assigned to each feature by a random forest regressor in predicting the factor values. The *SAP* score and *MIG* are mostly focused on *compactness*. The *SAP* score reports the difference between the top two most predictive latent codes of a given factor, while *MIG* reports the difference between the top two latent variables with the highest mutual information to a certain factor.

The degree of *explicitness* captured by any of the disentanglement metrics remains unclear. In prior work, it was found that there is a positive correlation between the value of disentanglement metrics and down-stream performance on single factor classification [Locatello et al., 2018]. However, it is not obvious whether disentangled representations are useful for down-stream performance per se, or if the correlation is driven by the explicitness captured in the scores. In particular, the *DCI Disentanglement* score and the *SAP* score compute disentanglement by training a classifier on the representation. The former uses a random forest regressor to determine the relative importance of each feature, and the latter considers the gap in prediction accuracy of a support vector machine trained on each feature in the representation. *MIG* is based on the matrix of pairwise mutual information between factors of variations and dimensions of the representation, which also relates to the explicitness of the representation. On

[2] Current disentanglement metrics each require access to the ground-truth factors of variation, which may hinder the practical feasibility of learning disentangled representations. On the other hand, our goal is to assess the usefulness of disentangled representations more generally (i.e. regardless of whether they can be obtained or not without making additional assumptions), which can be verified independently.

the other hand, the *BetaVAE* and *FactorVAE* scores predict the index of a fixed factor of variation and not the exact value.

Methods for Learning Disentangled Representations

Several methods have been proposed to learn disentangled representations. Here we are interested in evaluating the benefits of disentangled representations that have been learned through *unsupervised* learning. In order to control for potential confounding factors that may arise when using a single model, we use the representations learned from four state-of-the-art approaches from the literature: *β-VAE* [Higgins et al., 2017a], *FactorVAE* [Kim and Mnih, 2018], *β-TCVAE* [Chen et al., 2018], and *DIP-VAE* [Kumar et al., 2018].

Using notation from Tschannen et al. [2018], we can view all of these models as autoencoders that are trained with a regularized variational objective (compare with (2.18) in Section 2.2.1, but which is here written as a *loss* function) of the form:

$$\mathbb{E}_{p(\boldsymbol{x})}[\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[-\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})]] + \lambda_1 \mathbb{E}_{p(\boldsymbol{x})}[R_1(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}))] + \lambda_2 R_2(q_\phi(\boldsymbol{z})). \quad (7.1)$$

The output of the neural network *encoder* that parametrizes $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ yields the representation. Regularization serves to control the information flow through the bottleneck induced by the encoder, while different regularizers primarily vary in the notion of disentanglement that they induce. *β-VAE* restricts the capacity of the information bottleneck by penalizing the KL-divergence, using $\beta = \lambda_1 > 1$ with $R_1(q_\phi(\boldsymbol{z} \mid \boldsymbol{x})) := D_{KL}[q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,||\, p(\boldsymbol{z})]$, and $\lambda_2 = 0$; *FactorVAE* penalizes the Total Correlation (TC [Watanabe, 1960]) of the latent variables via adversarial training, using $\lambda_1 = 0$ and $\lambda_2 = 1$ with $R_2(q_\phi(\boldsymbol{z})) := TC(q_\phi(\boldsymbol{z}))$; *β-TCVAE* also penalizes the Total Correlation but estimates its value via a biased Monte Carlo estimator; and finally *DIP-VAE* penalizes a mismatch in moments between the aggregated posterior and a factorized prior, using $\lambda_1 = 0$ and $\lambda_2 \geq 1$ with $R_2(q_\phi(\boldsymbol{z})) := || \operatorname{Cov}_{q_\phi(\boldsymbol{z})} - I ||_F^2$.

7.1.2 Abstract Visual Reasoning

Datasets for Abstract Visual Reasoning

We evaluate the benefits of disentangled representations on abstract visual reasoning tasks. Abstract reasoning tasks require a learner to infer abstract relationships between multiple entities (e.g. objects in images) and re-apply this knowledge in

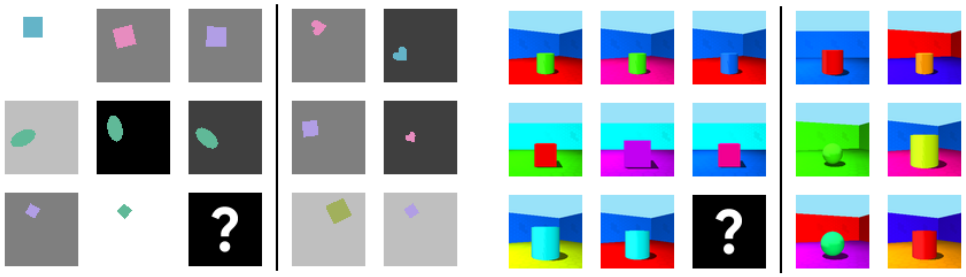


Figure 7.1. Examples of RPM-like abstract visual reasoning tasks based on *dSprites* (left) and *3dshapes* (right). Focusing on the right example, note that the correct answer cannot be determined by solely considering the incomplete context sequence and the answer panels. In particular, we can not tell whether 1, 2, or 3 relationships hold, and if for example the wall color or the object color is constant. As a result, one must consider the other two rows of context panels to deduce that it is the background color, the azimuth, and the shape-type, that are equal among the panels in a sequence. Then, this insight can be applied to the bottom row to observe that a cylinder, a specific viewpoint, and a lighter blue background are required in the correct solution. Based on this, the correct answer panel fulfilling these criteria can be selected (middle right).

newly encountered settings [Kemp and Tenenbaum, 2008]. Humans are known to excel at this task, as is evident from experiments with simple visual IQ tests such as *Raven's Progressive Matrices* (RPMs) [Raven, 1941]. An RPM consists of several context panels organized in multiple sequences, with one sequence being incomplete. The task consists of completing the final sequence by choosing from a given set of answer panels. Choosing the correct answer panel requires one to infer the relationships between the panels in the complete context sequences, and apply this knowledge to the remaining partial sequence.

In recent work [Santoro et al., 2018b], the abstract reasoning capabilities of deep neural networks were evaluated on this task. Using a data set of RPM-like matrices it was found that standard deep neural network architectures struggle at abstract visual reasoning under different training and generalization regimes. Their results indicate that it is difficult to solve these tasks by relying purely on superficial image statistics, and suggest that they can only be solved efficiently through some form of relational reasoning. This makes this setting particularly appealing for investigating the benefits of disentangled representations.

Generating RPM-like Matrices Rather than evaluating disentangled representations using the Procedurally Generated Matrices (PGM) dataset [Santoro et al., 2018b] we construct two new abstract RPM-like visual reasoning datasets based on two existing datasets for disentangled representation learning. Our motivation for this is twofold: it is not clear what a ground-truth disentangled representation should look like for the PGM dataset, while the two existing disentanglement data sets include the ground-truth factors of variation. Secondly, in using established data sets for disentanglement, we can reuse hyperparameter ranges that have proven successful. We note that the scope of our study is much broader (e.g. in terms of the considered metrics, methods, and hyperparameters) compared to prior work [Steenbrugge et al., 2018] that evaluates the representation of a single trained β-VAE [Higgins et al., 2017a] on the original PGM data set.

To construct the abstract reasoning tasks, we use the ground-truth generative model of the *dSprites* [Higgins et al., 2017a] and *3dshapes* [Kim and Mnih, 2018] data sets with the following changes:[3] For *dSprites*, we ignore the orientation feature for the abstract reasoning tasks as certain objects such as squares and ellipses exhibit rotational symmetries. To compensate, we add background color (5 different shades of gray linearly spaced between white and black) and object color (6 different colors linearly spaced in HUSL hue space) as two new factors of variation. Similarly, for the abstract reasoning tasks (but not when learning representations), we only consider three different values for the scale of the object (instead of 6) and only four values for the x and y position (instead of 32). For *3dshapes*, we retain all of the original factors but only consider four different values for scale and azimuth (out of 8 and 16) for the abstract reasoning tasks. We refer to Appendix A.4.2 for samples from these data sets.

For the modified *dSprites* and *3dshapes*, we now create corresponding abstract reasoning tasks. The key idea is that one is given a 3×3 matrix of context image panels with the bottom right image panel missing, as well as a set of six potential answer panels (see Figure 7.1 for an example). One then has to infer which of the answers fits in the missing panel of the 3×3 matrix based on relations between image panels in the rows of the 3×3 matrices. Due to the categorical nature of ground-truth factors in the underlying data sets, we focus on the *AND* relationship in which one or more factor values are equal across a sequence of context panels [Santoro et al., 2018b].

We generate instances of the abstract reasoning tasks in the following way: First, we uniformly sample whether 1, 2, or 3 ground-truth factors are fixed across

[3] These were implemented to ensure that humans can visually distinguish between the different values of each factor of variation.

rows in the instance to be generated. Second, we uniformly sample without replacement the set of factors in the underlying generative model that should be kept constant. Third, we uniformly sample a factor value from the ground-truth model for each of the three rows and for each of the fixed factors[4]. Fourth, for all other ground-truth factors we also sample 3×3 matrices of factor values from the ground-truth model with the single constraint that the factor values are not allowed to be constant across the first two rows (in that case we sample a new set of values). After this, we have obtained ground-truth factor values for each of the 9 panels in the correct solution to the abstract reasoning task, and we can sample corresponding images from the ground-truth model. To generate difficult alternative answers, we take the factor values of the correct answer panel and randomly resample the non-fixed factors as well as a random fixed factor until the factor values no longer satisfy the relations in the original abstract reasoning task. We repeat this process to obtain five incorrect answers and finally insert the correct answer in a random position. Examples of the resulting abstract reasoning tasks can be seen in Figure 7.1 as well as in Figure A.1 and Figure A.2.

Abstract Visual Reasoning Model

We will make use of the *Wild Relation Network (WReN)* to solve the abstract visual reasoning tasks [Santoro et al., 2018b]. It incorporates a relational inductive bias and was introduced in prior work specifically for such tasks. The *WReN* is evaluated for each answer panel $a \in A = \{a_1, ..., a_6\}$ in relation to all the context-panels $C = \{c_1, ..., c_8\}$ as follows: First an embedding is computed for each panel using a deep Convolutional Neural Network (CNN):

$$E = \{\text{CNN}(c_1), ..., \text{CNN}(c_8)\} \cup \{\text{CNN}(a)\}, \tag{7.2}$$

which then serve as input to a Relation Network (RN) module [Santoro et al., 2017]:

$$\text{WReN}(a, C) = f_\phi(\sum_{e_1, e_2 \in E} g_\theta(e_1, e_2)), \tag{7.3}$$

where f_ϕ and g_θ are neural networks, each with their respective parameters. The Relation Network reasons about the different relationships between the context and answer panels and outputs a score. The answer panel $a \in A$ with the highest score (after applying softmax normalization) is chosen as the final output. The entire system is trained in a supervised fashion using the standard cross-entropy loss.

[4]Note that different rows may have different values.

The Relation Network implements a suitable inductive bias for (relational) reasoning [Battaglia et al., 2018]. It separates the reasoning process into two stages. First g_θ is applied to all pairs of panel embeddings to consider relations between the answer panel and each of the context panels, and relations among the context panels. Weight-sharing of g_θ between the panel-embedding pairs makes it difficult to overfit to the image statistics of the individual panels. Finally, f_ϕ produces a score for the given answer panel in relation to the context panels by globally considering the different relations between the panels as a whole. By using the same *WReN* for different answer panels it is ensured that each answer panel undergoes the same reasoning process.

Notice how the relational inductive bias is here implemented at the level of visual scenes as opposed to the level of individual objects as was encountered in Chapters 5 and 6. This is not problematic since the comparisons that need to be performed in this case are between the features of different *scenes* (referring to the *same* object or background), which are described in a common format. In that sense, the representation of each visual scene for this task can be viewed as a single 'object representation'. Therefore, we expect any findings regarding the utility of this particular representational format for abstract visual reasoning about the content of different scenes, to equally apply to reasoning about *individual objects* in a shared context (i.e. as encountered in Chapters 4 to 6), provided that their representations are similarly structured (i.e. disentangled).

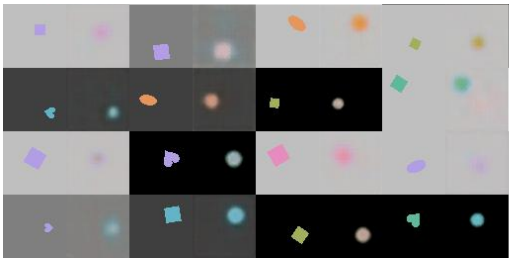
7.2 Results

7.2.1 Learning Disentangled Representations

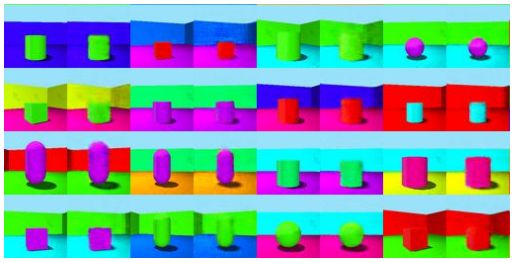
We train *β-VAE* [Higgins et al., 2017a], *FactorVAE* [Kim and Mnih, 2018], *β-TCVAE* [Chen et al., 2018], and *DIP-VAE* [Kumar et al., 2018] on the panels from the modified *dSprites* and *3dshapes* data sets[5]. For *β-VAE* we consider two variations: the standard version using a fixed β, and a version trained with the *controlled capacity increase* presented in Burgess et al. [2017]. Similarly for *DIP-VAE* we consider both the *DIP-VAE-I* and *DIP-VAE-II* variations of the proposed regularizer [Kumar et al., 2018]. For each of these methods, we considered six different values for their (main) hyperparameter and five different random seeds. The remaining experimental details are presented in Appendix A.4.

After training, we end up with 360 encoders, whose outputs are expected to cover a wide variety of different representational formats with which to encode

[5]Code is made available as part of *disentanglement_lib* at `https://git.io/JelEv`.



(a) FactorVAE trained on *dSprites*.



(b) DIP-VAE-I trained on *3dshapes*.

Figure 7.2. Reconstructions for different data sets and models (representative samples of median reconstruction error). Odd columns show real samples and even columns their reconstruction. *3dshapes* appears to be easier than *dSprites* where disentangling the shape was found to be more difficult.

information about the images. Figures B.23 and B.24 in Appendix B.2 show histograms of the reconstruction errors obtained after training, and the scores that various disentanglement metrics assigned to the corresponding representations. The reconstructions are mostly good (see also Figure 7.2), which confirms that the learned representations tend to accurately capture the image content. Correspondingly, we expect any observed difference in down-stream performance when using these representations to be primarily the result of *how* information is encoded. In terms of the scores of the various disentanglement metrics, we observe a wide range of values. It suggests that in going by different definitions of disentanglement, there are large differences among the quality of the learned representations.

7.2.2 Abstract Visual Reasoning

We train different WReN models where we control for two potential confounding factors: the representation produced by a specific model used to embed the input images, as well as the hyperparameters of the WReN model. For hyperparameters, we use a random search space as specified in Appendix A.4. We used the following training protocol: We train each of these models using a batch size of 32 for 100K iterations where each mini-batch consists of *newly generated* random instances of the abstract reasoning tasks. Similarly, every 1000 iterations, we evaluate the accuracy on 100 mini-batches of fresh samples. We note that this corresponds to the statistical optimization setting, sidestepping the need to investigate the

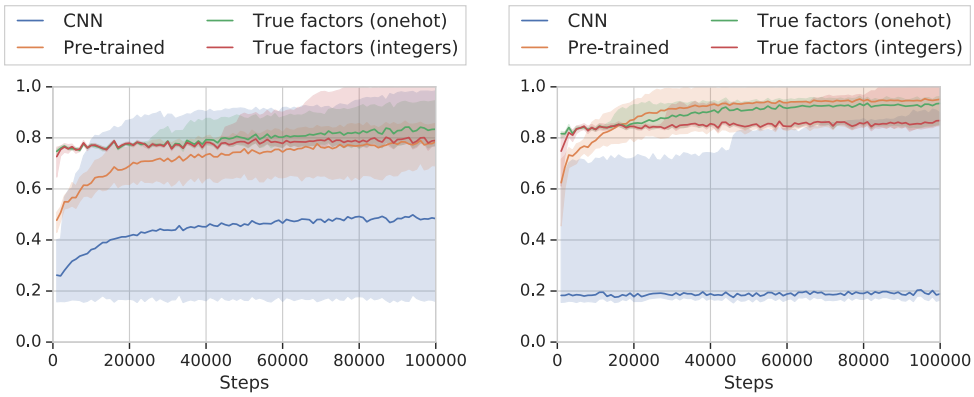


Figure 7.3. Average down-stream accuracy of baselines, and models using pre-trained representations on *dSprites* (left) and *3dshapes* (right). Shaded area indicates min and max accuracy.

impact of empirical risk minimization and overfitting[6].

Initial Study

First, we trained a set of baseline models to assess the overall complexity of the abstract reasoning task. We considered three types of representations: (i) CNN representations which are learned from scratch (with the same architecture as in the disentanglement models) yielding standard WReN, (ii) pre-trained frozen representations based on a random selection of the pre-trained disentanglement models, and (iii) directly using the ground-truth factors of variation (both one-hot encoded and integer encoded). We trained 30 different models for each of these approaches and data sets with different random seeds and different draws from the search space over hyperparameter values.

An overview of the training behavior and the accuracies achieved can be seen in Figure 7.3. We observe that the standard WReN model struggles to obtain good results on average, even after having seen many different samples at 100K steps. This is because training from scratch is hard and runs may get stuck in local minima where they predict each of the answers with equal probabilities. Given the pre-training and the exposure to additional unsupervised samples, it is not surprising that the learned representations from the disentanglement models perform better. The WReN models that are given the true factors also perform

[6]Note that the state space of the data generating distribution is very large: 10^6 factor combinations per panel and 14 panels for each instance yield more than 10^{144} potential instances (minus invalid configurations).

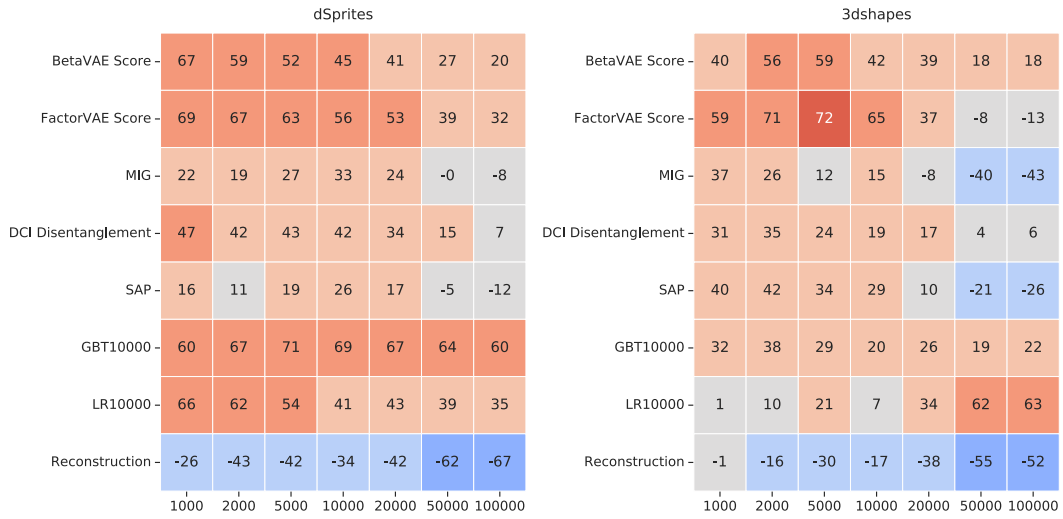


Figure 7.4. Rank correlation between various metrics and down-stream accuracy of the abstract visual reasoning models throughout training (i.e. for different number of samples).

well, already after only a few steps of training. We also observe that different runs exhibit a significant spread, which motivates why we analyze the average accuracy across many runs in the next section.

It appears that *dSprites* is the harder task, with models reaching an average score of 80%, while reaching an average of 90% on *3dshapes*. Finally, we note that most learning progress takes place in the first 20K steps, and thus expect the benefits of disentangled representations to be most clear in this regime.

Full Study

Based on the results from the initial study, we train a full set of WReN models in the following manner: We first sample a set of 10 hyperparameter configurations from our search space and then train WReN models using these configurations for each of the 360 representations from the disentanglement models (in this case replacing (7.2) with the pre-trained representation). We then compare the average down-stream validation accuracy of WReN with the *BetaVAE* score, the *FactorVAE* score, *MIG*, the *DCI Disentanglement* score, and the *Reconstruction* error obtained by the decoder on the unsupervised learning task. Out of curiosity, we also compare with the accuracy of a *Gradient Boosted Tree (GBT10000)* ensemble and a *Logistic Regressor (LR10000)* on single factor classification (averaged across factors) as measured on 10K samples.

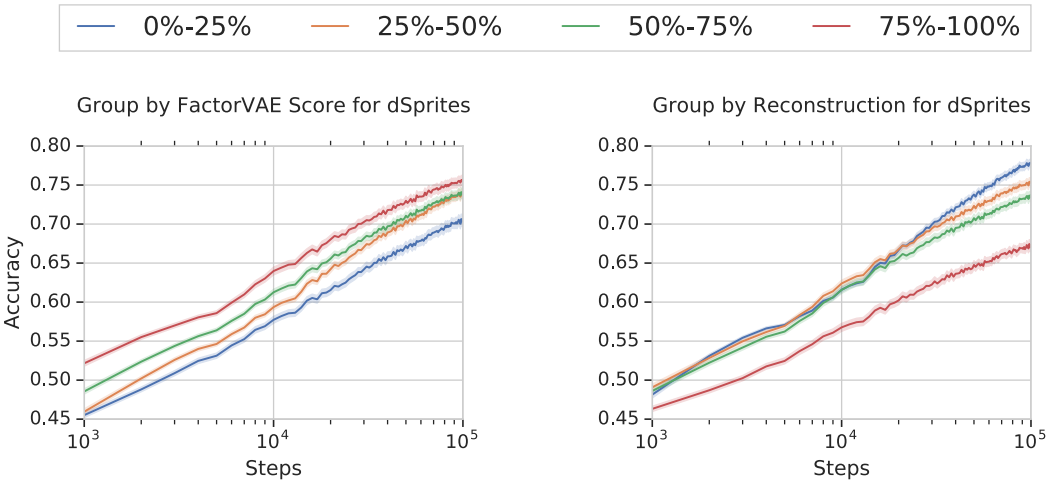


Figure 7.5. Down-stream accuracy of the abstract visual reasoning models throughout training, binned in quartiles based on the values assigned by the *FactorVAE* score (left), and *Reconstruction* error (right).

Differences in Disentanglement Metrics Figure 7.4 displays the rank correlation (Spearman) between these metrics and the down-stream classification accuracy, evaluated after training for 1K, 2K, 5K, 10K, 20K, 50K, and 100K steps. If we focus on the disentanglement metrics, several interesting observations can be made. In the few-sample regime (up to 20K steps) and across both data sets it can be seen that both the *BetaVAE* score and the *FactorVAE* score are highly correlated with down-stream accuracy. The *DCI Disentanglement* score is correlated less, while the *MIG* and *SAP* score exhibit even weaker correlation.

These differences between the different disentanglement metrics are perhaps not surprising, as they are also reflected in their overall correlation (see Figure B.22). Note that the *BetaVAE* score and the *FactorVAE* score directly measure the effect of intervention, i.e. what happens to the representation if all factors but one are varied, which is expected to be beneficial in efficiently comparing the content of two representations as required for this task. Similarly, it may be that the *MIG* and *SAP* score have a more difficult time in differentiating representations that are only partially disentangled, due to only comparing the top two latent codes. Finally, we note that the best performing metrics on this task are mostly measuring *modularity*, as opposed to *compactness*. A more detailed overview of the correlation between the various metrics and down-stream accuracy can be seen in Figures B.25 and B.26.

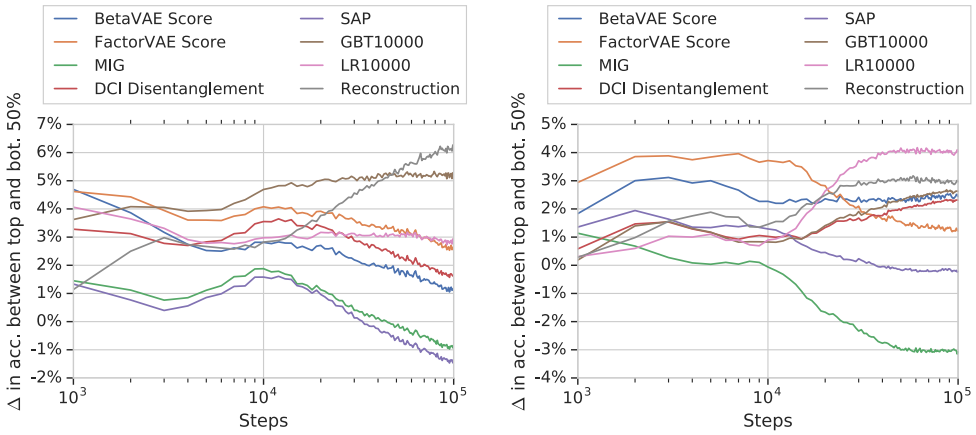


Figure 7.6. Difference in down-stream accuracy between top 50% and bottom 50% of the abstract visual reasoning models throughout training, according to various metrics on *dSprites* (left) and *3dshapes* (right). Note the different scale of the y-axis in both plots.

Disentangled Representations in the Few-Sample Regime If we compare the correlation of the disentanglement metric with the highest correlation (*FactorVAE*) to that of the *Reconstruction* error in the few-sample regime, then we find that disentanglement correlates much better with down-stream accuracy. Indeed, while low *Reconstruction* error indicates that all information is available in the representation (to reconstruct the image) it makes no assumptions about *how* this information is encoded. We observe strong evidence that disentangled representations (according to this metric) yields better down-stream accuracy using relatively few samples, and we therefore conclude that they are indeed more sample efficient compared to entangled representations in this regard.

Figure 7.5 demonstrates the down-stream accuracy of the WReNs throughout training, binned into quartiles according to their degree of being disentangled as measured by the *FactorVAE* score (left), and in terms of *Reconstruction* error (right). It can be seen that representations that are more disentangled give rise to better relative performance consistently throughout all phases of training. If we group models according to their *Reconstruction* error then we find that this (reversed) ordering is much less pronounced. An overview for all other metrics can be seen in Figures B.27 and B.28.

Disentangled Representations in the Many-Sample Regime In the many-sample regime (i.e. when training for 100K steps on batches of randomly drawn instances in Figure 7.4) we find that there is no longer a strong correlation between the

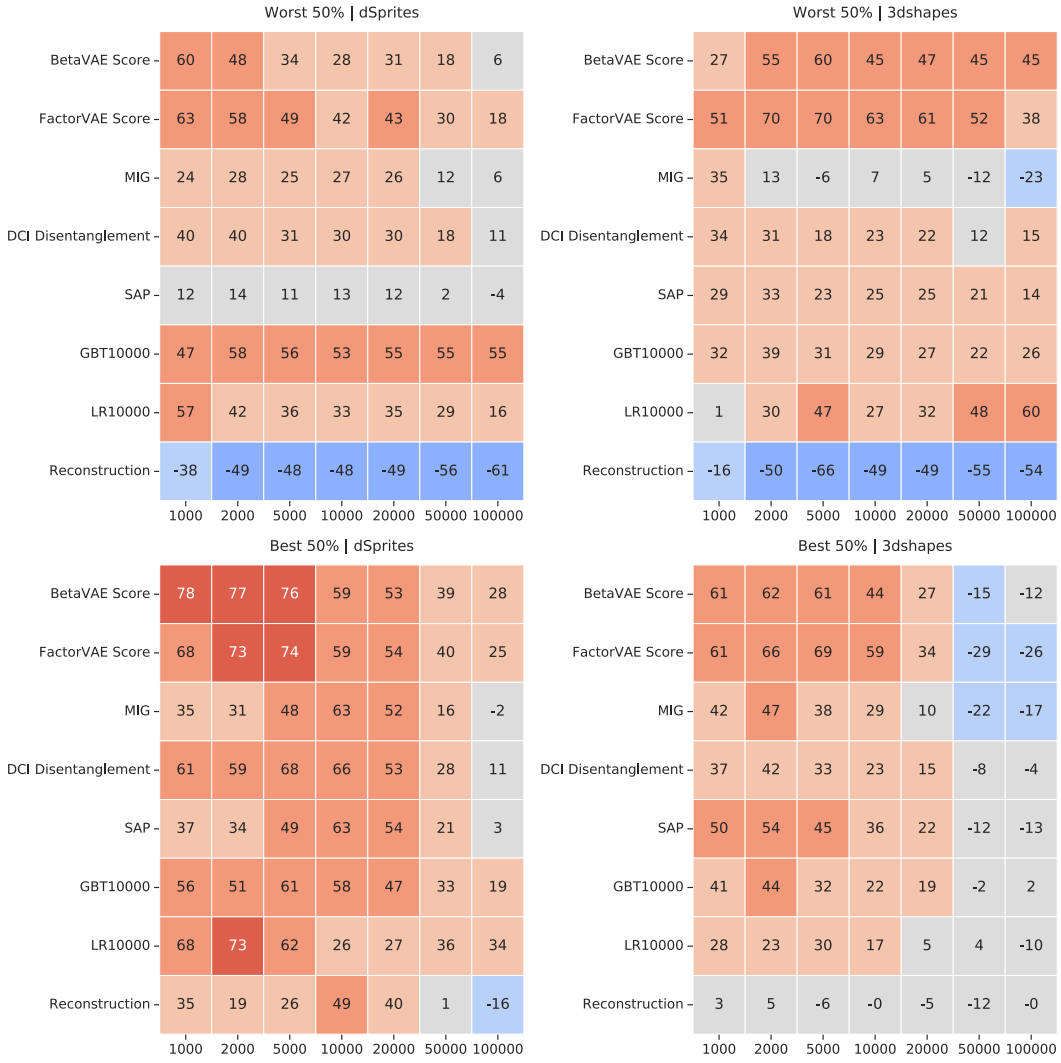


Figure 7.7. Rank correlation between various metrics and down-stream accuracy of the abstract visual reasoning models throughout training (i.e. for different numbers of samples). The results in the top row are based on the worst 50% of the models (according to final accuracy), and those in the bottom row based on the best 50% of the models. Columns correspond to different data sets.

scores assigned by the various disentanglement metrics and down-stream performance. This is perhaps not surprising as neural networks are general function approximators that, given access to enough labeled samples, are expected to overcome potential difficulties in using entangled representations. The observation that *Reconstruction* error correlates much more strongly with down-stream

accuracy in this regime further confirms that this is the case.

A similar observation can be made if we look at the difference in down-stream accuracy between the top and bottom half of the models according to each metric in Figure 7.6. For most disentanglement metrics, larger positive differences are observed in the few-sample regime that gradually reduce as more samples are provided. Meanwhile, the gap gradually increases for *Reconstruction* error upon seeing additional samples.

Differences in terms of Final Accuracy In our final analysis, we consider the rank correlation between down-stream accuracy and the various metrics, split according to their final accuracy. Figure 7.7 shows the rank correlation for the worst-performing fifty percent of the models after 100K steps (top), and for the best performing fifty percent (bottom). While these results should be interpreted with care, as the split depends on the final accuracy, we can still make several interesting observations: It can be seen that disentanglement (i.e. *FactorVAE* score) remains strongly correlated with down-stream performance for both splits in the few-sample regime. At the same time, the benefit of lower *Reconstruction* error appears to be limited to the worst 50% of models. This is intuitive, as when the *Reconstruction* error is too high there may not be enough information present to solve the down-stream tasks. However, regarding the top-performing models (best 50%), it can be seen that the *relative* gains from further reducing reconstruction error are of limited use, and in fact in some cases yield positive correlation.

7.3 Discussion

In this chapter, we investigated whether disentangled representations enable one to learn non-trivial down-stream tasks using fewer samples. First, we created two abstract visual reasoning tasks based on existing data sets for which the ground-truth factors of variation are known. Then, we trained a diverse set of disentanglement models based on four state-of-the-art disentanglement approaches and evaluated their representations using multiple abstract reasoning models. This allowed us to observe compelling evidence that more disentangled representations are more sample-efficient in the considered down-stream learning task.

We draw two main conclusions from these results: First, they provide concrete motivation for why one might want to pursue disentanglement as a property of learned representations in the unsupervised case, e.g. as a format for object

representations. Second, we still observed differences between disentanglement metrics, which should motivate further work in understanding what different properties they precisely capture. This should include an analysis of how different metrics respond when information is missing from the representation and to what extent they measure explicitness. None of the metrics achieved perfect correlation in the few-sample regime, which also suggests that it is not yet fully understood what makes one representation better than another in terms of learning. Finally, we point out that it might be useful to extend the methodology in this study to other complex down-stream tasks, or include an investigation of other purported benefits of disentangled representations, such as out of distribution generalization.

Chapter 8

Conclusion

This dissertation provides several contributions towards solving the problem of representation learning for visual reasoning tasks using deep neural networks.

Underlying several of our contributions is the binding problem in neural networks, which was proposed in Chapter 3 as the principal cause for the inability of existing neural networks to efficiently form, represent and relate symbol-like entities that are essential to many high-level cognitive tasks, such as reasoning. This perspective allowed us to identify several challenges and requirements that must be overcome to address this issue and point out commonalities between seemingly independent methods that are part of a rich history of research on vision and more symbolic reasoning tasks using neural networks.

Our other contributions in Chapters 4 to 7 demonstrate how several of these challenges can be addressed in the context of visual reasoning, namely through formulating neural networks that incorporate a suitable inductive bias. In that sense, these contributions can be viewed as working towards a solution to the binding problem in neural networks. An alternative perspective is that of the methods proposed in each of these chapters contributing to problem-solving in their respective domains. However, in this case, several of the restrictions that we imposed (e.g. regarding access to supervised data) may be regarded as overly limiting and certain other design choices become more difficult to justify.

In the following we highlight our contributions included in Chapters 3 to 7 of this disseration, followed by an outlook on open problems and promising directions for future research.

The Binding Problem in Artificial Neural Networks [Greff et al., 2020]
We have proposed that there exists an underlying cause for the lack of emergent symbolic processing in existing neural networks, which we refer to as the binding

problem. It is due to their inability to dynamically and flexibly combine information that is distributed throughout the network, which is required to effectively form, represent, and relate symbol-like entities. We have proposed to view the binding problem in terms of three subproblems: representation, segregation, and composition.

The representation problem revolves around object representations, which act as basic building blocks for neural processing to behave symbolically. Encoding information about object representations in neural networks is difficult, which we have argued is mostly due to the lack of a suitable mechanism for information separation when pursuing a common format. Another challenge is concerned with ensuring their utility as a stable foundation for reasoning and other types of information processing in a dynamic world that constantly changes.

The segregation problem is about the process of forming grounded object representations from raw unstructured inputs. We have argued how this is complicated by the notion of an object, which is context- and task-dependent and therefore difficult to formalize. To overcome this problem we have put forth several criteria that characterize their functional properties, such as modularity and hierarchy. Moreover, we have argued how incorporating a form of top-down feedback, multistability, and object identity, provide key challenges that must be addressed.

The composition problem is about leveraging the modularity of object representations to build structured models for inference, prediction, and behavior that generalize more systematically. It involves learning about general relations that can be combined with object representations to build structures that imply different patterns of generalization. We have argued how this requires mechanisms for dynamic variable binding, structure inference, and a means to interface with segregation and representation.

Our primary contribution in Sections 3.2 to 3.4 was to outline these challenges and requirements in detail, while also providing a survey of (recent) prior works that offer promising techniques for addressing some of these.

Neural Expectation Maximization [Greff* and van Steenkiste* et al., 2017a] In Chapter 4 we have proposed Neural Expectation Maximization (N-EM), which addresses several key aspects of segregation and representation. It learns about objects by focusing on their functional role as abstract computational units that are modular and reusable across many different contexts. Moreover, it makes use of instance slots that are described in a common format and separate information about individual objects.

N-EM combines neural spatial mixture models with generalized EM to implement a trainable clustering algorithm that can adapt to accommodate various definitions of an object, based solely on the observed statistical regularities in the data. In our experiments, we were able to confirm that N-EM is able to learn to group pixels according to objects and represent an image as a composition of object representations, unlike standard representation learning approaches.

We also contributed RNN-EM, a variation of N-EM that treats each object representation as the hidden state of a recurrent neural network. RNN-EM naturally extends to sequential data when modifying its learning objective to make predictions about the future. This makes it easier to learn about objects since pixels belonging to the same object usually share a common fate, and we were able to show how RNN-EM is able to learn object representations even in the presence of heavy occlusion.

Relational Neural Expectation Maximization [van Steenkiste et al., 2018a]
In Chapter 5 we have proposed Relational Neural Expectation Maximization (R-NEM), a structured model for common-sense physical reasoning that addresses several aspects of composition. It combines the learned object representations of RNN-EM with a compositional interaction function, which provides a mechanism for modeling general relations between objects via a kind of dynamic variable binding. R-NEM was the first end-to-end neural approach to leverage the underlying compositionality of learned object representations to perform physical prediction in a purely unsupervised fashion.

In our experiments, we found that the incorporated structure allows R-NEM to capture the (physical) dynamics of various environments more accurately than other methods. It was also shown how, due to its compositional structure, this learned knowledge can be extrapolated to environments with different numbers of objects. In that sense, R-NEM was found to be able to generalize more systematically and more predictably in a way that is similar to how one would expect humans to be able to generalize. Finally, it was demonstrated how R-NEM can be used as an approximate simulator of the environment.

Object Compositionality in GANs [van Steenkiste et al., 2020]
In Chapter 6 we have investigated an alternative approach to learning about objects based on GANs. We found that by incorporating architectural structure in the generator of a GAN, it is able to learn to generate images as a composition of individual objects and background, while also considering their relations.

We demonstrated how this approach can more easily be applied to images

containing color, background, and complex object shapes. In particular, our approach was the first purely unsupervised object-centric approach that was able to scale to the complex CLEVR dataset. Compared to a strong baseline of GANs, we were able to demonstrate how the proposed structure helps in learning a generative process that is more faithful to the observed reference distribution.

In the context of segregation, we have argued how it is important that the learned generative process can be 'inverted' to learn about objects. To that extend, we formulated an inference procedure that facilitates instance segmentation. It leverages the empirical observation that the proposed structured generator distinguishes objects at a representational level and that it learns a generative process that can be semantically understood in terms of this structure. We demonstrated how the resulting instance segmentation model is able to generalize to unseen images for the purpose of segregation.

Evaluating Disentangled Representations [van Steenkiste et al., 2019]
In Chapter 7 we contributed a large-scale evaluation of disentangled representations as a way of encoding information about objects for the purpose of learning down-stream tasks. Rather than focusing on a simple single factor classification task, we evaluated their usefulness on two new abstract visual reasoning tasks that challenge the current capabilities of state-of-the-art deep neural networks. On these tasks, we were able to observe compelling evidence that more disentangled representations yield better sample-efficiency.

These observations offer a more positive outlook for learning disentangled representations compared to a prior study and provide concrete motivation for why one would want to pursue this particular format. We were also able to observe large differences between metrics that measure different notions of disentanglement, which motivates further work in understanding what different properties they precisely capture. None of the metrics achieved perfect correlation in the few-sample regime, which indicates that it is not yet fully understood what makes one representational format better than another in terms of down-stream learning.

8.1 Future Directions

In Sections 3.2 to 3.4 we have hinted at many avenues for future research that specifically focus on addressing the binding problem. In addition, we would like to emphasize the following three high-level directions in particular.

The Friction Between Architectural Structure and Flexibility It can not go unnoticed that several of the results presented in this dissertation focus on synthetic tasks that are still far from real-world complexity. In contrast, there exist neural network approaches that have shown capable of modeling more complex (real-world) visual settings, e.g. [Babaeizadeh et al., 2018; Brock et al., 2019; Donahue and Simonyan, 2019]. A defining characteristic of these models is that they do not incorporate structure at the representational level of objects, which we have argued is problematic for the purpose of more systematic generalization. Nonetheless, they can be applied to these more visually complex settings and generalize reasonably well under i.i.d. assumptions. This may suggest that the architectural inductive biases that we have considered in this work are too restrictive.

One source of friction, which we already encountered in Chapter 6, is due to learning objectives that operate on the level of pixels. When paired with a representation that distinguishes objects, this requires prescribing how objects are combined to form realistic images, which is a complex process that involves many subtle interactions, such as lighting, occlusion, depth, and shadow. These interactions appear to conflict with the understanding that, at a representational level, objects should act as modular (and independent) building blocks, and which drives current approaches to segregation. We believe that an important future direction is to address this friction (between incorporating more structure and offering flexibility) to scale structured models that address the binding problem to more real-world datasets.

Extending Object Representations to Other Sensory Domains In this dissertation, we have mostly focused on learning object representations in the visual domain, although the notion of an object in its most general form equally applies to many other sensory domains such as audio or tactile. For example, auditory objects may correspond to different sources of sound, such as speakers talking simultaneously in the same room (cocktail-party problem [Cherry, 1953]). Likewise, it implies that objects can be simultaneously grounded in sensory information from multiple domains, which may help resolve ambiguities (e.g. the McGurk Effect [Mcgurk and Macdonald, 1976]). A promising avenue for research, therefore, is to investigate how existing techniques for the image domain, like those explored in this dissertation, can be applied to other sensory domains, and whether more general principles can be derived for discovering, representing, and relating abstract concepts across (multiple) different domains. Similarly, this may let us refine our current understanding of what visual objects are, and help pave the way towards a practical definition of objects in their most general form.

Synergistic Interactions between Problem-Solving Agents and Model Building The problems of segregation, representation, and composition in this work were treated mostly from the perspective of model building and unsupervised learning. Indeed, while R-NEM provides a single system that addresses several aspects of each of these subproblems, it did not address several others that critically rely on interaction with an agent. Studying how model-building and problem-solving, e.g. in the form of an agent interacting with the world that seeks to maximize reward, can mutually benefit each other is crucial to take this next step. Neural Controller-Model (CM) systems [Schmidhuber, 2015b] where the controller 'steers' the model to provide a form of top-down feedback is one promising direction, and recent work is taking a step in that direction [Mott et al., 2019]. Future research should address how such feedback can be used to guide segregation, and explore other synergistic interactions between interacting agents and model building. For example, unsupervised reinforcement learning objectives based on artificial curiosity [Schmidhuber, 1991b; Haber et al., 2018] may help a model acquire a biased set of observations (focusing on a particular aspect of the world) from which it may be easier to learn corresponding representations.

Appendix A

Additional Experiment Details

A.1 Neural Expectation Maximization

In all experiments we use ADAM [Kingma and Ba, 2015] with default parameters (unless otherwise mentioned) and a batch size of 64 to train the neural networks. We use 50K observations for training, 10K for validation, and 10K for testing. The quality of the learned groupings is evaluated by computing the Adjusted Mutual Information (AMI [Vinh et al., 2010]) with respect to the ground-truth, while ignoring the background and overlap regions (as is consistent with earlier work [Greff et al., 2015, 2016]). We use early stopping when the validation loss has not improved for 10 epochs.

A.1.1 Static Shapes

Each input consists of a 28×28 binary image containing three regular shapes ($\triangle\triangledown\square$) located in random positions [Reichert and Serre, 2013].

For N-EM we implement f_ϕ using a single-layer fully-connected neural network with a Sigmoid activation function. It receives a real-valued 250-dimensional vector $\boldsymbol{\theta}_k$ as input and outputs for each pixel a value that parametrizes a Bernoulli distribution. We squash $\boldsymbol{\theta}_k$ with a Sigmoid before passing it to the network and train an additional weight η to implement the learning rate that is used to combine the gradient ascent updates into the current parameter estimate.

For RNN-EM we use an RNN with 250 Sigmoidal hidden units and a fully-connected output-layer with a Sigmoid activation function that parametrizes a Bernoulli distribution for each pixel in the same fashion.

We train both networks with $K = 4$ for 15 EM steps and add bitflip noise ($p = 0.1$) to each of the pixels. The prior for each pixel in the data is set to a

Bernoulli distribution with $p = 0$. The outer-loss is only evaluated at the final EM-step.

A.1.2 Flying Shapes

Each input consists of a sequence of binary 28×28 images containing a fixed number of shapes ($\triangle\triangledown\square$) that start in random positions and float along randomly sampled trajectories within the image for 20 steps.

We use a convolutional encoder-decoder architecture based on Chen et al. [2016] with a recurrent neural network as bottleneck (Table A.1). Instead of using transposed convolutions (to implement the "up-convolution") we first reshape the image using the default nearest-neighbor interpolation followed by a normal convolution to avoid frequency artifacts [Odena et al., 2016]. Note that we do not add LayerNorm [Ba et al., 2016b] on the recurrent connection.

Encoder
4×4 conv, 32 ELU, stride 2, LayerNorm
4×4 conv, 64 ELU, stride 2, LayerNorm
FC, 512 ELU, LayerNorm
Recurrent
RNN, 100 Sigmoid, LayerNorm (on output)
Decoder
FC, 512 ReLU, LayerNorm
FC, $7 \times 7 \times 64$ ReLU, LayerNorm
4×4 upconv, 32 ReLU, stride 2, LayerNorm
4×4 upconv, 1 Sigmoid, stride 2

Table A.1. Encoder, Recurrent, and Decoder architectures on Flying Shapes.

At each time-step t we feed $\boldsymbol{\gamma}_{:,k}[t-1] \odot (\boldsymbol{\psi}_{:,k}[t-1] - \tilde{\boldsymbol{x}}[t])$ as input to the network, where $\tilde{\boldsymbol{x}}$ is the input with added bitflip noise ($p = 0.2$). RNN-EM is trained with a next-step prediction objective implemented by replacing $\boldsymbol{x}$ with $\boldsymbol{x}[t+1]$ in (4.5), which we evaluate at each time-step. A single RNN-EM step is used for each time-step. The prior for each pixel in the data is set to a Bernoulli distribution with $p = 0$. We prevent conflicting gradient updates by not backpropagating any gradients through $\boldsymbol{\gamma}$.

A.1.3 Flying MNIST

Each input consists of a sequence of gray-scale 24×24 images containing a fixed number of down-sampled (by a factor of two along each dimension) MNIST digits [LeCun et al., 1998] that start in random positions and "fly" across randomly sampled trajectories within the image for T time-steps.

We use a slightly deeper version of the architecture used for flying shapes (Table A.2).

Encoder
4×4 conv, 32 ELU, stride 2, LayerNorm
4×4 conv, 64 ELU, stride 2, LayerNorm
4×4 conv, 128 ELU, stride 2, LayerNorm
FC, 512 ELU, LayerNorm
Recurrent
RNN, 250 Sigmoid, LayerNorm (on output)
Decoder
FC, 512 ReLU, LayerNorm
FC, $3 \times 3 \times 128$ ReLU, LayerNorm
4×4 upconv, 64 ReLU, stride 2, LayerNorm
4×4 upconv, 32 ReLU, stride 2, LayerNorm
4×4 upconv, 1 linear, stride 2

Table A.2. Encoder, Recurrent, and Decoder architectures on Flying MNIST.

The training procedure is largely identical to the one described for flying shapes except that we replace the bitflip noise with masked uniform noise: we first sample a binary mask from a multivariate Bernoulli distribution with $p = 0.2$ and then use this mask to interpolate between the original image and samples from a Uniform(0,1) distribution (range is the same as for the input data). We use a learning rate of 0.0005 (from the second stage onwards in case of stage-wise training), scale the second-loss term by a factor of 0.2 and find it beneficial to normalize the masked differences between the prediction and the image (zero mean, standard deviation one) before passing it to the network.

A.2 Relational Neural Expectation Maximization

In all experiments we use ADAM [Kingma and Ba, 2015] with default parameters and a batch size of 64 to train the neural networks. We use 50K observations for training, 10K for validation, and 10K for testing. The quality of the learned groupings is evaluated by computing the Adjusted Rand Index (ARI [Hubert and Arabie, 1985]) with respect to the ground-truth while ignoring the background and overlap regions (as is done in N-EM). We use early stopping when the validation loss has not improved for 10 epochs.

At each time-step t we feed $\boldsymbol{\gamma}_{:,k}[t-1] \odot (\boldsymbol{\psi}_{:,k}[t-1] - \tilde{\boldsymbol{x}}[t])$ as input to the encoder, where $\tilde{\boldsymbol{x}}$ is the input with added bitflip noise ($p = 0.2$). Consistent with RNN-EM, R-NEM is trained with a next-step prediction objective by adapting (4.5) and we use a Bernoulli for each pixel likelihood since observations are binary. Similarly, the pixel prior is set to a Bernoulli with $p = 0$, and we prevent conflicting gradient updates by not backpropagating any gradients through $\boldsymbol{\gamma}$. We also assume fixed equal mixing probabilities for all components and ignore the mixing prior $p(\boldsymbol{z}|\boldsymbol{\pi})$.

A.2.1 Bouncing Balls

The bouncing balls data set is similar to previous work [Sutskever et al., 2009] with a few modifications. It consists of sequences of 64×64 binary images over 30 time-steps and balls are randomly sampled from two types: one ball is six times heavier and 1.25 times larger in radius than the other. The balls are initialized with random initial positions and velocities. Balls bounce elastically against each other and the image window.

We use the convolutional encoder-decoder architecture with a recurrent neural network as bottleneck described in Table A.3 for the RNN-EM part, which is similar to Table A.2, that is updated according to (5.1).

The Interaction Function $\boldsymbol{\Upsilon}^{\text{R-NEM}}$ is structured as in Table A.4. We experimented with deeper architectures but were unable to observe significant improvements.

Comparison and Extrapolation In the comparison experiment both R-NEM and RNN-EM are trained with $K = 5$ (unless otherwise mentioned), following our insight in Chapter 4 that training with slightly larger K is beneficial. On the extrapolation task, we adjusted the number of components at test-time to $K = 8$.

When comparing to RNN-EM, we used $\boldsymbol{\Upsilon} = \boldsymbol{\Upsilon}^{\text{RNN-EM}}$. For comparing to RNN we set $K = 1$, and used $\boldsymbol{\Upsilon} = \boldsymbol{\Upsilon}^{\text{RNN-EM}}$, yielding a standard recurrent autoencoder that receives at each time-step the difference between its previous prediction and

Encoder
4×4 conv, 16 ELU, stride 2, LayerNorm
4×4 conv, 32 ELU, stride 2, LayerNorm
4×4 conv, 64 ELU, stride 2, LayerNorm
FC, 512 ELU, LayerNorm
Recurrent
RNN, 250 Sigmoid, LayerNorm (on output)
Decoder
FC, 512 ReLU, LayerNorm
FC, $8 \times 8 \times 64$ ReLU, LayerNorm
4×4 upconv, 32 ReLU, stride 2, LayerNorm
4×4 upconv, 16 ReLU, stride 2, LayerNorm
4×4 upconv, 1 Sigmoid, stride 2

Table A.3. Encoder, Recurrent, and Decoder architectures on Bouncing Balls.

Interaction Function	
MLP^{enc} :	FC, 250 ReLU, LayerNorm
MLP^{emb} :	FC, 250 ReLU, LayerNorm
MLP^{eff} :	FC, 250 ReLU, LayerNorm
MLP^{att} :	FC, 100 Tanh, LayerNorm $\rightarrow$ FC, 1 Sigmoid

Table A.4. Interaction Function architecture on Bouncing Balls.

the noisy ground-truth as input. In case of LSTM, we additionally replace the RNN update in (5.1) with an LSTM update. The R-NEM *no att* model is the same as R-NEM but without MLP^{att}, such that α is always 1 in (5.6).

Simulation Since the E-step relies on the ground-truth, which is not available for simulation, we used a thresholded version of $\max_k \boldsymbol{\psi}_{:,k}$ at 0.1 (such that everything below becomes 0 and everything above becomes 1) as $\boldsymbol{\gamma}$ instead. The observation at time-step t for simulation steps is computed as $\boldsymbol{x}[t] \approx \sum_k \boldsymbol{\gamma}_{:,k}[t-1] \odot \boldsymbol{\psi}_{:,k}[t-1]$, from which we can then compute the top-down error as before that is fed to the encoder. In this way, we were able to replicate the dynamics when real observations are available as much as possible.

Occlusion On the occlusion data set, we used three balls with equal mass. The curtain was spawned at a random location for each sequence. We trained R-NEM with $K = 5$.

A.2.2 Space Invaders

We used a pre-trained DQN [Mnih et al., 2015] to produce a data set with sequences of 25 time-steps. The DQN receives a stack of four frames as input and we recorded every first frame of this stack. These frames were first pre-processed as in Mnih et al. [2015] and then thresholded at 0.0001 to obtain binary images.

Since the images are 84×84, we used a different encoder and decoder (Table A.5). We used the same architecture for $\Upsilon^{\text{R-NEM}}$ (Table A.4), with the only difference that at each time-step we concatenated an embedding of the action produced by the agent to the hidden state. We used a single-layer MLP with 10 units and a *ReLU* activation function to compute this embedding.

In the Atari experiment, we trained with $K = 4$ and reduced the input noise to 0.02 in order to preserve tiny elements such as bullets (that only occupy 1-2 pixels) as much as possible.

Encoder
4×4 conv, 16 ELU, stride 2, LayerNorm
4×4 conv, 32 ELU, stride 2, LayerNorm
4×4 conv, 32 ELU, stride 2, LayerNorm
4×4 conv, 32 ELU, stride 2, LayerNorm
FC, 512 ELU, LayerNorm
Recurrent
RNN, 250 Sigmoid, LayerNorm (on output)
Decoder
FC, 512 ReLU, LayerNorm
FC, $8 \times 8 \times 64$ ReLU, LayerNorm
4×4 upconv 32 ReLU, stride 2, LayerNorm
4×4 upconv 32 ReLU, stride 2, LayerNorm
4×4 upconv 16 ReLU, stride 2, LayerNorm
4×4 upconv 1 Sigmoid, stride 2

Table A.5. Encoder, Recurrent, and Decoder architectures on Space Invaders.

A.3 Object Compositionality in GANs

A.3.1 Model specifications

The generator and discriminator neural network architectures in all our experiments are based on DCGAN [Radford et al., 2015].

Object Generators *k-GAN ind.* introduces $K = k$ copies of an object generator (i.e. tied weights, DCGAN architecture) that each generate an image from an independent sample $\boldsymbol{z}_i$ of a 64-dimensional Uniform$(-1, 1)$ prior $p_{\boldsymbol{Z}}$.

Relational Structure When relational structure is incorporated (*k-GAN rel.*) each of the $\boldsymbol{z}_i$ is first updated, before being passed to the generators. These updates are computed using one or more *attention blocks*, which integrate Multi-Head Dot-Product Attention (MHDPA) [Vaswani et al., 2017] with a post-processing step [Zambaldi et al., 2019]. A single head of an attention block updates $\boldsymbol{z}_i$ according to (6.3), (6.4), (6.5), and (6.6).

In our experiments, we implement the relational structure as in Table A.6. Different heads in the same block use different parameters for these MLPs. If multiple heads are present, then their outputs are concatenated and transformed via a single-layer neural network (FC, 64 ReLU, LayerNorm) to obtain the new $\hat{\boldsymbol{z}}_i$. If the relational structure incorporates multiple attention blocks that *iteratively* update $\boldsymbol{z}_i$, then we consider two variations: using unique weights for each MLP in each block or sharing their weights across blocks.

Relational Structure
MLP^{query} : FC, 32 ReLU, LayerNorm
MLP^{key} : FC, 32 ReLU, LayerNorm
MLP^{value} : FC, 32 ReLU, LayerNorm
$\text{MLP}^{project}$: FC, 64 ReLU $\rightarrow$ FC, 64 ReLU $\rightarrow$ LayerNorm (after adding to $\boldsymbol{z}_i$)

Table A.6. Neural network architectures used to implement the relational structure. Different heads use their own parameters for these MLPs.

Background Generation When a background generator is incorporated (e.g. *k-GAN rel. bg*) it uses the same DCGAN architecture as the object generators, yet maintains its own set of weights. It receives as input its own latent sample $\boldsymbol{z}_b$

where p_{z_b} is again Uniform$(-1, 1)$ similar to for the object generators, although one may, in theory, choose a different distribution. We explore both variations in which $\boldsymbol{z}_b$ participates in the relational structure, and in which it does not.

A.3.2 Hyperparameter Configurations

Each model is optimized with ADAM [Kingma and Ba, 2015] using a learning rate of 0.0001, and batch size 64 for 1M steps. Each generator step is followed by 5 discriminator steps, as is considered best practice in training GANs. Checkpoints are saved every 20K steps and we consider only the checkpoint with the lowest FID for each hyperparameter configuration. FID is computed using 10K samples from a hold-out set.

Baseline We conduct an extensive grid search over 48 different GAN configurations to obtain a strong GAN baseline on each data set. It is made up of hyperparameter ranges that were found to be successful in training GANs on standard data sets [Kurach et al., 2019].

We consider [SN-GAN / WGAN], using [NO / WGAN] gradient penalty with λ [1 / 10]. In addition, we consider these configurations [WITH / WITHOUT] spectral normalization. We consider [(0.5, 0.9) / (0.5, 0.999) / (0.9, 0.999)] as (β_1, β_2) in ADAM. We explore 5 different seeds for each configuration.

k-GAN We conduct a similar grid search for the GANs that incorporate our proposed structure. However, in order to maintain a similar computational budget compared to our baseline, we consider a *subset* of the previous ranges to compensate for the additional hyperparameters of the different structured components that we would like to search over.

In particular, we consider SN-GAN with WGAN gradient penalty, with a default λ of 1, [WITH / WITHOUT] spectral normalization. We use (0.9, 0.999) as fixed values for (β_1, β_2) in ADAM. Additionally, we consider K = [3 / 4 / 5] copies of the generator, and the following configurations for the relational structure:

- Independent
- Relational (1 block, no weight-sharing, 1 head)
- Relational (1 block, no weight-sharing, 2 heads)
- Relational (2 blocks, no weight-sharing, 1 head)

- Relational (2 blocks, weight-sharing, 1 head)
- Relational (2 blocks, no weight-sharing, 2 heads)
- Relational (2 blocks, weight-sharing, 2 heads)

This results in 42 hyperparameter configurations, for which we each consider 5 seeds. We do not explore the use of a background generator on the non-background data sets. On the background data sets, we explore variations with and without the background generator. In the former case, we search over an additional hyperparameter that determines whether the latent representation of the background generator should participate in the relational structure or not, while for the latter we increment all values of K by 1.

IODINE We made use of the official trained model for CLEVR released by the authors [Greff et al., 2019]. We used $K = 7$ to compute the FID, which was also used for training. Lower values of K were considered but were not found to yield any improvements.

A.3.3 Instance Segmentation

We select the 5 best *3-GAN* models (according to FID) on *Independent MM*, *Triplet MM*, *RGB Occluded MM*, *CIFAR10 + MM*. These include relational, and purely independent models, although on *CIFAR10 + MM* we ensure that we select only models that also incorporate the background extension. Segmentation data is obtained by either thresholding the output of the object generators at 0.1 (on *Independent MM*, *Triplet MM*) or using the (implicit) alpha masks (*RGB Occluded MM*, *CIFAR10 + MM*). Ground-truth data is obtained in the same fashion, but using the real images of digits and background before combining them.

Our segmentation architecture is similar to the one considered in Chen et al. [2019] but trained in a supervised fashion. We use a straight-through estimator that first considers all viable permutations of assigning segmentation outputs to labels, and then backpropagates only the gradients of the permutation for which we measured the lowest cross-entropy loss. We trained for 150 epochs using ADAM with a learning rate of 0.002 and a batch size of 64. The training data consisted of 50K images and corresponding segmentations (obtained via the procedure described above) and model selection was performed using a separate validation set consisting of 10K data points.

Reported results were obtained by evaluating on a separate test set of 10K *ground-truth* data points. In particular, we test the trained segmenter using

ground-truth segmentations and report the Adjusted Rand Index (ARI [Hubert and Arabie, 1985]) score on all pixels that correspond to digits (similar to as was done for N-EM and R-NEM). On *Independent MM*, and *Triplet MM* we additionally ignore overlapping pixels due to ambiguities.

A.3.4 Human Study

We asked human raters to compare the images generated by *k-GAN* ($k = 3, 4, 5$) to our *GAN* baseline on *RGB Occluded MM*, *CIFAR10 + MM* and *CLEVR*, using the configuration with a background generator for the last two data sets. For each model, we selected the 10 best hyperparameter configurations, from which we each generated 100 images. We conduct two different studies 1) in which we compare images from *k-GAN* against *GAN* and 2) in which we asked raters to answer questions about the content (properties) of the images. In all cases, human raters were shown several samples of real images, a description of the generative process, and examples of correct ratings for the "properties" experiments (but not for the subjective evaluation).

Comparison We asked raters to compare the quality of the generated images. We asked up to three raters for each image and report the majority vote or "equal" if no decision can be reached. Note that on $CLEVR$ we instructed the raters to ignore visual implausibilities due to floating objects (for both *k-GAN* and *GAN*) that may arise due to the fixed order in (6.7), and measured this effect separately in Figure B.5.

Properties For each data set, we asked (up to three raters for each image) the following questions.

On *RGB Occluded MM* we asked:

1. How many [red, blue, green] shapes are in the image? [0, 1, 2, 3, 4, 5]
2. How many are recognizable as digits? [0, 1, 2, 3, 4, 5]
3. Are there exactly 3 digits in the picture, one of them green, one blue, and one red? Yes / No

On *CIFAR10 + MM* we asked these same questions, and additionally asked:

4. Does the background constitute a realistic scene? Yes / No

On *CLEVR* we asked the following set of questions:

1. How many shapes are in the image? [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2. How many are recognizable as geometric objects? [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3. Are there any objects with mixed colors (e.g. part green part red)? Yes / No
4. Are there any objects with distorted geometric shapes? Yes / No
5. Are there any objects that appear to be floating? Yes / No
6. Does the scene appear to be crowded? Yes / No

A.4 Evaluating Disentangled Representations

A.4.1 Architectures

Disentanglement Methods

We use the same architecture, hyperparameters, and training setup as in prior work [Locatello et al., 2018], which we report here for completeness. The encoder and decoder architectures are depicted in Table A.7. All models share the following hyperparameters: We used a batch size of 64, 10-dimensional latent space, and Bernoulli decoders. We trained the models for 300K steps using ADAM [Kingma and Ba, 2015] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and a learning rate of 0.0001.

For *β-VAE*, we perform a sweep on β on the interval [1, 2, 4, 6, 8, 16]. For *β-VAE* with *controlled capacity increase*, we perform a sweep on c_{max} on the interval [5, 10, 25, 50, 75, 100]. The iteration threshold is set to 100K and $\gamma = 1000$. For *FactorVAE*, we perform a sweep on γ on the interval [10, 20, 30, 40, 50, 100]. For the discriminator of the *FactorVAE* we use the architecture described in Table A.8. Its other hyperparameters are: Batch size = 64, Optimizer = Adam with $\beta_1 = 0.5$, $\beta_2 = 0.9$, $\epsilon = 10^{-8}$, and learning rate = 0.0001. For *DIP-VAE-I*, we perform a sweep on λ_{od} on the interval [1, 2, 5, 10, 20, 50], and set $\lambda_d = 10$. For *DIP-VAE-II*, we perform a sweep on β on the interval [1, 2, 5, 10, 20, 50], and set $\lambda_d = 10$. For *β-TCVAE*, we perform a sweep on β on the interval [1, 2, 4, 6, 8, 10]. Each model is trained using 5 different random seeds.

Abstract Visual Reasoning Method

To solve the abstract reasoning tasks, we implemented the *Wild Relation Networks (WReN)* model as in Santoro et al. [2018b]. For the experiments, we use the following random search space over the hyperparameters: We uniformly sample a learning rate for ADAM from the set $\{0.01, 0.001, 0.0001\}$ while $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. For the edge MLP g in the *WReN* model, we uniformly choose either 256 or 512 hidden units and we uniformly sample whether it has 2, 3, or 4 hidden layers. Similarly, for the graph MLP f in the *WReN* model, we uniformly choose either 128 or 256 hidden units and we uniformly sample whether it has 1 or 2 hidden layers before the final linear layer to compute the final score. We also uniformly sample whether we apply no dropout, dropout of 0.25, dropout of 0.5, or dropout of 0.75 to units before this last layer [Srivastava et al., 2014] (Frazier-Logue and Hanson [2018] note that dropout is a variation of Hanson [1990]).

Encoder
4×4 conv, 32 ReLU, stride 2
4×4 conv, 32 ReLU, stride 2
4×4 conv, 64 ReLU, stride 2
4×4 conv, 64 ReLU, stride 2
FC, 256 ReLU
FC 2×10
Decoder
FC, 256 ReLU
FC, $4 \times 4 \times 64$ ReLU
4×4 upconv, 64 ReLU, stride 2
4×4 upconv, 32 ReLU, stride 2
4×4 upconv, 32 ReLU, stride 2
4×4 upconv 3, stride 2

Table A.7. Encoder and Decoder architectures.

Discriminator
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 1000 leaky ReLU
FC, 2

Table A.8. Architecture for the discriminator in *FactorVAE*.

A.4.2 Abstract Visual Reasoning Data

Figures A.1 and A.2 contain additional examples (including answers) of the visual reasoning tasks for each data set respectively.

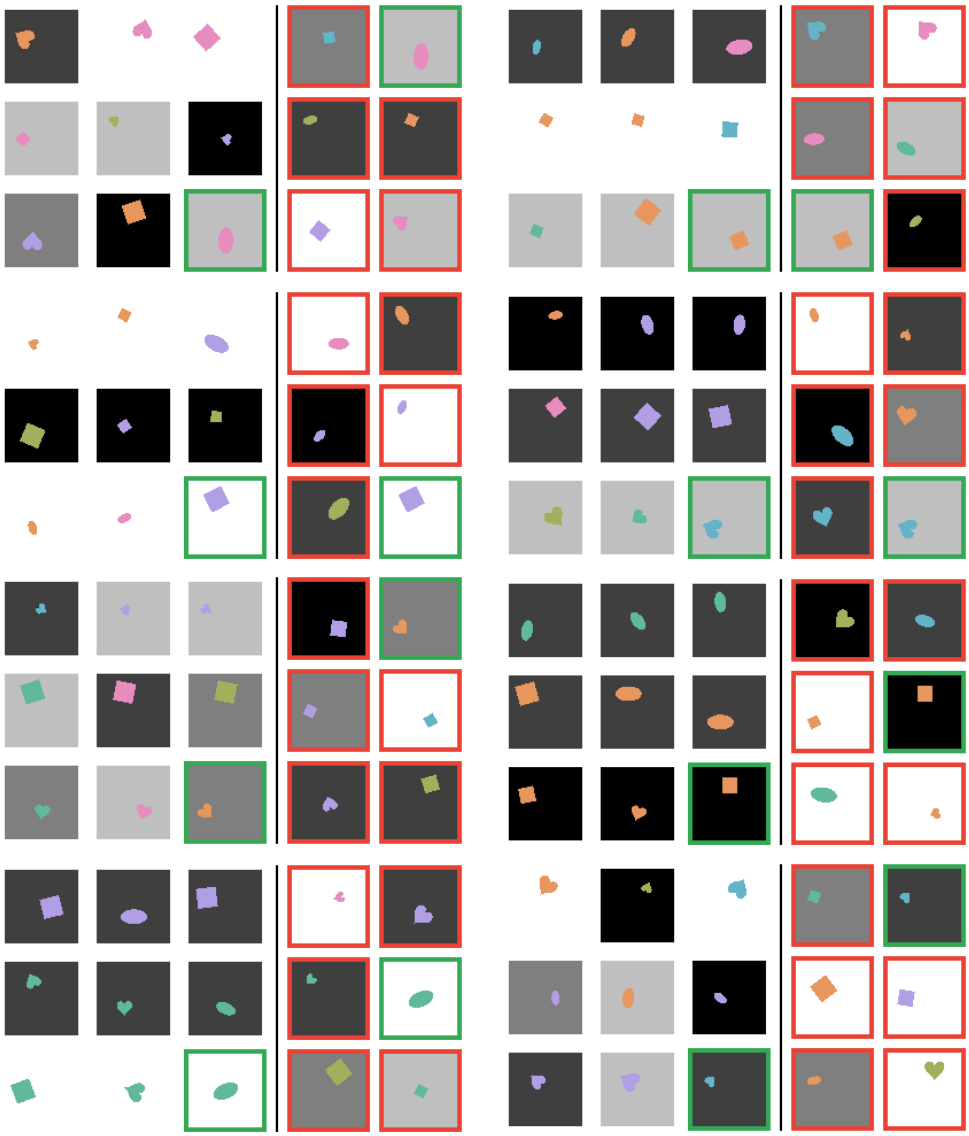


Figure A.1. Additional examples (including answers) of the RPM-like abstract visual reasoning task using *dSprites*.

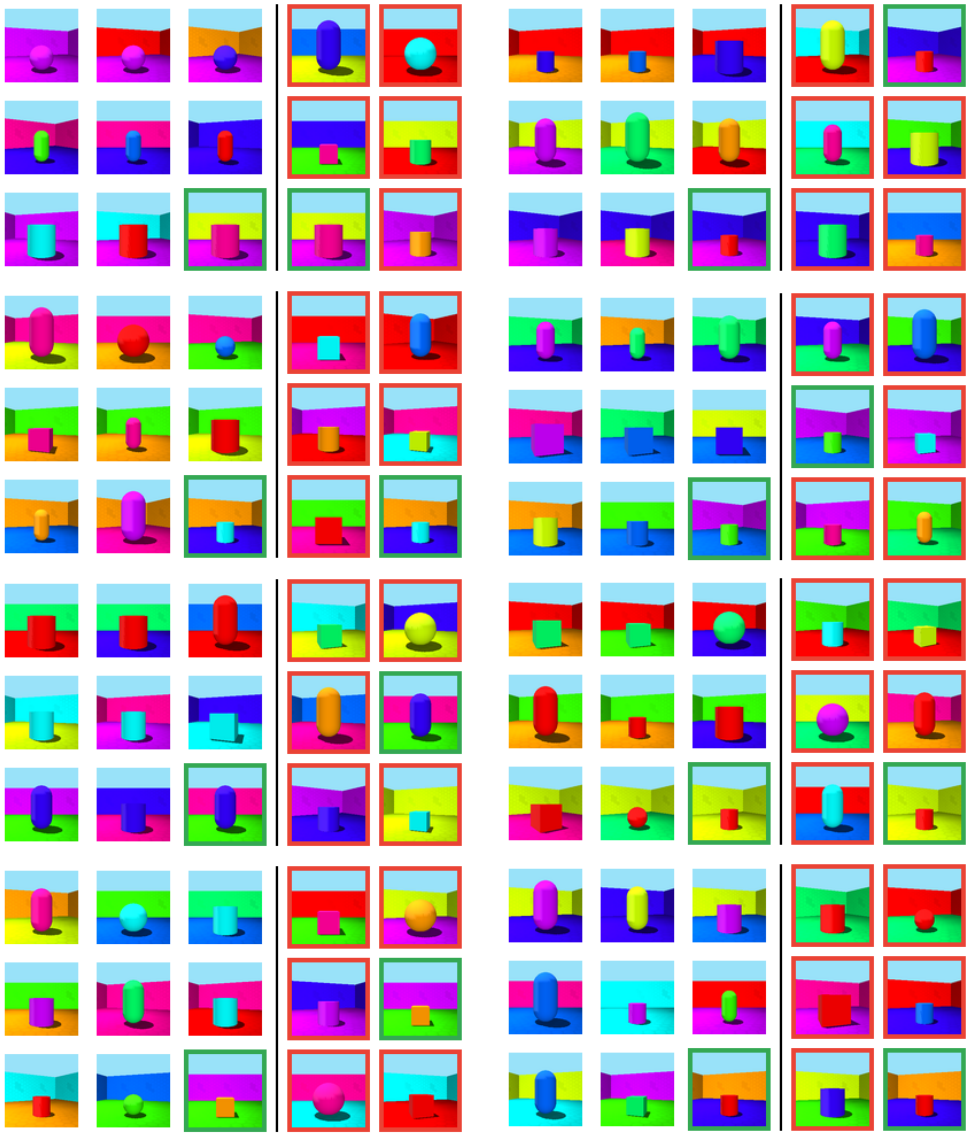


Figure A.2. Additional examples (including answers) of the RPM-like abstract visual reasoning task using *3dshapes*.

Appendix B

Additional Results

B.1 Object Compositionality in GANs

B.1.1 FID Study

Figures B.1 to B.3 and Table B.1.

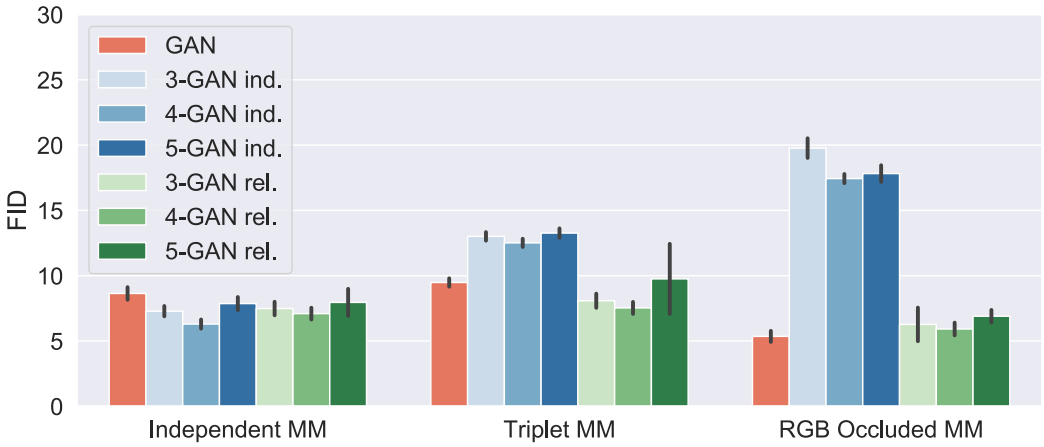


Figure B.1. **Analysis.** The best FID obtained by *GAN* and *k-GAN* on *Independent MM*, *Triplet MM*, and *RGB Occluded MM* following our grid search. The best configurations were chosen based on the smallest average FID (across 5 seeds). Standard deviations across seeds are illustrated with error bars.

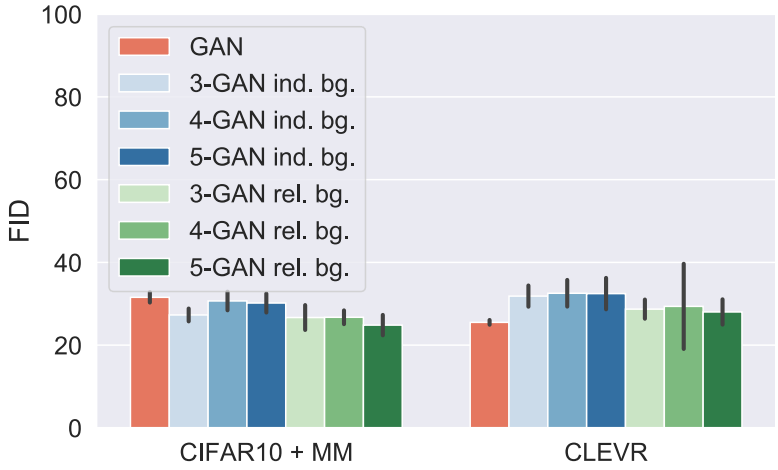


Figure B.2. **Analysis.** The best FID obtained by *GAN* and *k-GAN* on *MM + CIFAR10*, and *CLEVR* following our grid search. In this figure all *k-GAN* variations make use of the background extension. The best configurations were chosen based on the smallest average FID (across 5 seeds). Standard deviations across seeds are illustrated with error bars.

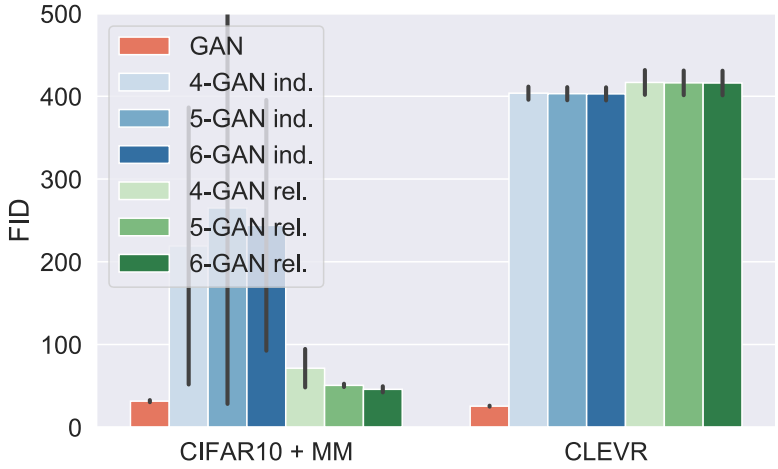


Figure B.3. **Analysis.** The best FID obtained by *GAN* and *k-GAN* on *MM + CIFAR10*, and *CLEVR* following our grid search. In this figure all *k-GAN* variations do not use the background extension. The best configurations were chosen based on the smallest average FID (across 5 seeds). Standard deviations across seeds are illustrated with error bars.

model	gan type	norm.	penalty	blocks	heads	share	bg. int.	β_1	β_2	λ
GAN	NS-GAN	spec.	none	x	x	x	x	0.5	0.999	10
3-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel.	NS-GAN	spec.	WGAN	1	1	no	x	0.9	0.999	1
4-GAN rel.	NS-GAN	spec.	WGAN	1	1	no	x	0.9	0.999	1
5-GAN rel.	NS-GAN	spec.	WGAN	2	1	no	x	0.9	0.999	1
GAN	NS-GAN	spec.	none	x	x	x	x	0.5	0.999	1
3-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel.	NS-GAN	spec.	WGAN	1	1	no	x	0.9	0.999	1
4-GAN rel.	NS-GAN	spec.	WGAN	1	2	no	x	0.9	0.999	1
5-GAN rel.	NS-GAN	spec.	WGAN	2	1	no	x	0.9	0.999	1
GAN	NS-GAN	spec.	none	x	x	x	x	0.5	0.999	1
3-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind.	NS-GAN	spec.	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel.	NS-GAN	none	WGAN	2	2	yes	x	0.9	0.999	1
4-GAN rel.	NS-GAN	none	WGAN	2	2	no	x	0.9	0.999	1
5-GAN rel.	NS-GAN	none	WGAN	2	2	yes	x	0.9	0.999	1
GAN	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel. bg.	NS-GAN	none	WGAN	2	1	yes	yes	0.9	0.999	1
4-GAN rel. bg.	NS-GAN	none	WGAN	2	1	yes	yes	0.9	0.999	1
5-GAN rel. bg.	NS-GAN	none	WGAN	2	2	yes	no	0.9	0.999	1
GAN	WGAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
4-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
5-GAN ind. bg.	NS-GAN	none	WGAN	x	x	x	x	0.9	0.999	1
3-GAN rel. bg.	NS-GAN	none	WGAN	2	1	no	yes	0.9	0.999	1
4-GAN rel. bg.	NS-GAN	none	WGAN	1	2	no	no	0.9	0.999	1
5-GAN rel. bg.	NS-GAN	none	WGAN	2	2	no	no	0.9	0.999	1

Table B.1. Best hyperparameter configuration for each model that were obtained following our grid search. Configurations were chosen based on the smallest average FID (across 5 seeds) as reported in Figure 6.5. Each block corresponds to a data set (from top to bottom: *Independent MM*, *Triplet MM*, *RGB Occluded MM*, *CIFAR10 + MM*, *CLEVR*).

B.1.2 Human Study - Properties

Figures B.4 to B.6 .

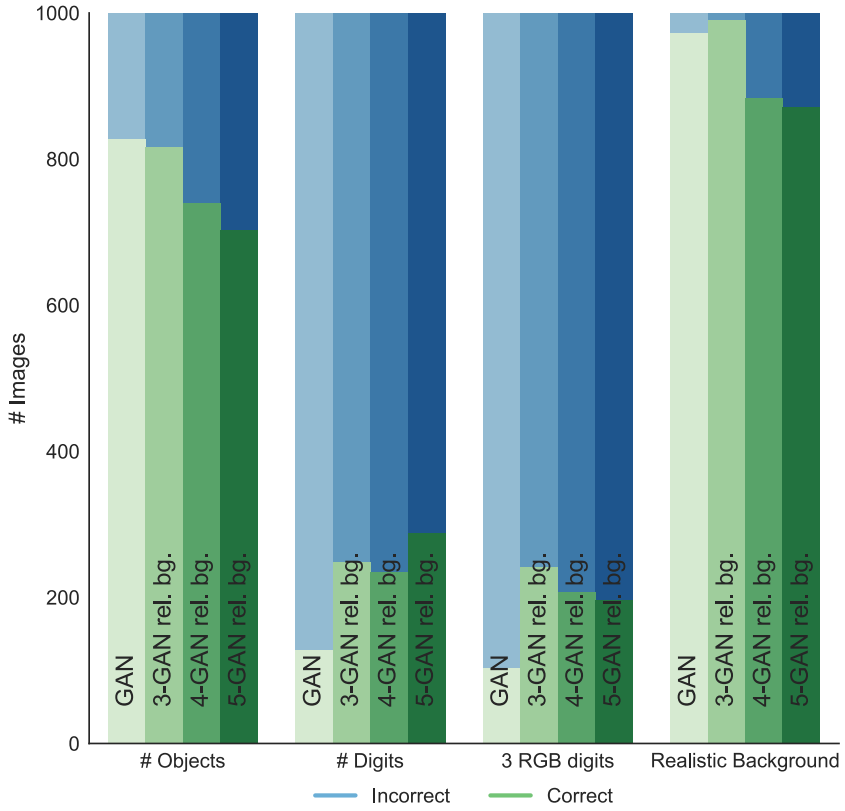


Figure B.4. Additional results of human evaluation. Properties of generated images by *k-GAN* (k = 3,4,5) and *GAN* on *CIFAR10 + MM*.

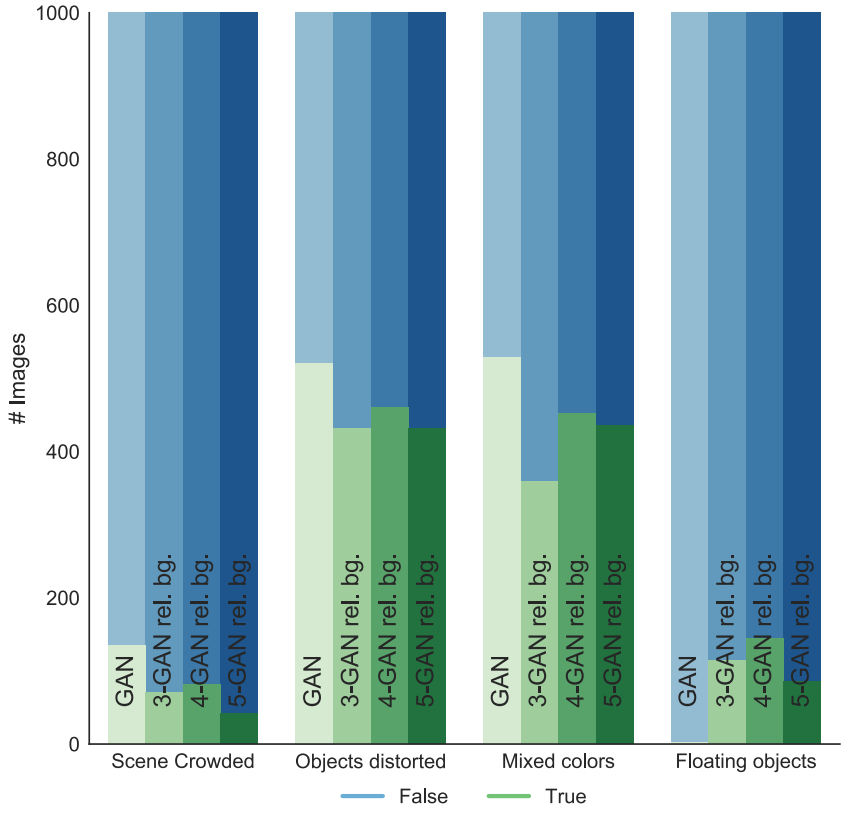


Figure B.5. Additional results of human evaluation. Properties of generated images by *k-GAN* (k = 3,4,5) and *GAN* on CLEVR. Note that on CLEVR all evaluated properties are undesirable, and thus a larger number of "False" responses is better.

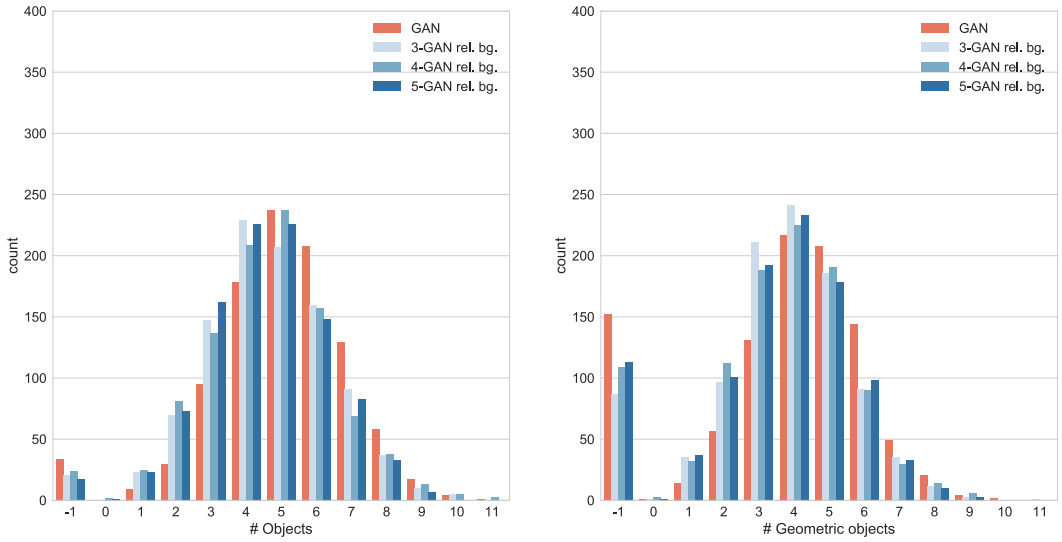


Figure B.6. Additional results of human evaluation. Number of (geometric) objects in generated images by *k-GAN* (k = 3,4,5) and *GAN* on CLEVR. A value of -1 implies a majority vote could not be reached.

B.1.3 Examples of Generated Images

Generated samples (8 × 8 grid) are shown for the best (lowest FID) structured GAN following our grid search, as well as the best baseline GAN for each data set. Real samples from the data set can also be seen.

Independent Multi MNIST

Figures B.7 to B.9.

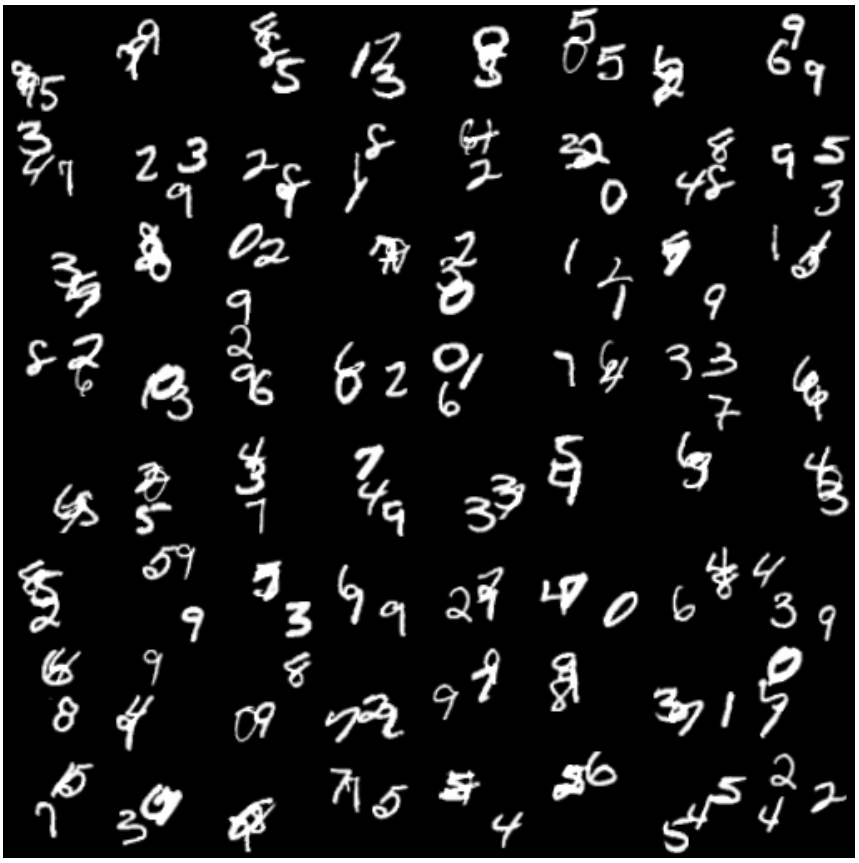


Figure B.7. Real images of Independent MM.

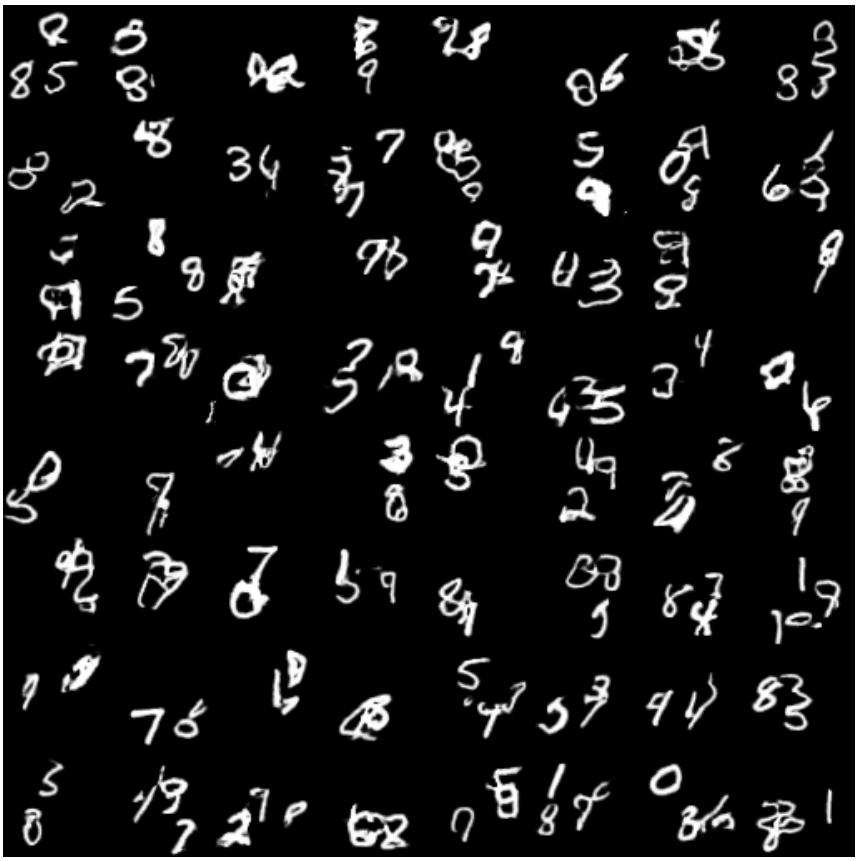


Figure B.8. Generated images by NS-GAN with spectral norm on Independent MM.

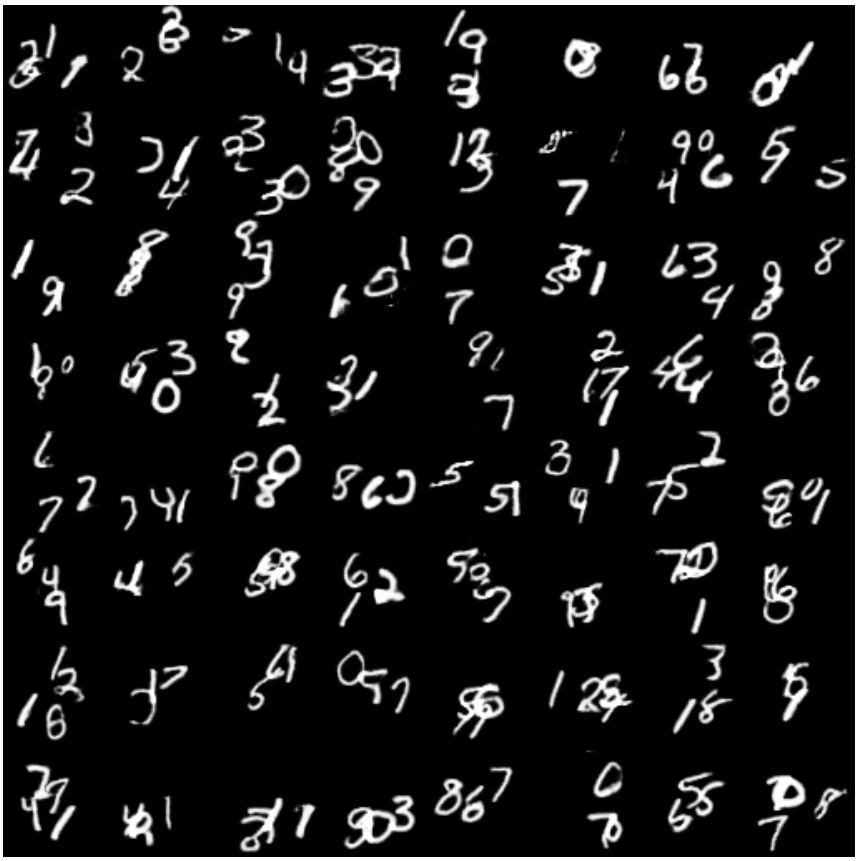


Figure B.9. Generated images by 4-GAN ind. with spectral norm and WGAN penalty on Independent MM.

Triplet Multi MNIST

Figures B.10 to B.12.

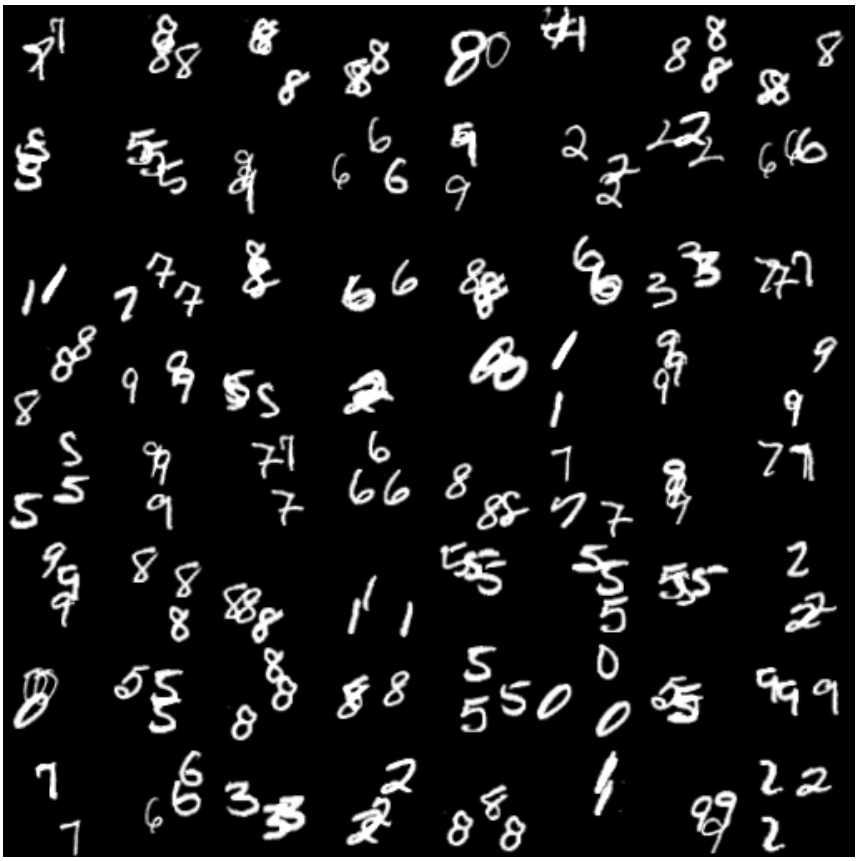


Figure B.10. Real images of Triplet MM.

Figure B.11. Generated images by NS-GAN with spectral norm on Triplet MM.

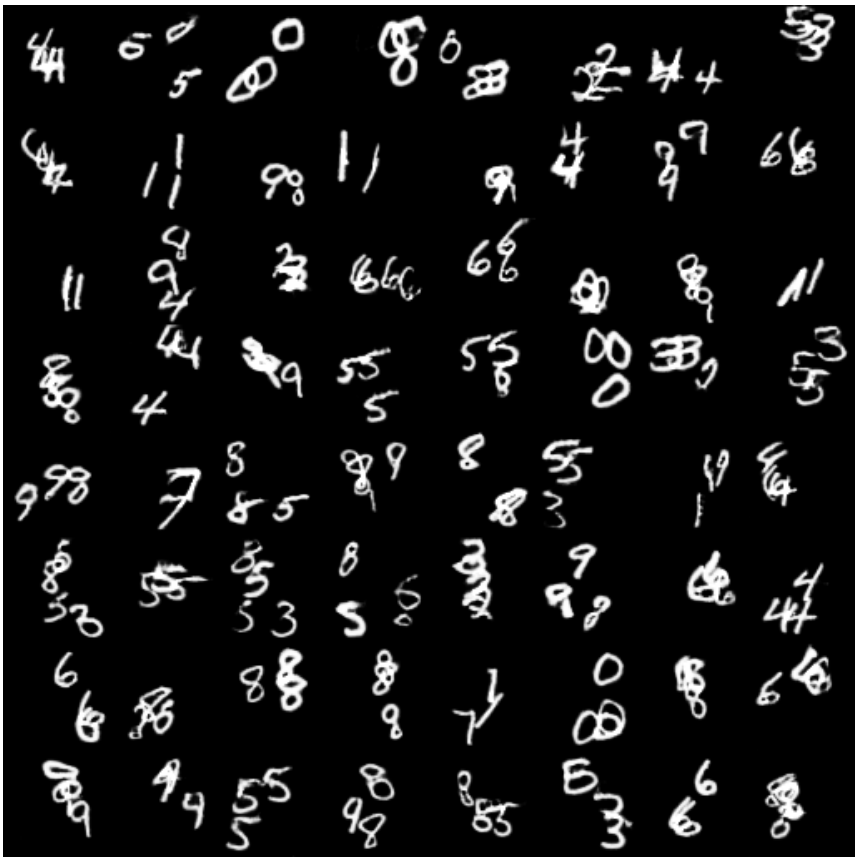


Figure B.12. Generated images by 4-GAN rel. (1 block, 2 heads, no weight sharing) with spectral norm and WGAN penalty on Triplet MM.

RGb-Occluded Multi MNIST

Figures B.13 to B.15.

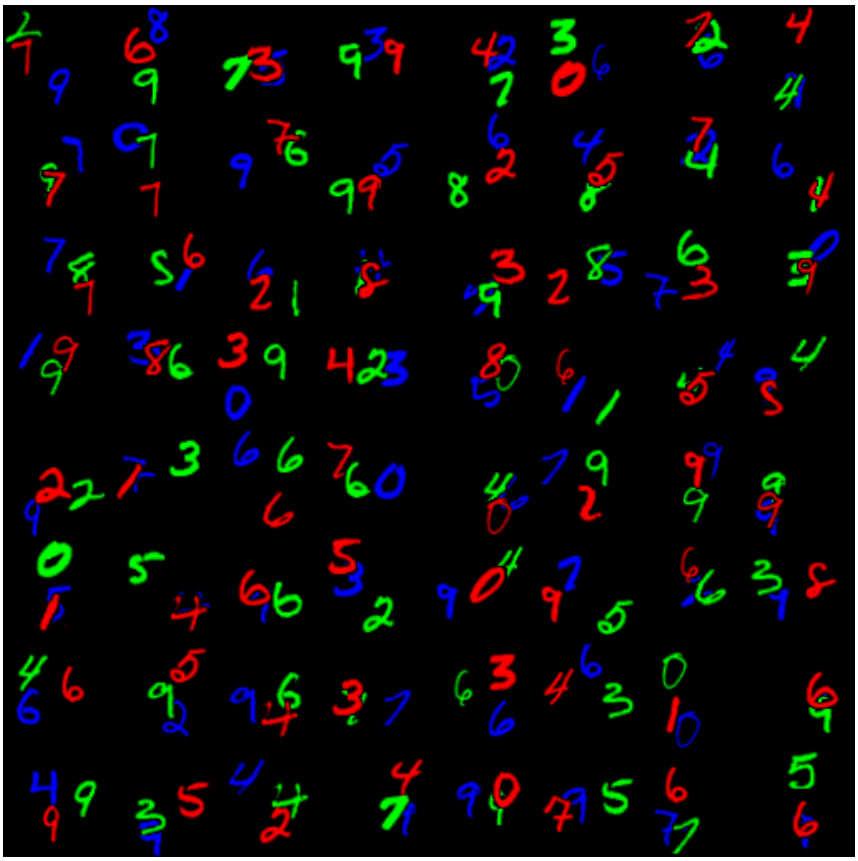


Figure B.13. Real images of RGB Occluded MM.

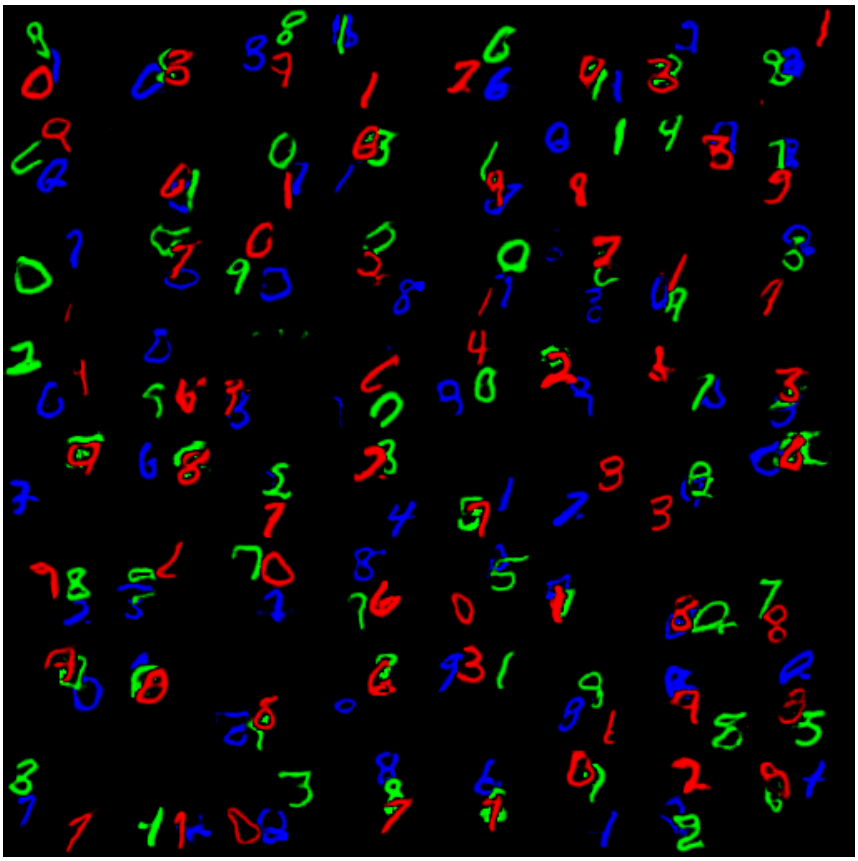


Figure B.14. Generated images by NS-GAN with spectral norm on RGB Occluded MM.

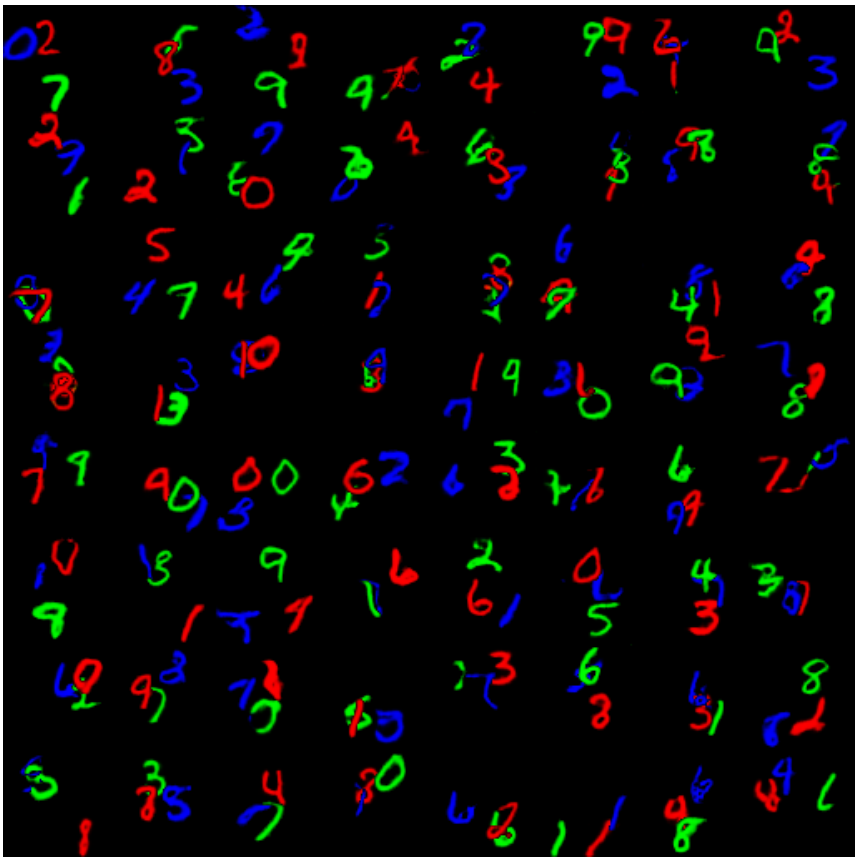


Figure B.15. Generated images by 3-GAN rel. (2 blocks, 2 heads, no weight sharing) with spectral norm and WGAN penalty on RGB Occluded MM.

CIFAR10 + MM

Figures B.16 to B.18.

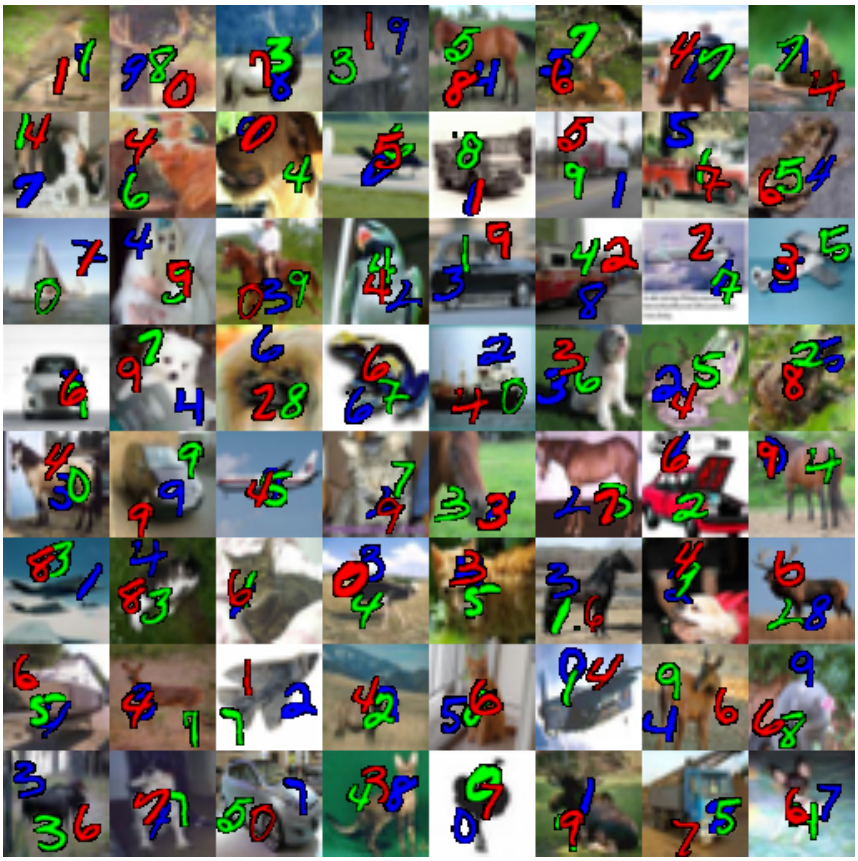


Figure B.16. Real images of CIFAR10 + MM.

Figure B.17. Generated images by WGAN with WGAN penalty on CIFAR10 + MM.

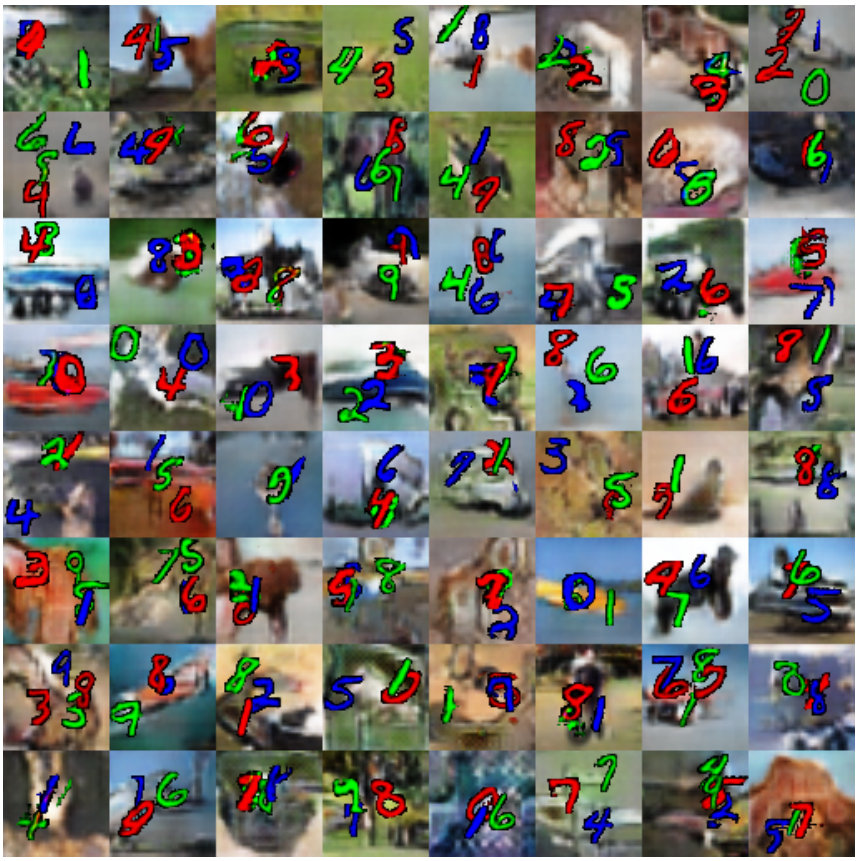


Figure B.18. Generated images by 5-GAN rel. (1 block, 2 heads, no weight sharing) bg. (no interaction) with WGAN penalty on CIFAR10 + MM.

CLEVR

Figures B.19 to B.21.

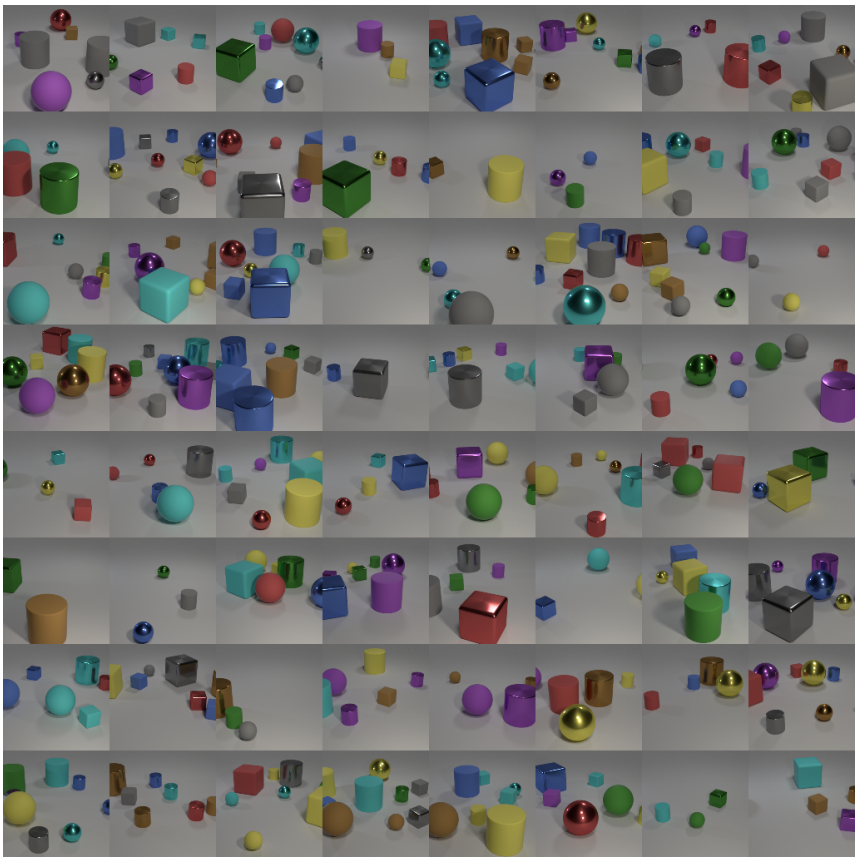


Figure B.19. Real images of CLEVR.

Figure B.20. Generated images by WGAN with WGAN penalty on CLEVR.

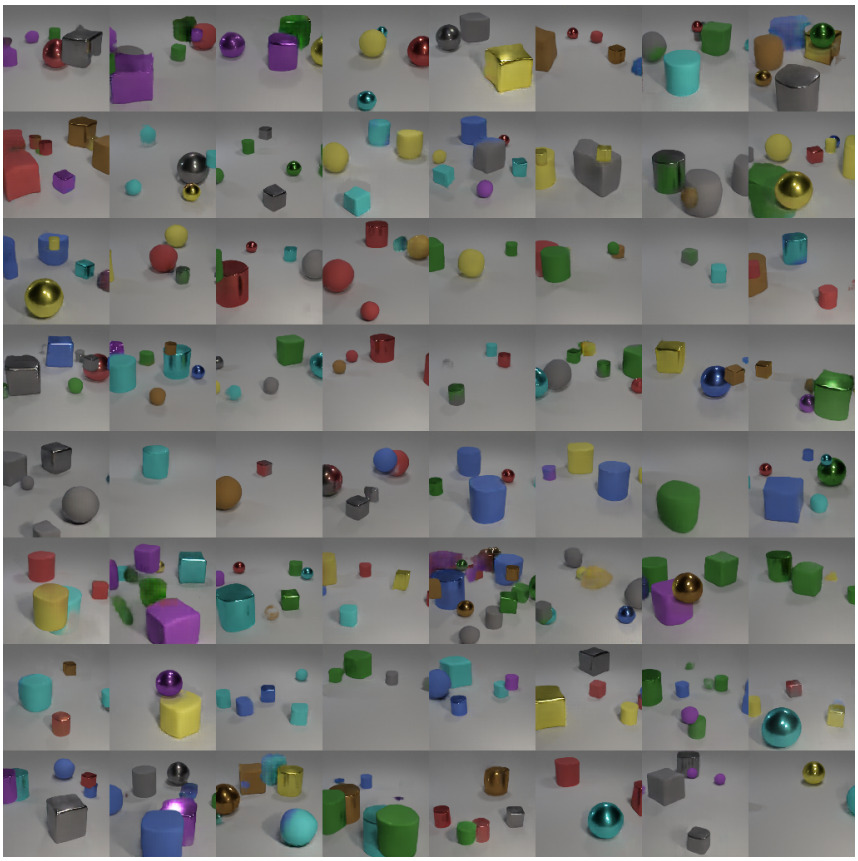


Figure B.21. Generated images by 3-GAN rel. (2 heads, 2 blocks, no weight sharing) bg. (with interaction) with WGAN penalty on CLEVR.

B.2 Evaluating Disentangled Representations

B.2.1 Learning Representations

This subsection contains additional results to evaluate the training phase of the 360 disentanglement models. Figure B.22 displays the rank correlation between the various metrics on the learned representations, and Figures B.23 and B.24 present histograms of the scores assigned by various metrics to the learned representations on *dSprites* and *3dshapes* respectively.

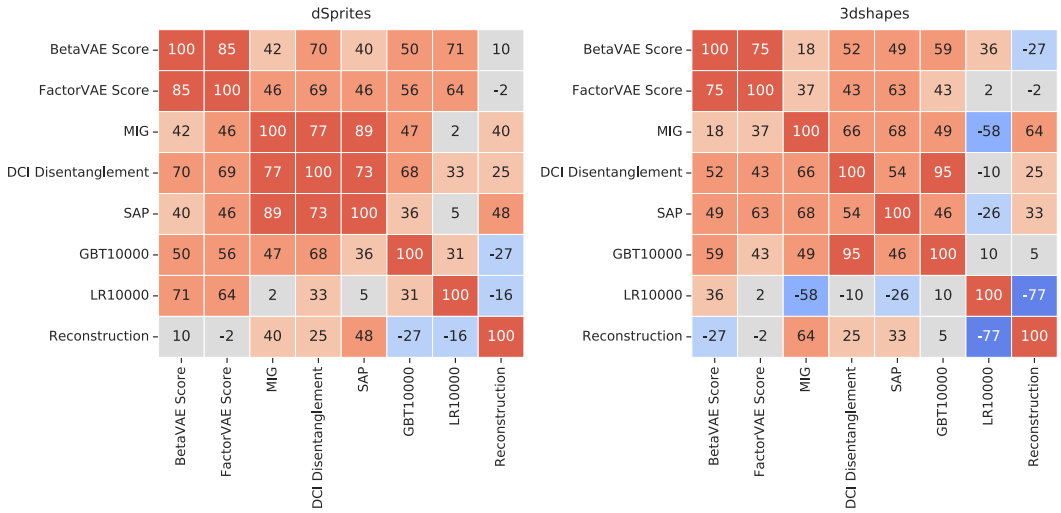


Figure B.22. Rank correlations between the different metrics considered in this paper.

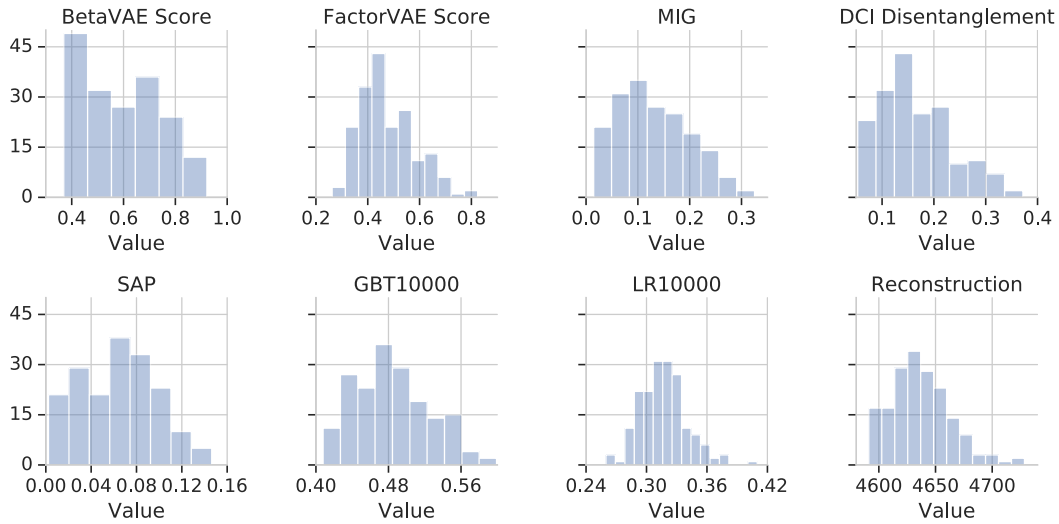


Figure B.23. Distribution of scores assigned by various metrics to the learned representations on *dSprites*.

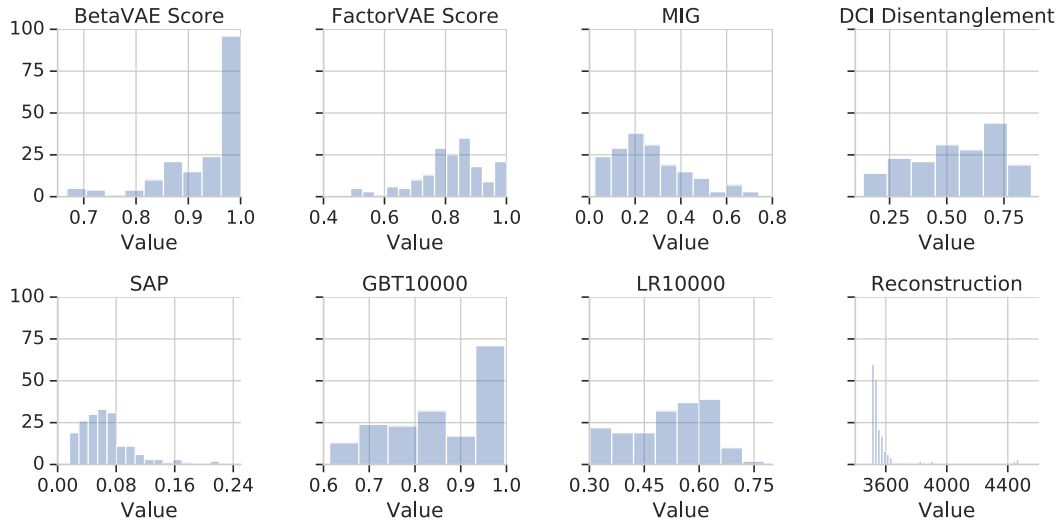


Figure B.24. Distribution of scores assigned by various metrics to the learned representations on *3dshapes*.

B.2.2 Abstract Visual Reasoning

This subsection contains additional results obtained after training 3600 WReN models on the down-stream abstract visual reasoning tasks. Figures B.25 and B.26 provide an in-depth view of the correlation between the scores assigned by various metrics and the down-stream accuracy.

Figures B.27 and B.28 present the down-stream accuracy at various stages of training of models, grouped in quartiles according to the scores assigned by a given metric on *dSprites* and *3dshapes* respectively.

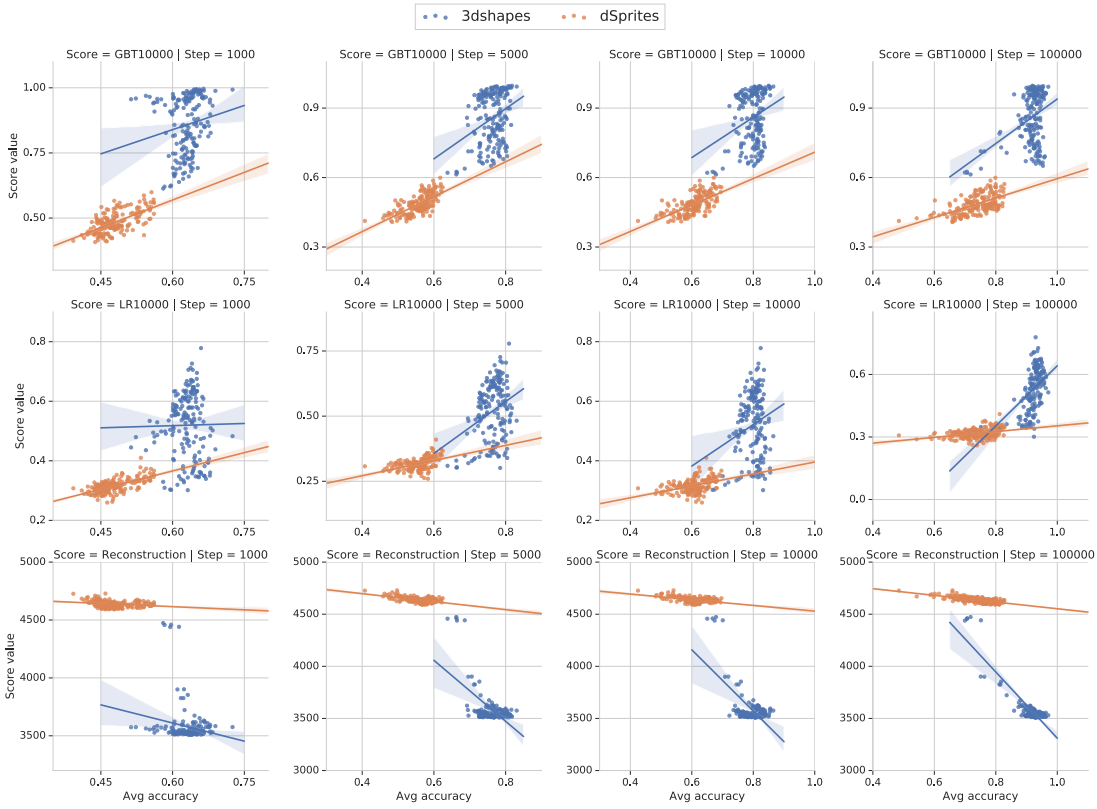


Figure B.25. Correlation between *GBT10000*, *LR10000*, and *Reconstruction* error (rows) and down-stream accuracy of the abstract visual reasoning models. Columns correspond to 1K, 5K, 10K, 100K training steps (i.e. number of samples).

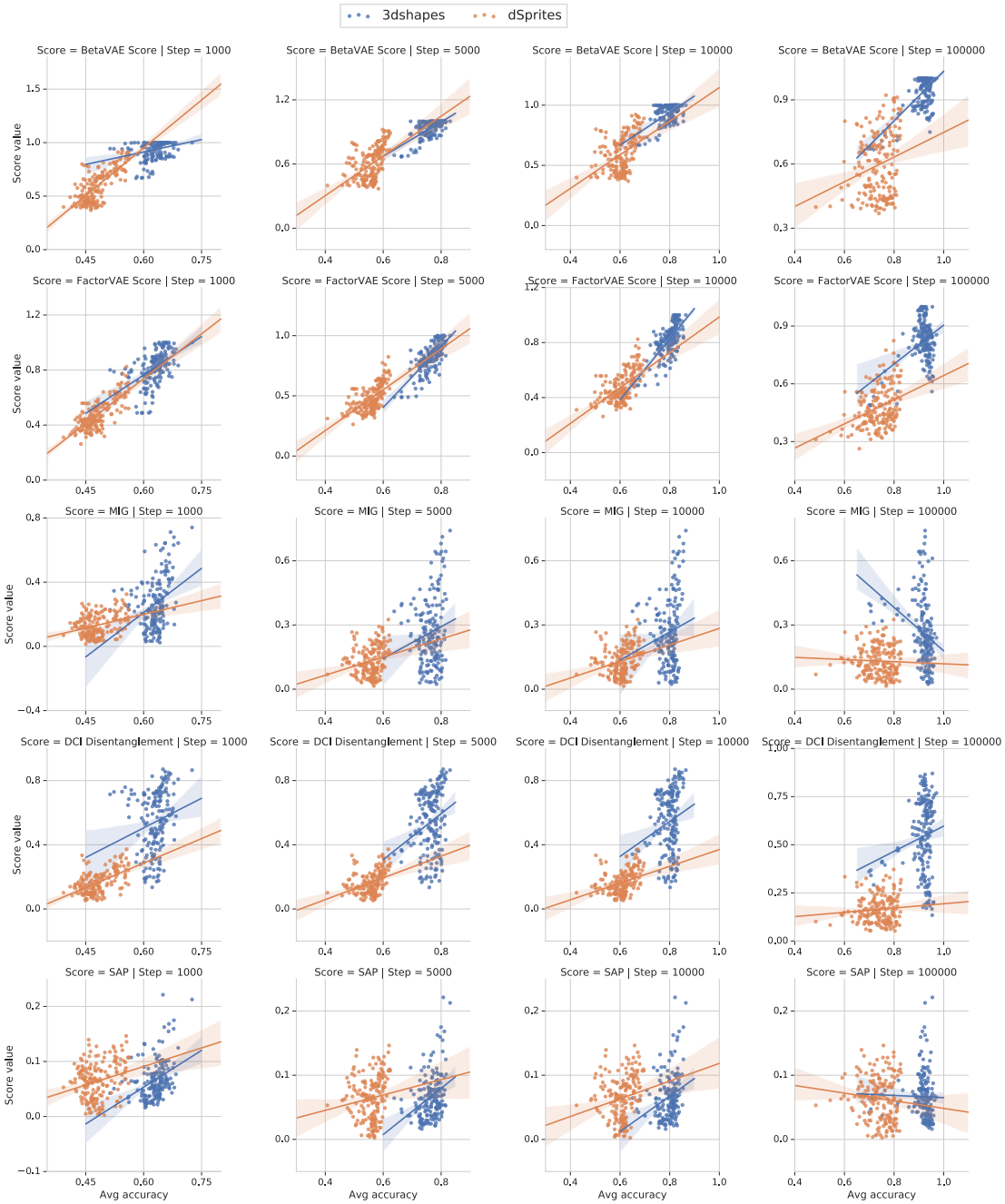


Figure B.26. Correlation between *BetaVAE* score, *FactorVAE* score, *MIG*, *DCI Disentanglement* score, and *SAP* score (rows) and down-stream accuracy of the abstract visual reasoning models. Columns correspond to 1K, 5K, 10K, 100K training steps (i.e. number of samples).

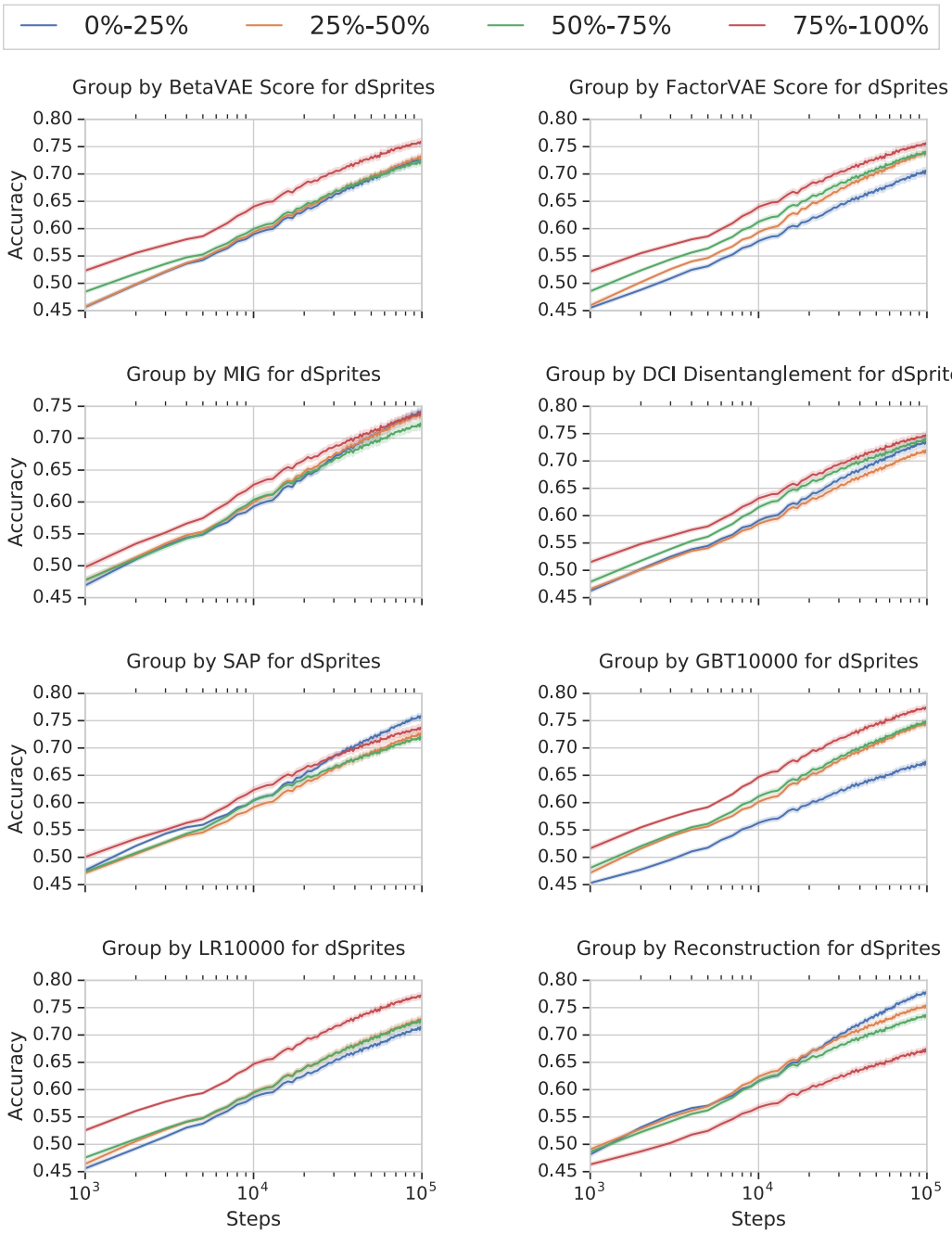


Figure B.27. Down-stream accuracy of abstract visual reasoning models on *dSprites* throughout training (i.e. for different number of samples) binned in quartiles based on different metrics.

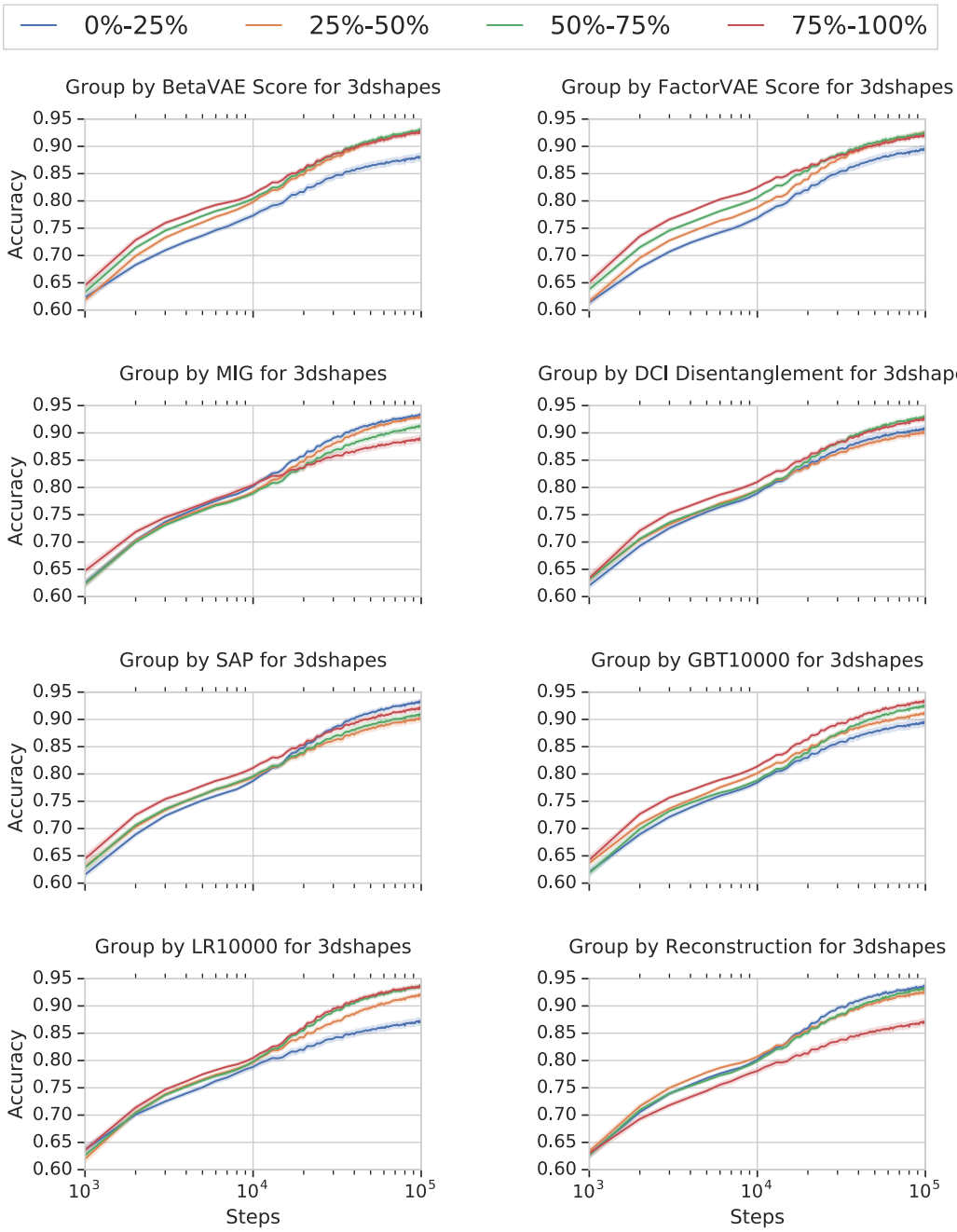


Figure B.28. Down-stream accuracy of abstract visual reasoning models on *3dshapes* throughout training (i.e. for different number of samples) binned in quartiles based on different metrics.

Bibliography

Alessandro Achille, Tom Eccles, Loic Matthey, Chris Burgess, Nicholas Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. In *Advances in Neural Information Processing Systems*, pages 9873–9883, 2018.

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Linda Acredolo and Susan Goodwyn. Symbolic gesturing in normal infants. *Child development*, pages 450–466, 1988.

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

Anurag Ajay, Maria Bauza, Jiajun Wu, Nima Fazeli, Joshua B Tenenbaum, Alberto Rodriguez, and Leslie P Kaelbling. Combining physical simulators and object-based networks for control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3217–3223. IEEE, 2019.

Ferran Alet, Erica Weng, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Neural relational inference with fast modular meta-learning. In *Advances in Neural Information Processing Systems*, pages 11827–11838, 2019.

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On*, pages 73–80. ieeexplore.ieee.org, 2010.

Luis B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings, 1st First International Conference on Neural Networks*, volume 2, pages 609–618. ci.nii.ac.jp, 1987.

George A Alvarez and Steven L Franconeri. How many objects can you track?: Evidence for a resource-limited attentive tracking mechanism. *Journal of vision*, 7(13):14–14, 2007.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks. *Neural Information Processing Systems (NIPS) Workshop on Interpreting, Explaining and Visualizing Deep Learning-Now What?*, 2017.

Jacob Andreas. Measuring compositionality in representation learning. In *International Conference on Learning Representations*, 2019.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

Relja Arandjelović and Andrew Zisserman. Object discovery with a copy-pasting gan. *arXiv preprint arXiv:1905.11369*, 2019.

P Arbeláez, M Maire, C Fowlkes, and J Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

Fred Attneave. Multistability in perception. *Scientific American*, 225(6):62–71, 1971. ISSN 0036-8733(Print). doi: 10.1038/scientificamerican1271-62.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

Yuval Atzmon, Jonathan Berant, Vahid Kezami, Amir Globerson, and Gal Chechik. Learning to generalize to new compositions in image understanding. *arXiv preprint arXiv:1608.07639*, 2016.

Samaneh Azadi, Deepak Pathak, Sayna Ebrahimi, and Trevor Darrell. Compositional gan: Learning image-conditional binary composition. *arXiv preprint arXiv:1807.07560*, 2019.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using Fast Weights to Attend to the Recent Past. In *Advances in Neural Information Processing Systems*, pages 4331–4339, 2016a.

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

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018.

Alan D Baddeley and Graham Hitch. Working memory. In *Psychology of learning and motivation*, volume 8, pages 47–89. Elsevier, 1974.

Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration – A structured survey. *arXiv:cs/0511042*, November 2005.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

Renee Baillargeon, Elizabeth S. Spelke, and Stanley Wasserman. Object permanence in five-month-old infants. *Cognition*, 20(3):191–208, 1985.

Pierre F Baldi and Kurt Hornik. Learning in linear neural networks: A survey. *IEEE Transactions on neural networks*, 6(4):837–858, 1995.

Valentina BamBini, Cristiano Chesi, and Andrea Moro. A conversation with Noam Chomsky: New insights on old foundations. *Phenomenology and Mind*, (3): 166–178, 2012.

Steven C Hayes Dermot Barnes-Holmes and Bryan Roche. *Relational frame theory: A post-Skinnerian account of human language and cognition*. Springer Science & Business Media, 2001.

Brian Barton, Edward F Ester, and Edward Awh. Discrete resource allocation in visual working memory. *Journal of Experimental Psychology: Human Perception and Performance*, 35(5):1359, 2009.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and others. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference On*, pages 693–696. IEEE, 2009.

Suzanna Becker and Geoffrey E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemys\ law Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, and Chris Hesse. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680*, 2019.

Adam Bielski and Paolo Favaro. Emergence of object segmentation in perturbed generative models. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2019.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112 (518):859–877, 2017.

Konstantinos Blekas, Aristidis Likas, Nikolas P Galatsanos, and Isaac E Lagaris. A spatially constrained mixture model for image segmentation. *IEEE transactions on Neural Networks*, 16(2):494–498, 2005.

Daniel G. Bobrow. Natural language input for a computer problem solving system. 1964.

Matko Bošnjak, Tim Rocktäschel, Jason Naradowsky, and Sebastian Riedel. Programming with a differentiable forth interpreter. In *International Conference on Machine Learning*, pages 547–556, 2017.

Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, volume 3176. Springer, 2011.

Jeffrey S. Bowers, Ivan I. Vankov, Markus F. Damian, and Colin J. Davis. Neural networks learn highly selective representations in order to overcome the superposition catastrophe. *Psychological Review*, 121(2):248–261, 2014. ISSN 1939-1471, 0033-295X. doi: 10.1037/a0035943.

Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2019.

John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Neil D. B. Bruce and John K. Tsotsos. Saliency based on information maximization. In *Advances in Neural Information Processing Systems*, pages 155–162, 2006.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *Neural Information Processing Systems (NIPS) Workshop on Learning Disentangled Representations: From Perception to Control*, 2017.

Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv:1901.11390 [cs, stat]*, January 2019.

Nicholas J. Butko and Javier R. Movellan. Optimal scanning for faster object detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2751–2758. ieeexplore.ieee.org, June 2009. doi: 10.1109/CVPR.2009.5206540.

Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. In *International Conference on Learning Representations*, 2017.

Murray Campbell, A. Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

Brian Cantwell-Smith. *On the Origin of Objects*. A Bradford Book. MIT Press, Cambridge, Mass., 1st paperback ed edition, 1998. ISBN 978-0-262-69209-0.

Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2956–2964, 2015.

Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: A hebbian learning rule. *Annual Review of Neuroscience*, 31:25–46, 2008.

George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. In *Advances in Neural Information Processing Systems*, pages 4608–4617, 2019.

Michael B. Chang, Tomer Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations*, 2016.

Nick Chater. Reconciling simplicity and likelihood principles in perceptual organization. *Psychological review*, 103(3):566–581, 1996. doi: 10.1037/0033-295X.103.3.566.

Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In *Advances in Neural Information Processing Systems*, pages 12705–12716, 2019.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in vaes. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

E Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the acoustical society of America*, 25(5):975–979, 1953.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, 2014.

Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *International Joint Conference on Artificial Intelligence*, pages 1237–1242, 2011.

Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International Conference on Machine Learning*, pages 1321–1330, 2019.

Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420, 2019.

Robert Csordas and Juergen Schmidhuber. Improving differentiable neural computers through memory masking, de-allocation, and link distribution sharpness control. In *International Conference on Learning Representations*, 2019.

Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1. ieeexplore.ieee.org, June 2005. doi: 10.1109/CVPR.2005.177.

Sreerupa Das, C. Lee Giles, and Guo-Zheng Sun. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of The Fourteenth Annual Conference of Cognitive Science Society. Indiana University*, page 14, 1992.

Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Luca Del Pero, Joshua Bowdish, Daniel Fried, Bonnie Kermgard, Emily Hartley, and Kobus Barnard. Bayesian geometric modeling of indoor scenes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2719–2726. ieeexplore.ieee.org, June 2012. doi: 10.1109/CVPR.2012.6247994.

Luca Del Pero, Joshua Bowdish, Bonnie Kermgard, Emily Hartley, and Kobus Barnard. Understanding Bayesian rooms using composite 3d object models. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference On*, pages 153–160. cv-foundation.org, 2013.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society.*, pages 1–38, 1977.

Zhiwei Deng, Megha Nawhal, Lili Meng, and Greg Mori. Continuous Graph Flow. *arXiv:1908.02436 [cs, stat]*, September 2019.

Misha Denil, Sergio Gómez Colmenarejo, Serkan Cabi, David Saxton, and Nando de Freitas. Programmable agents. *arXiv preprint arXiv:1706.06383*, 2017.

Barry J Devereux, Lorraine K Tyler, Jeroen Geertzen, and Billi Randall. The centre for speech, language and the brain (cslb) concept property norms. *Behavior research methods*, 46(4):1119–1127, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *Fifth International Conference on Learning Representations*, 2017.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pages 10542–10552, 2019.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *Fifth International Conference on Learning Representations*, 2017.

Leonidas A. A. Doumas, John E. Hummel, and Catherine M. Sandhofer. A theory of the discovery and predication of relational concepts. *Psychological Review*, 115(1):1–43, 2008. ISSN 1939-1471, 0033-295X. doi: 10.1037/0033-295X.115.1.1.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in Neural Information Processing Systems*, pages 1087–1098, 2017.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *Fifth International Conference on Learning Representations*, 2017.

Cian Eastwood and Christopher K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems*, pages 6059–6068, 2018.

Ian Endres and Derek Hoiem. Category independent object proposals. In *Lecture Notes in Computer Science*, pages 575–588. Springer, Berlin, Heidelberg, September 2010. doi: 10.1007/978-3-642-15555-0_42.

Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2019.

James T Enns and Ronald A Rensink. Sensitivity to three-dimensional orientation in visual search. *Psychological Science*, 1(5):323–326, 1990.

SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, N Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2525–2534, 2019.

Jerome Feldman. The neural binding problem (s). *Cognitive neurodynamics*, 7(1): 1–11, 2013.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*, pages 220–229. Springer, 2007.

Jerry A. Fodor. *The Language of Thought*, volume 5. Harvard university press, 1975.

Jerry A. Fodor and Zeno W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, March 1988. ISSN 0010-0277.

Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

Michael Frank, Edward Vul, Vikash Mansinghka, and George Alvarez. What limits performance in multiple object tracking? *Journal of Vision*, 8(6):498–498, 2008.

Noah Frazier-Logue and Stephen José Hanson. Dropout is a special case of the stochastic delta rule: Faster and more accurate deep learning. *arXiv preprint arXiv:1808.03578*, 2018.

Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *UAI'97*, pages 175–181, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

Fabian B. Fuchs, Adam R. Kosiorek, Li Sun, Oiwi Parker Jones, and Ingmar Posner. End-to-end Recurrent Multi-Object Tracking and Trajectory Prediction with Relational Reasoning. *arXiv preprint arXiv:1907.12887*, 2019.

Keisuke Fukuda, Edward Awh, and Edward K. Vogel. Discrete capacity limits in visual working memory. *Curr Opin Neurobiol*, 20(2):177–182, April 2010. ISSN 0959-4388. doi: 10.1016/j.conb.2010.03.005.

Kunihiko Fukushima. Neural network model for a mechanism of pattern recognition unaffected by shift in position — Neocognitron. *Trans. IECE*, J62-A(10): 658–665, 1979.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International Conference on Machine Learning*, pages 2063–2072, 2019.

Dashan Gao and Nuno Vasconcelos. Discriminant saliency for visual recognition from cluttered scenes. In *Advances in Neural Information Processing Systems*, pages 481–488, 2005.

Ross W. Gayler. Multiplicative binding, representation operators & analogy. *Cogprints*, 1998.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 2, pages 850–855, 1999.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

C. Lee Giles, Guo-Zheng Sun, Hsing-Hen Chen, Yee-Chun Lee, and Dong Chen. Higher order recurrent networks and grammatical inference. In *Advances in Neural Information Processing Systems*, pages 380–387, 1990.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272, 2017.

Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587. cv-foundation.org, 2014.

Roy J Glauber. Time-dependent statistics of the ising model. *Journal of mathematical physics*, 4(2):294–307, 1963.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016. ISBN 978-0-262-03561-3.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2. ieeexplore.ieee.org, July 2005. doi: 10.1109/IJCNN.2005.1555942.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538 (7626):471–476, October 2016. ISSN 0028-0836. doi: 10.1038/nature20101.

Edwin James Green. A theory of perceptual objects. *Philos. Phenomenol. Res.*, 106:7345, June 2018. ISSN 0031-8205. doi: 10.1111/phpr.12521.

Klaus Greff*, Sjoerd van Steenkiste*, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6694–6704, 2017a.

Klaus Greff*, Sjoerd van Steenkiste*, and Jürgen Schmidhuber. Neural expectation maximization. *International Conference on Learning Representations, Workshop Track*, 2017b.

Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. Binding via reconstruction clustering. *International Conference on Learning Representations, Workshop Track*, 2015.

Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Juergen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems*, pages 4484–4492, 2016.

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

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *In preparation*, 2020.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, pages 399–406, 2010.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pages 1462–1471, 2015.

Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20. ACM, 1998.

Jose A. Guerrero-Colón, Eero P. Simoncelli, and Javier Portilla. Image denoising using mixtures of Gaussian scale mixtures. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference On*, pages 565–568. IEEE, 2008.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. In *Advances in Neural Information Processing Systems*, pages 8388–8399, 2018.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

Jessica B. Hamrick, Kelsey R. Allen, Victor Bapst, Tina Zhu, Kevin R. McKee, Joshua B. Tenenbaum, and Peter W. Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Stephen José Hanson. A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena*, 42(1-3):265–272, 1990.

Stevan Harnad. The symbol grounding problem. *Physica D*, 42(1):335–346, June 1990. ISSN 0167-2789. doi: 10.1016/0167-2789(90)90087-6.

Hedi Harzallah, Frédéric Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 237–244. ieeexplore.ieee.org, 2009. doi: 10.1109/ICCV.2009.5459257.

Gary Hatfield and William Epstein. The status of the minimum principle in the theoretical analysis of visual perception. *Psychological Bulletin*, 97(2):155, 1985.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep Convolutional Networks on Graph-Structured Data. *arXiv:1506.05163 [cs]*, June 2015.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. In *International Conference on Learning Representations*, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490, 2017b.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018a.

Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matko Bošnjak, Murray Shanahan, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner. Scan: Learning hierarchical compositional visual concepts. In *International Conference on Learning Representations*, 2018b.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, 2011.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Generating multiple objects at spatially distinct locations. In *International Conference on Learning Representations*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. In *Proc. International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering occlusion boundaries from an image. *Int. J. Comput. Vis.*, 91(3):328–346, February 2011. ISSN 0920-5691. doi: 10.1007/s11263-010-0400-4.

Ha Hong, Ethan Solomon, Dan Yamins, and James J DiCarlo. Large-scale characterization of a universal and compact visual perceptual space. *space (P-space)*, 10(20):30, 2014.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8): 2554–2558, 1982.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Patrik O. Hoyer, Dominik Janzing, Joris M. Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems*, pages 689–696, 2009.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pages 1878–1889, 2017.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*, 2018.

Sean Joseph Hughes and Patrick Michael Dermot Barnes-Holmes. Relational frame theory: the basic account. In *The Wiley handbook of contextual behavioral science*, pages 129–178. Wiley-Blackwell, 2016.

John E. Hummel and Keith J. Holyoak. Distributing structure over time. *Behavioral and Brain Sciences*, 16(3):464–464, September 1993. ISSN 1469-1825, 0140-525X. doi: 10.1017/S0140525X00031083.

John E. Hummel, Keith J. Holyoak, Collin Green, Leonidas AA Doumas, Derek Devnich, Aniket Kittur, and Donald J. Kalar. A solution to the binding problem for compositional connectionism. In *Compositional Connectionism in Cognitive Science: Papers from the AAAI Fall Symposium, Ed. SD Levy & R. Gayler*, pages 31–34, 2004.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. The compositionality of neural networks: Integrating symbolism and connectionism. *arXiv preprint arXiv:1908.08351*, 2019.

Marcus Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media, 2004.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

Aapo Hyvärinen and Jukka Perkiö. Learning to Segment Any Random Vector. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 4167–4172. IEEE, 2006.

Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Crisp boundary detection using pointwise mutual information. In *Lecture Notes in Computer Science*, pages 799–814. Springer, Cham, September 2014. doi: 10.1007/978-3-319-10578-9_52.

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Learning visual groups from co-occurrences in space and time. *arXiv:1511. 06811 [cs]*, November 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE, 2017.

Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20 (11):1254–1259, November 1998. ISSN 0162-8828. doi: 10.1109/34.730558.

Michael Iuzzolino, Yoram Singer, and Michael C. Mozer. Convolutional bipartite attractor networks. *arXiv:1906.03504 [cs, stat]*, June 2019.

Alekseĭ Grigor'evich Ivakhnenko and Valentin Grigorévich Lapa. *Cybernetic Predicting Devices*. CCM Information Corporation, 1965.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems 28*, pages 2017–2025, 2015.

Anil K. Jain, Richard C. Dubes, et al. *Algorithms for Clustering Data*, volume 6. Prentice hall Englewood Cliffs, NJ, 1988.

Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in Neural Information Processing Systems*, pages 4016–4027, 2018.

Michael Janner, Sergey Levine, William T. Freeman, Joshua B. Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-centric models. In *International Conference on Learning Representations*, 2019.

Allan D. Jepson and Michael J. Black. Mixture models for optical flow computation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761. ieeexplore.ieee.org, June 1993. doi: 10.1109/CVPR.1993.341161.

Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information distillation for unsupervised image segmentation and clustering. *arXiv preprint arXiv:1807.06653*, 2(3):8, 2018.

Jindong Jiang, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2020.

Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017a.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998, 2017b.

Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1219–1228, 2018.

Philip N. Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250, 2010.

Nebojsa Jojic and Brendan J. Frey. Learning flexible sprites in video layers. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference On*, volume 1, pages I–I. IEEE, 2001.

Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, pages 190–198, 2015.

Leslie Pack Kaelbling. *Learning in embedded systems*. MIT press, 1993.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Lukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. In *International Conference on Learning Representations*, 2016.

Pentti Kanerva. Binary spatter-coding of ordered K-tuples. In *Artificial Neural Networks — ICANN 96*, pages 869–873. Springer Berlin Heidelberg, 1996. doi: 10.1007/3-540-61510-5_146.

Anitha Kannan, John Winn, and Carsten Rother. Clustering appearance and shape by learning jigsaws. In *Advances in Neural Information Processing Systems*, pages 657–664, 2007.

Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *International Conference on Machine Learning*, pages 1809–1818, 2017.

Theofanis Karaletsos, Serge Belongie, and Gunnar Rätsch. Bayesian representation learning with oracle constraints. In *International Conference on Learning Representations*, 2016.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv:1506.02078 [cs]*, November 2015.

Matthew A Kelly, Dorothea Blostein, and Douglas JK Mewhort. Encoding structure in holographic reduced representations. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 67(2):79–93, 2013.

Charles Kemp and Joshua B Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008.

Richard Kempter, Wulfram Gerstner, and J Leo Van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498, 1999.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2020.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2654–2663, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697, 2018.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

SC Kleene. Representations of events in nerve nets and finite automata. *Automata Studies [Annals of Math. Studies 34]*, 1956.

Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer, third edition, 1989. ISBN 978-0-387-18314-5.

Shu Kong and Charless C. Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9018–9028, 2018.

Alfred Korzybski. *Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics*. Institute of GS, 1958.

Adam Kosiorek, Alex Bewley, and Ingmar Posner. Hierarchical attentive recurrent tracking. In *Advances in Neural Information Processing Systems*, pages 3053–3061, 2017.

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pages 8606–8616, 2018.

Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E. Hinton. Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*, pages 15486–15496, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

Tejas D. Kulkarni, Pushmeet Kohli, Joshua B. Tenenbaum, and Vikash K. Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition*, pages 4390–4399. openaccess.thecvf.com, 2015a.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, 2015b.

Tejas D. Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *Advances in Neural Information Processing Systems*, pages 10723–10733, 2019.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.

Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural Random-Access Machines. In *International Conference on Learning Representations*, 2016.

Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in gans. In *International Conference on Machine Learning*, pages 3581–3590, 2019.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954, 2017.

Hanock Kwak and Byoung-Tak Zhang. Generating images part by part with composite generative adversarial networks. *arXiv preprint arXiv:1607.05387*, 2016.

Brenden Lake and Marco Baroni. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In *International Conference on Machine Learning*, pages 2873–2882, 2018.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. ieeexplore.ieee.org, June 2008. doi: 10.1109/CVPR.2008.4587586.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pages 609–616, 2009.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743, 2019.

Te-Won Lee and Michael S Lewicki. Unsupervised image classification, segmentation, and enhancement using ica mixture models. *IEEE Transactions on Image Processing*, 11(3):270–279, 2002.

Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. In *International Conference on Machine Learning*, pages 430–438, 2016.

Chengtao Li, Daniel Tarlow, Alexander L Gaunt, Marc Brockschmidt, and Nate Kushman. Neural program lattices. In *International Conference on Learning Representations*, 2017.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *In Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K. Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, pages 4257–4267, 2019.

Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9455–9464, 2018.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable Shape Completion with Graph Convolutional Autoencoders. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1886–1895, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00202.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph Normalizing Flows. In *Advances in Neural Information Processing Systems*, pages 13578–13588, 2019a.

Zhijian Liu, Jiajun Wu, Zhenjia Xu, Chen Sun, Kevin Murphy, William T. Freeman, and Joshua B. Tenenbaum. Modeling parts, structure, and system dynamics via predictive learning. In *International Conference on Learning Representations*, 2019b.

Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, 2018.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, 2020.

Romain Lopez, Jeffrey Regier, Michael I Jordan, and Nir Yosef. Information constraints on auto-encoding variational bayes. In *Advances in Neural Information Processing Systems*, pages 6114–6125, 2018.

David Lopez-Paz, Krikamol Muandet, Bernhard Schölkopf, and Iliya Tolstikhin. Towards a learning theory of cause-effect inference. In *International Conference on Machine Learning*, pages 1452–1461, 2015.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations*, 2017.

João Loula, Marco Baroni, and Brenden Lake. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, 2018.

David G Lowe. Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, 3(2):119–130, 1989.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems*, pages 700–709, 2018.

Steven J Luck and Edward K Vogel. The capacity of visual working memory for features and conjunctions. *Nature*, 390(6657):279–281, 1997.

Zhaoliang Lun, Changqing Zou, Haibin Huang, Evangelos Kalogerakis, Ping Tan, Marie-Paule Cani, and Hao Zhang. Learning to group discrete graphical patterns. *ACM Trans. Graph.*, 36(6):225:1–225:11, November 2017. ISSN 0730-0301. doi: 10.1145/3130800.3130841.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. *International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *Int. J. Comput. Vis.*, 43(1):7–27, June 2001. ISSN 0920-5691. doi: 10.1023/A:1011174803800.

Vikash K. Mansinghka, Tejas D. Kulkarni, Yura N. Perov, and Joshua B. Tenenbaum. Approximate Bayesian image interpretation using generative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528, 2013.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*, 2019.

Gary F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT press, 2003. ISBN 978-0-262-63268-3.

Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In Jennifer Dy and Andreas Krause, editors, *Proceedings of Machine Learning Research*, volume 80, pages 3403–3412, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Kenneth McGarry, Stefan Wermter, and John MacIntyre. Hybrid neural systems: from simple coupling to fully integrated neural networks. *Neural Computing Surveys*, 2(1):62–93, 1999.

Harry Mcgurk and John Macdonald. Hearing lips and seeing voices. *Nature*, 264 (5588):746, December 1976. ISSN 1476-4687. doi: 10.1038/264746a0.

Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent "grammar cells". In *Advances in Neural Information Processing Systems*, pages 1925–1933, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*, 2013.

George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

Peter M Milner. A model for visual shape recognition. *Psychological review*, 81(6): 521, 1974.

Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P. Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. In *Advances in Neural Information Processing Systems*, pages 92–102, 2019.

Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369, 2001.

Thomas M Mitchell et al. Machine learning, 1997.

Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems 27*, pages 2204–2212, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Instagan: Instance-aware image-to-image translation. In *International Conference on Learning Representations*, 2018.

Hossein Mobahi, Shankar R Rao, Allen Y Yang, Shankar S Sastry, and Yi Ma. Segmentation of natural images by texture and boundary compression. *International journal of computer vision*, 95(1):86–98, 2011.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *International Conference on Learning Representations, Workshop Track*, 2017.

Igor Mordatch. Concept learning with energy-based models. *International Conference on Learning Representations, Workshop Track*, 2018.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards Interpretable Reinforcement Learning Using Attention Augmented Agents. In *Advances in Neural Information Processing Systems*, pages 12350–12359, 2019.

Michael C. Mozer. Types and tokens in visual letter perception. *Journal of experimental psychology: Human perception and performance*, 15(2):287–303, 1989. doi: https://psycnet.apa.org/doi/10.1037/0096-1523.15.2.287.

Michael C. Mozer and Sreerupa Das. A connectionist symbol manipulator that discovers the structure of context-free languages. In *Advances in Neural Information Processing Systems*, pages 863–870, 1993.

Michael C. Mozer, Richard S. Zemel, and Marlene Behrmann. Learning to segment images using dynamic feature binding. In *Advances in Neural Information Processing Systems*, pages 436–443, 1992.

Michael C Mozer, Denis Kazakov, and Robert V Lindsey. State-denoised recurrent neural networks. *arXiv preprint arXiv:1805.08394*, 2018.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. In *Advances in neural information processing systems*, pages 8799–8810, 2018.

Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. In *Proceedings of the Conference. Association for Computational Linguistics. Meeting*, volume 1, page 397. NIH Public Access, 2017.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Charlie Nash, SM Ali Eslami, Chris Burgess, Irina Higgins, Daniel Zoran, Theophane Weber, and Peter Battaglia. The multi-entity variational autoencoder. *Neural Information Processing Systems (NIPS) Workshop on Learning Disentangled Representations: from Perception to Control*, 2017.

Aran Nayebi, Daniel Bear, Jonas Kubilius, Kohitij Kar, Surya Ganguli, David Sussillo, James J DiCarlo, and Daniel L Yamins. Task-driven convolutional recurrent models of the visual system. In *Advances in Neural Information Processing Systems*, pages 5290–5301, 2018.

C Rodrigues Neto and JF Fontanari. Multivalley structure of attractor neural networks. *Journal of Physics A: Mathematical and General*, 30(22):7945, 1997.

Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: Symbols and search. *Mind design*, page 4l, 1981.

Allen Newell, John C. Shaw, and Herbert A. Simon. Report on a general problem solving program. In *IFIP Congress*, volume 256, page 64. Pittsburgh, PA, 1959.

Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7588–7597, 2019.

Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images. *arXiv preprint arXiv:2002.08988*, 2020.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, pages 6338–6347, 2017.

Nils J Nillson. Principles of artificial intelligence. *Tioga, Palo Alto, CA*, 1980.

Malvina Nissim, Rik van Noord, and Rob van der Goot. Fair is Better than Sensational:Man is to Doctor as Woman is to Doctor. *arXiv:1905.09866 [cs]*, May 2019.

Dimitri Nowicki and Hava T. Siegelmann. Flexible kernel memory. *PLoS One*, 5(6): e10955, June 2010. ISSN 1932-6203. doi: 10.1371/journal.pone.0010955.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and Checkerboard Artifacts. *Distill*, 2016. doi: 10.23915/distill.00003.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature Visualization. *Distill*, 2(11):e7, November 2017. ISSN 2476-0757. doi: 10.23915/distill.00007.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom In: An Introduction to Circuits. *Distill*, 5(3):e00024.001, March 2020. ISSN 2476-0757. doi: 10.23915/distill.00024.001.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583): 607–609, June 1996. ISSN 0028-0836. doi: 10.1038/381607a0.

Randall C O'reilly and Richard S Busby. Generalizable relational binding from coarse-coded distributed representations. In *Advances in neural information processing systems*, pages 75–82, 2002.

Lucas Paletta, Gerald Fritz, and Christin Seifert. Q-learning of sequential attention for visual object recognition from informative local descriptors. In *International Conference on Machine Learning*, pages 649–656, 2005.

Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *Advances in Neural Information Processing Systems*, pages 3368–3378, 2018.

Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017.

Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: Identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT press, 2017.

Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229–2232, 1987.

Tony A Plate. Holographic reduced representations. *IEEE Transactions on Neural networks*, 6(3):623–641, 1995.

Jordan B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46 (1-2):77–105, 1990.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5): 1–17, 1964.

Isabeau Prémont-Schwarz, Alexander Ilin, Tele Hao, Antti Rasmus, Rinu Boney, and Harri Valpola. Recurrent ladder networks. In *Advances in neural information processing systems*, pages 6009–6019, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

R. A. Rao, G. Cecchi, C. C. Peck, and J. R. Kozloski. Unsupervised segmentation with dynamical units. *Neural Networks, IEEE Transactions on*, 19(1):168–182, 2008.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015.

John C Raven. Standardization of progressive matrices, 1938. *British Journal of Medical Psychology*, 19(1):137–150, 1941.

Scott Reed and Nando de Freitas. Neural programmer-interpreters. In *International Conference on Learning Representations*, November 2015.

David P. Reichert and Thomas Serre. Neuronal Synchrony in Complex-Valued Deep Networks. *arXiv preprint arXiv:1312.6115*, 2013.

Mengye Ren and Richard S. Zemel. End-to-End Instance Segmentation with Recurrent Attention. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 293–301, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.39.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 10–17 vol.1. ieeexplore.ieee.org, October 2003. doi: 10.1109/ICCV.2003.1238308.

Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. In *Advances in Neural Information Processing Systems*, pages 185–194, 2018.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961.

Adina L. Roskies. The binding problem. *Neuron*, 24(1):7–9, 111–25, September 1999. ISSN 0896-6273.

Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20 (1):23–38, 1998.

Christopher J. Rozell, Don H. Johnson, Richard G. Baraniuk, and Bruno A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial Intelligence: a Modern Approach*, volume 2. Englewood Cliffs: Prentice Hall, 1995.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

Ramin Samadani. A finite mixtures algorithm for finding proportions in sar images. *IEEE Transactions on Image Processing*, 4(8):1182–1186, 1995.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, pages 4967–4976, 2017.

Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 7299–7310, 2018a.

Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pages 4477–4486, 2018b.

E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7(1):51–71, 1995.

F Scarselli, M Gori, Ah Chung Tsoi, M Hagenbuchner, and G Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 1(20): 61–80, 2009.

Imanol Schlag and Jürgen Schmidhuber. Learning to reason with third order tensor products. In *Advances in Neural Information Processing Systems*, pages 9981–9993, 2018.

Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611*, 2019.

J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992a.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015a. doi: 10.1016/j.neunet.2014.09.003. Published online 2014; 888 references; based on TR arXiv:1404.7828 [cs.NE].

J. Schmidhuber, M. Eldracher, and B. Foltin. Semilinear predictability minimization produces well-known feature detectors. *Neural Computation*, 8(4): 773–786, 1996.

Juergen Schmidhuber and Rudolf Huber. Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2 (01n02):125–134, 1991.

Jürgen Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90, Institut für Informatik, Technische Universität München, 1990.

Jürgen Schmidhuber. Neural sequence chunkers. Technical report, Institut für Informatik, Technische Universität München, April 1991a.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991b.

Jürgen Schmidhuber. Learning Factorial Codes by Predictability Minimization. *Neural Computation*, 4(6):863–879, November 1992b. ISSN 0899-7667. doi: 10.1162/neco.1992.4.6.863.

Jürgen Schmidhuber. Learning Complex, Extended Sequences Using the Principle of History Compression. *Neural Computation*, 4(2):234–242, March 1992c. ISSN 0899-7667. doi: 10.1162/neco.1992.4.2.234.

Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*, 2015b.

Jürgen Schmidhuber. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Networks*, 2020.

Bernhard Schölkopf. Causality for Machine Learning. *arXiv preprint arXiv:1911.10500*, 2019.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. In *International Conference on Machine Learning*, pages 459–466, 2012.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Edward H. Shortliffe, Randall Davis, Stanton G. Axline, Bruce G. Buchanan, C. Cordell Green, and Stanley N. Cohen. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system. *Computers and biomedical research*, 8(4):303–320, 1975.

Hava T Siegelmann and Eduardo D Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, and Thore Graepel. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Wolf Singer. Neuronal synchrony: A versatile code for the definition of relations? *Neuron*, 24(1):49–65, 111–25, September 1999. ISSN 0896-6273.

Wolf Singer. Consciousness and the binding problem. *Annals of the New York Academy of Sciences*, 929(1):123–146, 2001.

Wolf Singer. Distributed processing and temporal codes in neuronal networks. *Cognitive neurodynamics*, 3(3):189–196, 2009.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2): 159–216, 1990.

Elliot Sollow. Assessing the Maintainability of XCQN-in-RIME: Coping with the Problems of a VERY Large Rule-Base. 1987.

C Spampinato, S Palazzo, P D'Oro, D Giordano, and M Shah. Adversarial framework for unsupervised learning of motion dynamics in videos. *International Journal of Computer Vision*, pages 1–20, 2019.

Elizabeth S. Spelke. Principles of object perception. *Cognitive science*, 14(1): 29–56, 1990.

Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

Alessandro Sperduti. Encoding labeled graphs by labeling raam. In *Advances in Neural Information Processing Systems*, pages 1125–1132, 1994.

Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

Alessandro Sperduti, Antonina Starita, and Christoph Goller. Learning distributed representations for the classification of terms. In *IJCAI*, pages 509–517. Citeseer, 1995.

Pablo Sprechmann, Alexander M. Bronstein, and Guillermo Sapiro. Learning efficient sparse and low rank models. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1821–1833, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015.

Kenneth O Stanley and Risto Miikkulainen. Evolving a roving eye for go. In *Genetic and Evolutionary Computation Conference*, pages 1226–1238. Springer, 2004.

Xander Steenbrugge, Sam Leroux, Tim Verbelen, and Bart Dhoedt. Improving generalization for abstract reasoning tasks using disentangled feature representations. *Neural Information Processing Systems (NeurIPS) Workshop on Relational Representation Learning, Montréal, Canada.*, 2018.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448, 2015.

Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. In *International Conference on Learning Representations*, 2019.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328, 2017.

Raphael Suter, Djordje Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065, 2019.

Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2009.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.

Yichuan Tang, Nitish Srivastava, and Ruslan R. Salakhutdinov. Learning generative models with visual attention. In *Advances in Neural Information Processing Systems*, pages 1808–1816, 2014.

Anne Treisman. The binding problem. *Current opinion in neurobiology*, 6(2): 171–178, 1996.

Anne Treisman. Solutions to the binding problem: Progress through controversy and convergence. *Neuron*, 24(1):105–10, 111–25, September 1999. ISSN 0896-6273.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *Neural Information Processing Systems (NeurIPS) Workshop on Bayesian Deep Learning, Montreal, Canada.*, 2018.

Pedro A. Tsividis, Thomas Pouncy, Jaqueline L. Xu, Joshua B. Tenenbaum, and Samuel J. Gershman. Human learning in Atari. In *2017 AAAI Spring Symposium Series*, March 2017.

Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):657–673, May 2002. ISSN 0162-8828. doi: 10.1109/34.1000239.

Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of computer vision*, 63(2):113–140, 2005.

Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

Tomer D. Ullman, Elizabeth Spelke, Peter Battaglia, and Joshua B. Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in Cognitive Sciences*, 21(9):649–665, 2017.

Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

Sjoerd van Steenkiste*, Klaus Greff*, and Jürgen Schmidhuber. A perspective on objects and systematic generalization in model-based rl. *International Conference on Machine Learning (ICML) Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI*, 2019.

Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization. *Neural Information Processing Systems (NIPS) Workshop on Cognitively Informed Artificial Intelligence*, 2017.

Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations*, 2018a.

Sjoerd van Steenkiste, Karol Kurach, and Sylvain Gelly. A case for object compositionality in deep generative models of images. *Neural Information Processing Systems (NeurIPS) Workshop on Modeling the Physical World: Perception, Learning, and Control*, 2018b.

Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? In *Advances in Neural Information Processing Systems*, pages 14222–14235, 2019.

Sjoerd van Steenkiste, Karol Kurach, Jürgen Schmidhuber, and Sylvain Gelly. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309 – 325, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.07.007.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional similarity networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 830–838, 2017.

Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. SfM-Net: Learning of Structure and Motion from Video. *arXiv preprint arXiv:1704.07804*, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *JMLR*, 11:2837–2854, 2010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

Paul Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518 vol.1. ieeexplore.ieee.org, 2001. doi: 10.1109/CVPR.2001.990517.

Chr von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100, 1973.

Christoph von der Malsburg. The Correlation Theory of Brain Function. Departmental technical report, MPI, 1981.

Christoph Von Der Malsburg. Am i thinking assemblies? In *Brain theory*, pages 161–176. Springer, 1986.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1 (1-2):1–305, 2008.

DeLiang Wang. The time dimension for scene analysis. *IEEE Transactions on Neural Networks*, 16(6):1401–1426, 2005.

Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual Attention Network for Image Classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6458, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.683.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, April 2018.

Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1):66–82, 1960.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in Neural Information Processing Systems*, pages 4539–4547, 2017.

Yair Weiss and Edward H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 321–326. ieeexplore.ieee.org, June 1996. doi: 10.1109/CVPR.1996.517092.

Joseph Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Ronald J. Williams. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical report, Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science, 1989.

Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of statistics*, pages 95–103, 1983. ISSN 0090-5364.

Jiajun Wu, Ilker Yildirim, Joseph J. Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems*, pages 127–135, 2015.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, and Klaus Macherey. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

Kun Xu, Haoyu Liang, Jun Zhu, Hang Su, and Bo Zhang. Deep structured generative models. *arXiv preprint arXiv:1807.03877*, 2018.

Daniel L Yamins, Ha Hong, Charles Cadieu, and James J DiCarlo. Hierarchical modular optimization of convolutional networks achieves representations similar to macaque it and human ventral stream. In *Advances in neural information processing systems*, pages 3093–3101, 2013.

Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 1338–1349, 2019.

Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. In *Fifth International Conference on Learning Representations*, 2017.

Yanchao Yang, Yutong Chen, and Stefano Soatto. Learning to manipulate individual objects in an image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2020.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2019.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

Richard S Zemel and Geoffrey E Hinton. Developing population codes by minimizing description length. In *Advances in neural information processing systems*, pages 11–18, 1994.

Richard S Zemel and Michael C Mozer. Localist attractor networks. *Neural Computation*, 13(5):1045–1064, 2001.

Richard S Zemel, Christopher KI Williams, and Michael C Mozer. Lending direction to neural networks. *Neural Networks*, 8(4):503–512, 1995.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018a.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv:1804.06893 [cs, stat]*, April 2018b.

Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. In *Advances in Neural Information Processing Systems*, pages 1588–1600, 2019.

Yang Zhang, Ivor W Tsang, Yawei Luo, Chang-Hui Hu, Xiaobo Lu, and Xin Yu. Copy and paste gan: Face hallucination from shaded thumbnails. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7355–7364, 2020.

Yibiao Zhao and Song-Chun Zhu. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems*, pages 73–81, 2011.

Charles Y. Zheng, Francisco Pereira, Chris I. Baker, and Martin N. Hebart. Revealing interpretable object representations from human behavior. In *International Conference on Learning Representations*, 2019.

Andrey Zhmoginov, Ian Fischer, and Mark Sandler. Information-bottleneck approach to salient region discovery. *arXiv preprint arXiv:1907.09578*, 2019.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations*, 2015.

Daniel Zoran, Mike Chrzanowski, Po-Sen Huang, Sven Gowal, Alex Mott, and Pushmeet Kohli. Towards robust image classification using sequential attention models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9483–9492, 2020.