Towards Unsupervised Multi-Object Perception in Neural Networks

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

Klaus Greff

under the supervision of

Prof. Jürgen Schmidhuber

August 2022

Dissertation Committee

Dissertation accepted on 29 August 2022

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.

Klaus Greff
Berlin, 29 August 2022

Abstract

By decomposing the world in terms of objects, humans are able to recombine their existing knowledge in a virtually unbounded number ways to understand unfamiliar situations, make novel inferences, or generate new behavior. This ability to form meaningful entities from unstructured sensory information is of central importance for our impressive ability far beyond our direct experience. Contemporary neural networks still fall short of human-level generalization, which we argue is due to their inability to dynamically and flexibly bind information that is distributed throughout the network. This *binding problem* affects their capacity to acquire a compositional understanding of the world in terms of symbol-like entities (like objects), which is crucial for generalizing in predictable and systematic ways. We focus in particular on the process of perceptually grouping raw sensory inputs into meaningful objects. Importantly, we aim to enable neural networks to learn about objects in an unsupervised fashion, because their required scope and flexibility, renders adequate supervision or engineering infeasible. To that end, we propose a functional definition of objects in terms of predictive modularity, and use it to derive a formalization of perceptual grouping as a particular form of clustering. We demonstrate the feasibility of this approach by developing several neural network models that learn to segment and represent meaningful objects without supervision. Using simple synthetic datasets, we show that these representations are useful for prediction and semi-supervised classification tasks, and that they facilitate certain kinds of systematic generalization. The resulting representations are also more interpretable than non-object centric representations. We believe that a compositional approach to AI, in terms of grounded symbol-like representations, is of fundamental importance for realizing human-level generalization, and we hope that this thesis may contribute towards that goal.

Acknowledgements

First, I would like to thank my supervisor Jürgen Schmidhuber for his guidance and support, and for inspiring and challenging me to think about artificial intelligence at the highest level that I am capable of. I am very grateful to my colleagues Jan Koutnìk, Faustino Gomez, Bas Steunebrink, Paulo Rauber, Imanol Schlag, Sohrob Kazerounian, Marijn Stollenga, Johnathan Masci, Varun Raj Kompella, Matthew Luciw and everyone else at IDSIA for making it such a stimulating and fun place to work. I thank Rupesh Kumar Srivastava and Sjoerd van Steenkiste in particular, for our many fruitful and inspiring collaborations. I would also like to thank Harri Valpola, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, and the others at the Curious AI Company in Helsinki, for a fantastic collaboration. I am also grateful to Alex Lerchner, Matt Botvinick, Loic Matthey, Chris Burgess, Raphael Rishabh Kabra, Lopez Kaufman, Nick Watters, Daniel Zoran and many others at DeepMind for a stimulating research internship. Furthermore, I would like to specifically thank Sjoerd van Steenkiste, Boyan Beronov, Bas Steunebrink, and Rupesh Kumar Srivastava for countless in-depth discussions that have profoundly shaped my understanding of artificial intelligence. I would also like to thank the members of my dissertation committee – Mike Mozer, Wolf Singer, Michael Bronstein and Rolf Krause – for taking the time to review my work. Finally I am infinitely grateful for my family and the many friends who have always supported me and allowed me to be my best self. I am especially grateful for the decades long friendship with Pina Merkert that has shaped and enriched my life in more ways than I could ever tell.

Chapter 1

Introduction

1.1 Motivation

The human capacity to comprehend reaches far beyond our 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 has 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 near-infinite ways. This compositionality underlies many high-level cognitive abilities such as language, causal reasoning, mathematics, planning, analogical thinking, etc.

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 (eg. General Problem Solver [275]) for theorem proving, or chess [45]; Expert systems that made use of decision trees to perform narrow problem solving for hardware design [359] and medical diagnosis [348]; 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 [407], solve basic algebra word problems [36], or control simple virtual blocks worlds [416]. 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].

Connectionism takes a different, brain-inspired, approach to Artificial Intelligence that stands in contrast to symbolic AI and its focus on the conscious mind [276, 88]. 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. The

[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 concisely by Douglas Hofstadter as "AI is whatever hasn't been done yet".

promise of connectionism has always been as a computational model of human intelligence, and under the right conditions, they have indeed shown a remarkable capacity for learning and modeling complex statistical structure in real-world data. 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* [132]). Modern neural networks have proven highly successful and superior to symbolic approaches in perceptual domains, such as in visual object recognition [56, 204] or speech recognition [114, 328], and even in some inherently symbolic domains such as language modeling (GPT2 [300], BERT[72]), translation ([418]), board games ([350]), and symbolic integration ([221]).

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. They require large amounts of data, struggle with transfer to novel tasks, and are fragile under distributional shift. For example, it is difficult for neural language models to generalize syntactic rules such as verb tenses or embedded clauses in a systematic manner [196, 217, 238, 162]. Similarly, in vision, neural approaches often learn overly specialized features that do not easily transfer to different datasets or held-out combinations of attributes [424, 12, 331]. In reinforcement learning, where the use of neural networks has led to superhuman performance in gameplay [261, 350, 32], it is found that agents are fragile under distributional shift [189, 430, 95] and require substantially more training data than humans [379]. Importantly, they seem to lack the ability to efficiently recombine their knowledge these failures suggest that the knowledge representation learned by neural networks is not compositional in the same way that human knowledge is. This limits their applicability in many areas of artificial intelligence, where limited amounts of supervised data are available, or where the ability to react to unforeseen circumstances is critical.

This raises the question as to why existing neural networks, despite their widespread success and considerable effort to address this issue, still fail at human-level generalization. We believe that the root cause of this problem is an intrinsic inability to effectively form, represent, and relate symbol-like entities. Importantly, this impedes the ability of neural networks to decompose their inputs in terms of meaningful object representations (perceptual grouping). In some cases, such as in vision, it may appear that object-level abstractions can emerge naturally as a byproduct of learning [434]. However, it has repeatedly been shown that such features are best understood as “a texture detector highly correlated with an object”[283, 372, 6, 39, 99]. In general, evidence indicates that neural networks learn mostly about surface statistics (eg. between textures and classifications in images) in place of the underlying concepts [180, 191, 216]. Most work on representation learning focuses on feature learning without even considering multiple objects or treats segmentation as an (often supervised) preprocessing step. In contrast, we argue for the importance of jointly learning to segment and represent objects in an unsupervised fashion. This unsupervised perceptual grouping, if successful, could help to bridge the gap between raw input data and symbolic reasoning, and thereby provide the foundation for compositional reasoning and systematic generalization in neural networks.

1.2 Contributions

The contributions of this thesis towards this goal are organized into two parts: The first part is dedicated to developing a better understanding of the problem and to building a corresponding conceptual framework for thinking about this important issue. We argue that the underlying cause for these problems is related to the *binding problem*. The binding problem originated in neuroscience, where it refers to an explanatory gap of how dynamic information processing is organized in the brain. In the context of neural networks, this lack of understanding translates into an inherent limitation of existing neural networks: Their inability to dynamically and flexibly *bind* information that is distributed throughout the network, while also keeping it separate from other unrelated information. We discuss how the binding problem affects the ability of current neural networks to form meaningful entities from unstructured sensory inputs (segregation), to maintain their separation at a representational level (representation), and to use these entities to construct new inferences, predictions, and behaviors (composition). We carefully analyze each of these three aspects, connect them to findings from cognitive psychology and neuroscience, and survey the machine learning literature for promising approaches.

Based on these insights, the second part of the thesis presents a series of methods that we have developed to tackle the segregation problem in neural networks. Our goal is to build a system that can learn to structure its input into meaningful object representations in a completely unsupervised manner. To that end, we first propose an unsupervised notion of objects in terms of mutual predictability of the individual inputs (i.e. pixels). We then develop a proof-of-concept method called *Reconstruction Clustering* (RC; 122), that implements perceptual grouping as a simple clustering procedure using the reconstruction quality from a *Denoising Autoencoder* (DAE; 27, 391) as the distance metric. RC is able to recover the objects from synthetic binary images of simple shapes without supervision. Next, we further develop this idea into a corresponding mathematical framework called *Neural Expectation Maximization* (N-EM; 123), and show how it can be implemented as a recurrent neural network architecture. The resulting system improves upon the results obtained by RC, and can also be trained end-to-end by backpropagation. It can also be applied to video data and is able to make use of the temporal information of simple moving shapes, but the system is still restricted to very simplistic synthetic datasets. To help scale perceptual grouping to more realistic images, we then propose the *iTerative Amortized Grouping* (TAG; 120) framework that incorporates direct dependencies between the object segmentation and their representations into the model. The resulting system (Tagger) can deal with more complex tasks, such as texture segmentation, and even demonstrates superior performance regarding a challenging semi-supervised classification task. Finally, we focus on improving the object representations by developing a method for perceptual grouping based on the successful *Variational Autoencoder* (VAE; 313, 199) framework. This system, which we call *Iterative Object Decomposition Inference NEtwork* (IODINE; 119), can learn meaningful disentangled object representations completely unsupervised from semi-realistic synthetic data.

1.3 Organization

The rest of this thesis is organized as follows: In Chapter 2 we present our perspective on the binding problem and its importance in connectionism. This chapter develops the conceptual framework for the rest of this thesis, which is centered around object representations. It also discusses the challenges and surveys relevant approaches regarding the efficient formation, representation, and combination of multi-object representations. Chapter 3 then briefly reviews the relevant mathematical background in probability theory, machine learning and neural networks. The following four chapters present a series of four methods for unsupervised perceptual grouping: Chapter 4 introduces an unsupervised notion of objects based on predictability and the corresponding *Reconstruction Clustering* (RC; 122) method. Chapter 5 further develops the Neural Expectation Maximization framework and its implementation as a form of recurrent neural network. In Chapter 6 we present an alternative approach called Tagger, which directly learns the process of segmentation and demonstrates the usefulness of perceptual grouping for semi-supervised classification. Chapter 7 introduces a further improved system for unsupervised perceptual grouping called IODINE, and demonstrates the high quality of its learned object representations. Finally, Chapter 8 closes the thesis with a summary and a brief discussion of some important open problems.

Chapter 2

The Binding Problem

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 (bind) 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 information based on a particular context and thereby accommodate different 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 event), motor-behavior (an action), and sensorimotor binding (hand-eye coordination) [377, 319, 86]. 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[1]. Several theories have been proposed as to how binding may be accomplished in the brain. The most basic explanation is that it happens on an architectural level through conjunction cells (also known as cardinal cells, line coding, grandmother neurons). That means for each distinguishable pattern of activity that needs to be bound, there is a separate neuron. This strategy is common in the brain, and well known especially at the lower layers of visual processing. But the number of neurons required for this strategy grows exponentially with the complexity of the patterns and becomes intractably large for higher level processing. Furthermore, when to encode a novel object (or novel appearance of a known object), a new neuron would have to be instantly recruited. To generalize correctly, this new neuron would have to be appropriately connected and with synaptic weights already calibrated. This is obviously impossible, which is why conjunction cells are clearly an inadequate explanation for the dynamic binding at the level of objects.

Synchronization provides an elegant alternative (or complementary) explanation for the dynamic binding of neuronal activity (Temporal correlation hypothesis; [257, 395]). The idea is that neurons can dynamically bind together through temporal synchronization of their firing patterns. Consider the example of simultaneously representing a “red square” and a “blue

[1]The term binding problem has also been used in the context of consciousness, as the problem of how a single unitary experience arises from the distributed sensory impressions and processing in the brain [352]

triangle”: The neurons for “red” and “square” would fire in sync (oscillating phase-locked) to indicate that they belong to the same objects. The same would be true for the neurons for “blue” and “triangle”, while at the same time both objects would remain out of sync (and thus remain separated). Grouping neuronal activity based on synchrony offers a degree of freedom separate from the firing rate (see also Section 2.1.3). Synchrony can easily be detected by other neurons, because they are very sensitive to the relative timing of incoming (pre-synaptic) spikes. A neuron sensitive to the combination of “red” and “square” is thus activated much more strongly, when both features belong to the same object (and are thus synchronized). Binding by synchrony scales and generalizes much better than conjunction codes and is potentially fast enough to support the dynamic creation and destruction of bindings in the timeframe of single saccade. It also seems plausible due to the ubiquity of oscillators in the brain, and the well-known large-scale rhythmic patterns found in neuronal activity. Finally, it is also physiologically plausible, since both neuronal firing and synaptic learning are sensitive to the timing of incoming spikes (*Spike-Timing Dependent Plasticity* (STDP; 48)). There is diverse experimental data to support this interpretation [see eg. 354, 382]), but the extent to which the brain uses synchrony for binding is still under debate.

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 we believe is common to all of the above subproblems. On the other hand, although we are convinced that this problem can be addressed by incorporating a general dynamic information binding mechanism, it is less clear how this can be implemented. Indeed, the search for an adequate mechanism for binding (in one form or another) is a long-standing problem, not just in neuroscience and cognitive psychology, but also in machine learning [355, 356, 371]. Rather than focusing on a particular subproblem, here we propose to tackle the binding problem in its full generality, which touches upon all these related areas of research. In this way, we can connect ideas from otherwise disjoint areas, and thus draw upon a large body of research towards developing a general binding mechanism. Inspired by Treisman [378], we organize our analysis along a functional division into three aspects pertaining to the role of binding for symbolic information processing in neural networks: *1) representation, 2) segregation, and 3) composition*, each of which takes a different perspective on the binding problem.

- The **Representation Problem** is about encoding relevant information in a way that combines the richness of neural representations with the compositionality of symbols.
- The **Segregation Problem** is about the process of forming grounded object representations from raw unstructured inputs, and is the main focus of this thesis.
- 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 chapter provides an in-depth analysis of these aspects of the binding problem from a functional perspective, and surveys relevant approaches for addressing them in the context of neural networks. It is based on the paper Greff et al. [124] which is the result of a tight collaboration with Sjoerd van Steenkiste, and both authors contributed substantially.

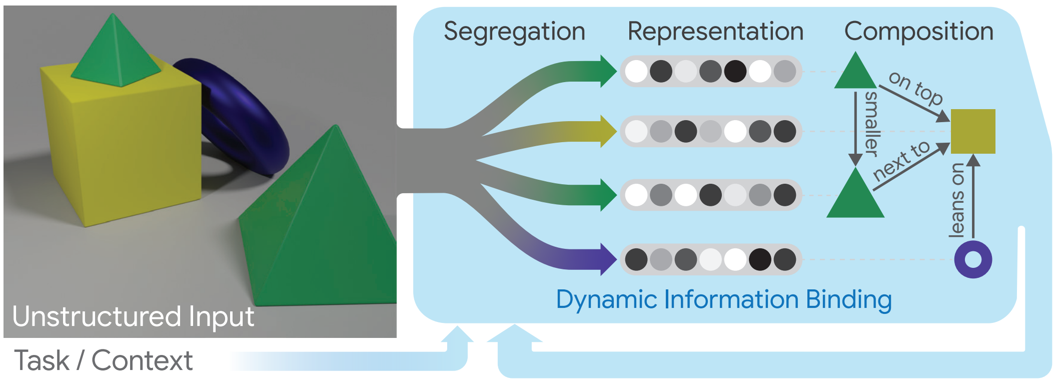


Figure 2.1. The binding problem in artificial neural networks can be understood from the perspectives of *segregation*, *representation*, and *composition*. Each of these subproblems focuses on a different functional aspect of dynamically binding neurally processed information with the aim of facilitating more symbolic information processing.

2.1 The Representation Problem

The Representation Problem is concerned with binding together information at a representational level that belongs to separate symbol-like entities. It revolves around so-called *object representations*, which 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 expressive distributed feature-based internal structure of connectionist representations, which are known to facilitate generalization [144, 29]. Hence, object representations encode relevant information in a way that combines the richness of neural representations with the compositionality of symbols. We chose the term “object” representation because it is evocative of physical objects, which are processed as symbols in many important cognitive tasks. However, we emphasize that object representations are 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[2].

Interestingly, even the seemingly basic task of incorporating object representations in neural networks faces several problems, such as the “superposition catastrophe” [396] portrayed in Figure 2.2. It suggests that fully-connected neural networks suffer from an “inherent tradeoff between distributed representations and systematic bindings among units of knowledge” [160]. A general treatment of object representation in neural networks involves addressing the superposition catastrophe, along with several other challenges.

Consider for example Figure 2.3, where you are able to distinguish between five different objects. You can readily describe each object in terms of its 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,

[2]We have considered several 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.

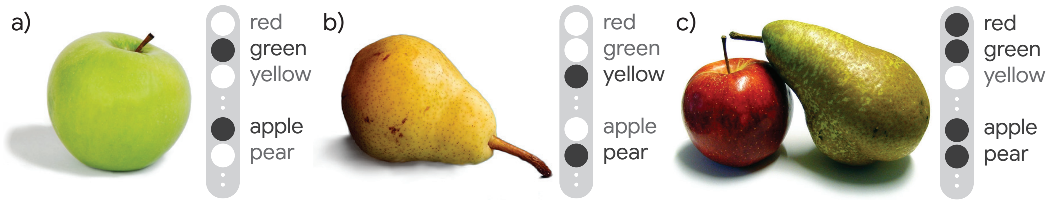


Figure 2.2. Illustration of the *superposition catastrophe*: A distributed representation in terms of disentangled features like color and shape **(a, b)** leads to ambiguity when confronted with multiple objects **(c)**: The representation in **(c)** could equally stand for a red apple and a green pear, or a green apple and a 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 [318], and has been debated in the context of neuroscience since [257, 395].

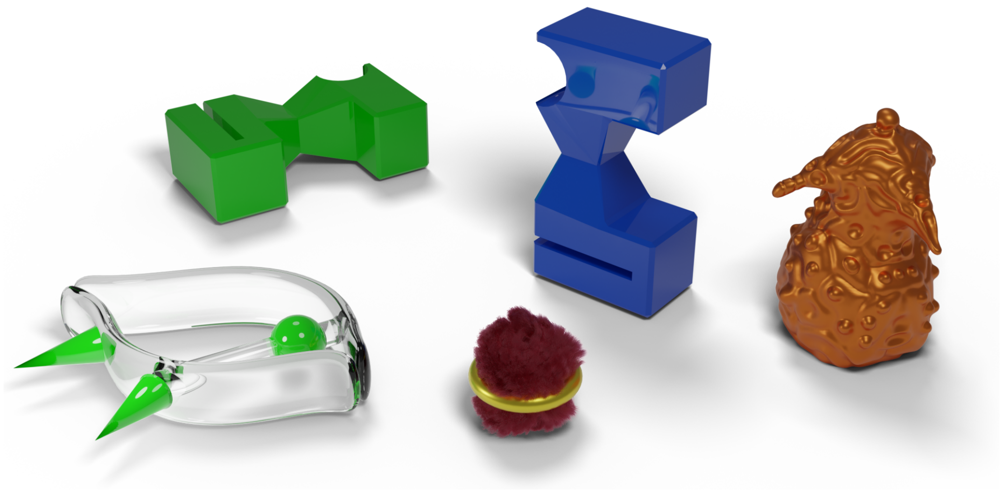


Figure 2.3. Example of several unfamiliar objects, that can nonetheless be described in terms of their features.

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 (Section 2.1.1). 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 also consider the representational *dynamics* that are required to support stable and coherent object representations over time (Section 2.1.2). Towards the end, we survey relevant approaches from the literature that may help incorporate these aspects of object representations into neural networks (Section 2.1.3).

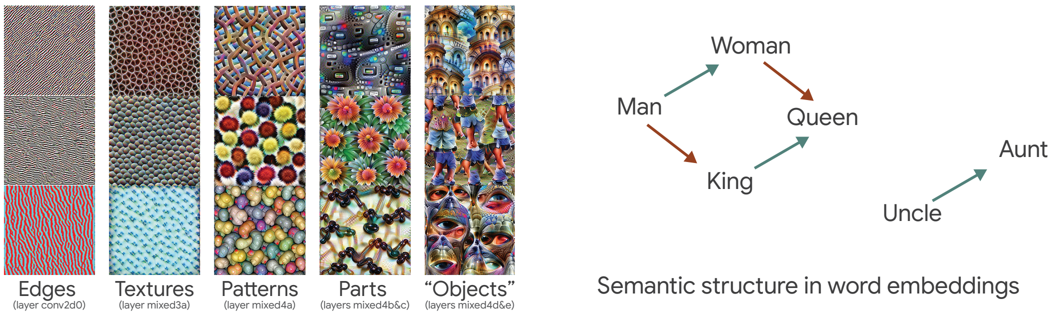


Figure 2.4. **Left:** Interpretable features learned on ImageNet as observed in Olah et al. [284]. **Right:** Learned word embeddings have been demonstrated to capture some of the semantic structure of text [255], although to a lesser extent than was initially reported [280].

2.1.1 Representational Format

We seek a representational format that distinguishes objects, while retaining the advantages of learned distributed representations. These representations have proven highly successful [eg. 55, 143, 211] and are known to partially capture the semantic structure of a task (Figure 2.4), such as interpretable image features [427, 283], or the semantic structure of text (255; but compare 280). In this way learned object representations can also benefit from known inductive biases that focus on feature hierarchies, invariances, and spatio-temporal coherence [25], sparsity [285], or non-Euclidean feature spaces [278].

Separation

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 capacity, it may suffice to represent only a few objects simultaneously, despite the fact that a typical scene potentially contains a large number of objects. Indeed, the capacity of the human working memory is generally believed to only be around 3–9 objects [93, 256].

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 2.3 you were able to freely compare a number of 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 2.3. 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.

Disentanglement

Individual object representations need to be able to describe a large variety of (possibly unseen) objects in terms of attributes that are useful for down-stream problem-solving. This requires focusing on factors of variation in the data, that are sufficiently expressive, but also compact and reusable (i.e. they can be varied independently). Indeed, humans arguably manage to accomplish this by focusing on a relatively small, but consistent set of attributes such as color, shape, etc. [71].

A *disentangled* representation aims to make these attributes explicit by establishing a local correspondence between (independent) factors of variation and features [19, 338, 141, 140, 314]. In this case, information about a specific factor can be readily accessed and is robust to unrelated changes in the input, which improves sample efficiency and down-stream generalization [142, 387]. In the context of object representations, disentanglement implies a factorized feature space that captures salient properties of objects. Together with a common format, it facilitates generalization to unseen feature combinations and enables useful comparisons between objects and other meaningful relations to be formed.

2.1.2 Representational Dynamics

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

Temporal Dynamics

An object representation requires ongoing updates across time for a number of reasons: 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 among all these cases is the need for an object representation to consider not only the input but also its own 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 fairytale prince that is transformed into a frog [245, 18].

Reliability

Structured mental models depend on object representations to provide a stable foundation for reasoning and other types of information processing [182]. The reliability of this foundation is especially important for more abstract computations to which object 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 (eg. 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 (for example see Figure 2.8 on page 15). 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 Sections 2.2.2 and 2.3.2).

2.1.3 Methods

In order to fulfill the desiderata outlined above, we require a number of specialized inductive biases. Indeed, it should now also be clear that a simple *Multi-Layer Perceptron* (MLP; 171, 170) 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[3]. Therefore, in the following we will review several approaches for representing multiple objects in neural networks. We will focus on common format, temporal dynamics, reliability, and in particular on separation, which thus far has received little attention in the main-stream neural networks literature.

Slots

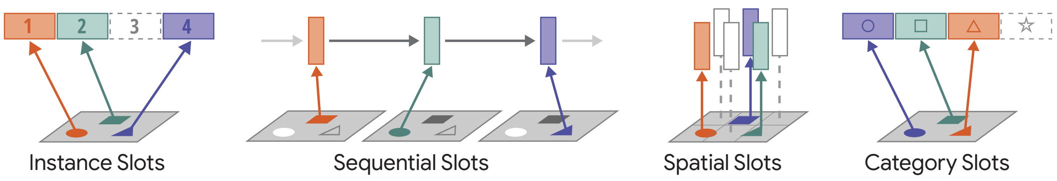


Figure 2.5. Illustration of the four different types of slot-based representations.

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.

[3]Others have suggested ways in which MLPs could *in principle* circumvent this problem [287, 299]. However, neither of these offer a solution that can convincingly fulfill all of the above desiderata simultaneously. In fact, even for plain *Recurrent Neural Networks* (RNNs; 250, 368, 316, 410) it was found that when they are trained to remember multiple objects internally, they resort to a localist representation [37].

Instance Slots In the 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. 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 enforced 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 one. Hence, the allocation of information to each slot must be determined by taking the other slots into account, which complicates the process of segregation (see also Section 2.2.2). Instance slots have been used in several approaches to learning object representations, including *Masked Restricted Boltzman Machine* (MRBM; 223), *Neural Expectation Maximization* (N-EM; 123), and *Iterative Object Decomposition Inference NEtwork* (IODINE; 119). They can also be found in the memory of memory-augmented neural networks [185, 117], in self-attention models [388, 66, 236], in *Recurrent Independent Mechanisms* (RIMs; 111), albeit without having a common format, in all approaches presented in this thesis (see Chapters 4 to 7), and in certain graph neural networks [20], where they are treated as internal representations that can be accessed simultaneously.

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 (*Attend Infer Repeat* (AIR; 84); 84) and to a lesser degree in DRAW [125]. However, due to recurrence, these slots may not always be fully independent, which impedes their function as modular building blocks. Recent approaches, such as *Multi-Object Network* (MONet; 43) and GENESIS [83], 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 [16] or in pointer networks [393].

Spatial Slots In *spatial slots*, each slot is associated with a particular spatial coordinate (eg. 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 *Convolutional Neural Networks* (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 [332], the Multi-Entity VAE [273], or the works by Zambaldi et al. [425], Stanić et al. [366] explicitly treat each spatial position in the filter-map of a CNN as a candidate object representation. Even more recent approaches, such as SPAIR [58], SPACE [231], and SCALOR [179], 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

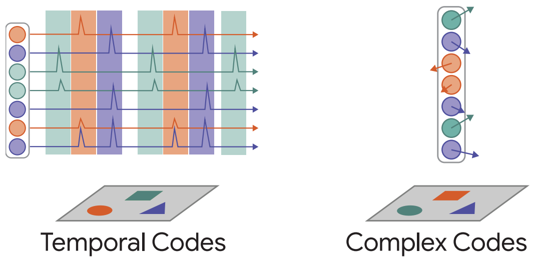


Figure 2.6. Illustration of the two main augmentation based approaches to object representations. **Left:** Neural activity over time for a temporal code, where synchronization is emphasized using color. **Right:** Complex valued activations are represented by arrows and colored according to their direction.

is that their spatial slots are typically tailored towards 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. This too can serve to break slot symmetries for the purpose of information routing, and is further expected to mitigate the dependence of spatial slots on spatially separated inputs. In this case, however, because now category and separation are entangled, it is then no longer possible to represent multiple objects of the same category[4]. The main example of category slots are capsules [145, 146], although other approaches such as Recurrent Entity Networks [138] 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 connectionist systems 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 (eg. two red objects), similar to when representing multiple objects of the same category using category slots footnote 4.

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 [353]. In other words, the features that form an object are represented by neurons that fire in synchrony (257, 395, 351). 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 [402]). Because temporal codes rely on spiking neurons, they are non-differentiable and also require simulating the dynamics of each neuron even for

[4] There is some evidence that humans struggle with feature overlap too and show reduced working memory capacity in these cases [266].

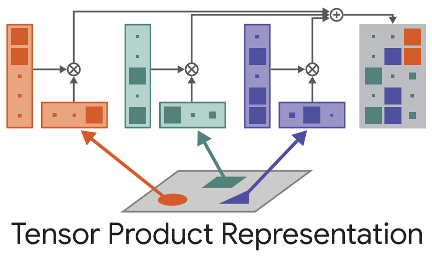


Figure 2.7. Illustration of a Tensor Product Representation (matrix on the right) that is formed through combining a role vector (horizontal) and a filler vector (vertical) for each object.

static inputs. This makes them incompatible with gradient-based training, and necessitates a completely different training framework [eg. 75, 76] typically based on Hebb's rule [195], or *Spike-Timing Dependent Plasticity* (STDP; 48).

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 smooth 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[5], which tend to be difficult to integrate with existing methods. Successful integrations include complex-valued Boltzmann Machines [309, 429] and complex-valued RNNs that could be trained either with backpropagation [269] or via Hebbian learning [304].

Tensor Product Representations

A *Tensor Product Representation* (TPR; 357) consists of a real-valued matrix (tensor) that is the result of combining distributed representations of *fillers* with distributed representations of *roles*. *Tensor Product Representations* (TPRs; 357) 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. 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 [357] and several modifications have since been proposed that consider different binding, unbinding, and conjunction operations (298, 187, 98; see 193 for an overview). In the recent literature, TPR-like mechanisms have been incorporated into neural networks using fast-weights [335] or self-attention [336] to perform reasoning in language.

[5]In some sense, complex codes can be seen as an instance of a more general – yet unexplored – class of vector-valued activations that use the additional degrees of freedom for grouping.

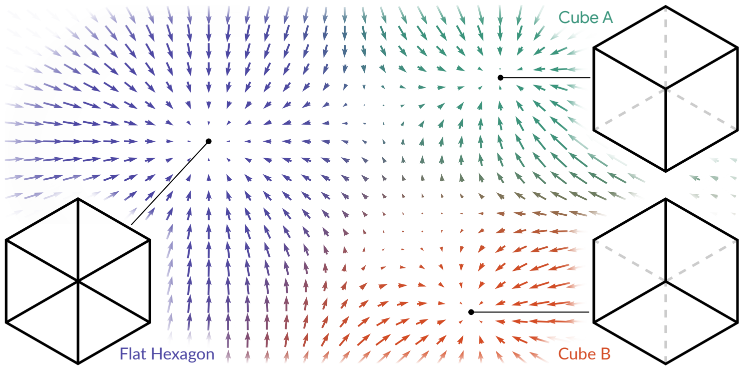


Figure 2.8. Correspondence of attractor states to visual interpretations for a tri-stable variant of the Necker cube. The vector field illustrates the (input-dependent) inference dynamics in feature space, with one attractor for each stable interpretation.

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 (Section 2.1.2). Robust object representations are well described by a stable attractor state in a larger dynamical system that models the representational dynamics based on a given 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 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 interpretations).

Attractor Networks incorporate attractor dynamics in neural networks and have a long history in connectionist research. Early work includes Hopfield networks [153], Boltzmann machines [3][6], and associative memory [203]. Attractor states were also found to occur naturally in RNNs, especially when using symmetric recurrent weights [5, 297]. In recent years, however, they have received little attention (but see [268, 169]), 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 [274]. Localist attractor networks [428] and flexible kernel memory [281] are two approaches that address this issue by introducing a separate representation for each attractor. However, note that spurious attractors that correspond to novel feature combinations may also be advantageous for generalization.

2.2 Segregation

The segregation problem is about the process of structuring raw sensory information into meaningful entities. It is concerned with the information binding required for dynamically *creating* object representations, as well as the characteristics of objects as modular building blocks for guiding this process. Unlike in Section 2.1, where we focused on the need for binding at a representational level to maintain a separation of information for *given* entities, here we focus on the process of *creating* object representations through binding previously

[6]Boltzmann machines build upon work on the dynamics of the Ising model in the physics of Spin-Glass [106, 346].

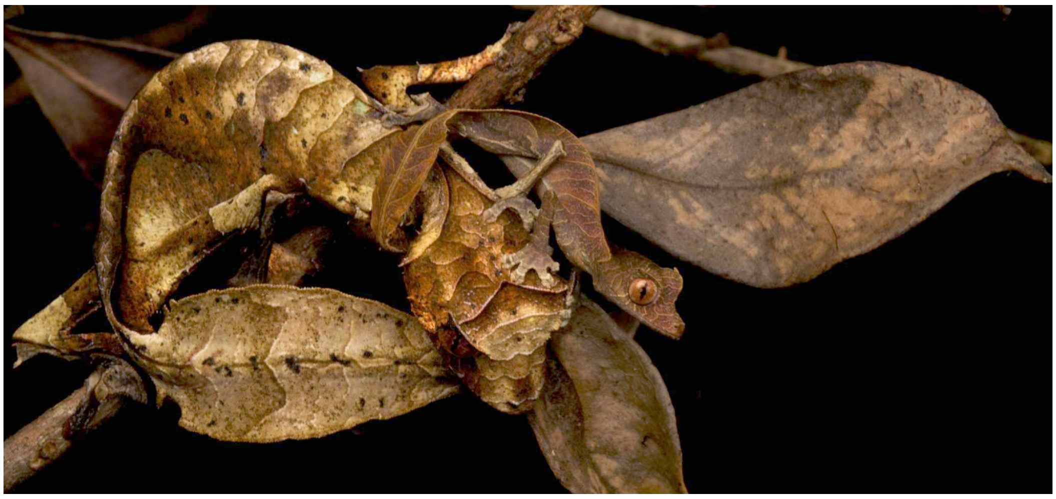


Figure 2.9. Photo of two leaf-tailed geckos — "young and old" © 2015 by Paul Bertner.

unstructured (raw) sensory information. Humans effortlessly perceive the world in terms of objects, yet this process of perceptual organization is surprisingly intricate [399]. Even for everyday objects like a mirror, a river, or a house, it is difficult to formulate precise boundaries or a definition that generalizes across multiple different contexts. The incredible variability among objects makes it intractable to resolve the segregation problem purely through supervision. 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. The segregation problem 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. Consequently, the segregation problem is about enabling neural networks to acquire an appropriate, context-dependent, notion of objects in a mostly unsupervised fashion.

Consider for example Figure 2.9, which demonstrates several challenges for segregation 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 the following,, we take a closer look at this process of segregation[7]. We first work towards a general *notion* of an object built around modularity and hierarchy (Section 2.2.1). Next, we focus on the process of *forming* object representations based on this notion (Section 2.2.2). Unlike segmentation, which is typically only concerned with a static split 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 survey relevant approaches from the literature that may help neural networks perform segregation (Section 2.2.3).

[7]We refer to this process as *segregation* rather than *binding*, to emphasize the fact that it typically requires a *separation* of the inputs and features into meaningful parts.

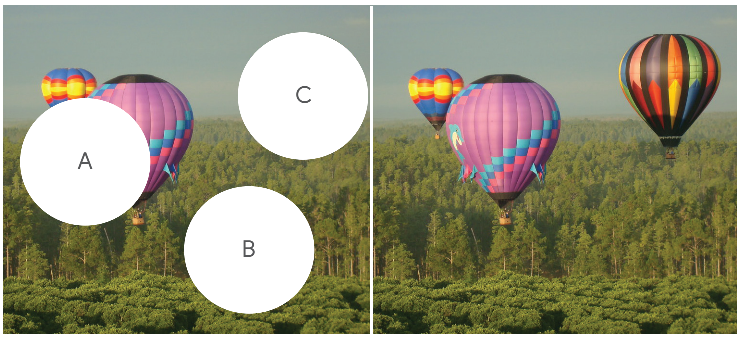


Figure 2.10. For partial objects **(A)** or only background **(B)**, the occluded regions can be inpainted reasonably well, while in the case of full object occlusion **(C)** that is usually impossible.

2.2.1 Objects

The question of what constitutes a meaningful object (i.e. for 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 [118, 46]. 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)[8].

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 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 intuitive notion of objects in the real world [118, 51].

Consider the example of three balloons in front of a forest as depicted in Figure 2.10. 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 (eg. 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.

[8]*"A map is not the territory it represents, but, if correct, it has a similar structure to the territory, which accounts for its usefulness."* [207].

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. However, in rare cases, two decompositions may also consider incompatible parts, as, for example, in a page of text that can be decomposed either into lines or sentences[9]. Notice that there is a difference between this part-whole hierarchy and the feature hierarchy typically found in 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 be viewed in terms of a number of general “is-part-of” relations that can be reused between objects (see also Section 2.3.1).

Multi-Domain

It is worth emphasizing that objects (as referred to in the context of this paper) are not restricted to vision, but also span sensory information from other domains such as audio or tactile[10] (and even be entirely abstract, although this is not the focus of segregation). For example, auditory objects may correspond to different sources of sound, such as speakers talking simultaneously in the same room (cocktail-party problem; 54). Objects in the tactile domain are perhaps less obvious, but consider the example of writing on a piece of paper with a pen, where you can clearly separate the sensations that arise from your fingers touching each other, touching the pen, and touching the paper [see also 190]). Notice how you are likely to associate the sensations of touching the pen and its visual perception with a common cause and therefore with the same object. This implies that objects can be simultaneously grounded in sensory information from multiple domains, which may help resolve ambiguities (eg. McGurk Effect; 251).

2.2.2 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 (see multi-modal separation uncertainty Section 2.1.2). Together, these result in a massive number of potential object representations (eg. $\geq$ 3000

[9]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* “borders of predictability”, which implies a weaker internal structure.

[10]It is even discussed whether humans are capable of object perception in the olfactory domain [22].

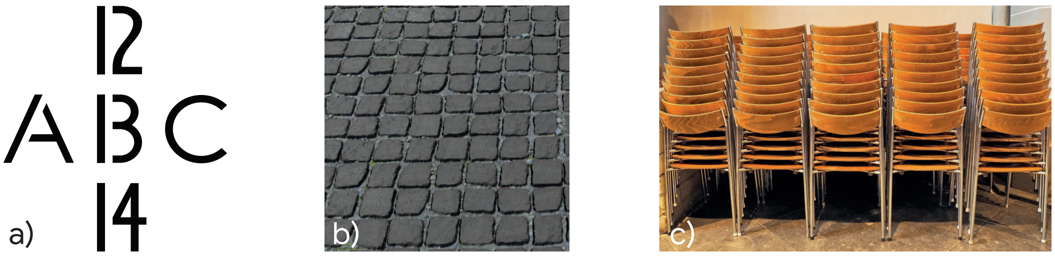


Figure 2.11. Human perception is multistable, which is often demonstrated using visual illusions as in **(a)**, yet it is also often encountered in the real world, eg. for different groupings of tiles **(b)**. To steer segregation towards a useful decomposition it is important to incorporate contextual information, for example to decide between a decomposition based on chairs or based on stacks in **(c)**.

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 [10]. This effect is often demonstrated with visual illusions as in Figure 2.11a, but is in fact much more common than these constructed examples suggest. For example, a simple tile pattern (as in Figure 2.11b) can easily be perceived in several ways, including rows or columns of tiles. Multistability can also be observed in other sensory modalities such as audio, tactile, and even olfaction [345]. Notice that it is possible to simultaneously perceive multiple objects from the same decomposition, but not from different decompositions (eg. perceiving 13 and B simultaneously in Figure 2.11a). 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. It 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 (see Figure 2.11c). 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 (eg. to guide visual search) or based on a measure of success at performing the given task. Memory could act as another source of contextual information, 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 (Section 2.1.2), and to ensure that the outcome of more abstract computations in terms of object representations remain valid in the environment (Section 2.1.2). It may also help to avoid "double-counting" of evidence (eg. during learning)[11]. 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, re-identification may require interfacing with some form of long-term memory.

2.2.3 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 only to a lesser degree. By also discussing the latter, we aim to provide inspiration for the development of neural approaches that can learn about objects directly from raw data (eg. by focusing on modularity).

Clustering Approaches to Image Segmentation

Image segmentation is concerned with segmenting the pixels [or edges 9] belonging to an image into groups (eg. 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 [173]. One particularly successful approach is the spectral graph-theoretic framework of normalized cuts [347], 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. It is straightforward to achieve a hierarchical segmentation in

[11]Consider the example from Marcus [245] about owning a three-legged dog. Despite the fact that 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.

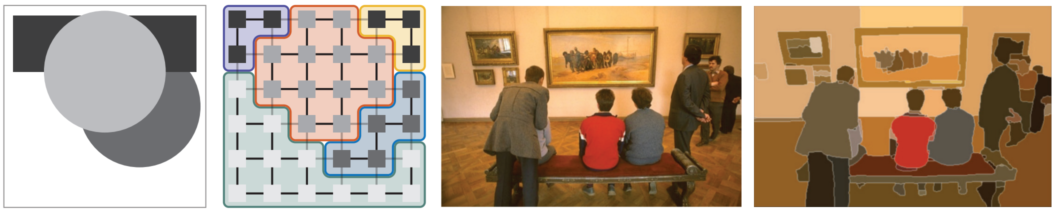


Illustration of Spectral Clustering
Example PMI based Image Segmentation

Figure 2.12. **Left:** An illustration of (spectral) clustering approaches, which treat image segmentation as a graph-partitioning problem. **Right:** Corresponding instance segments as obtained by Isola et al. [166].

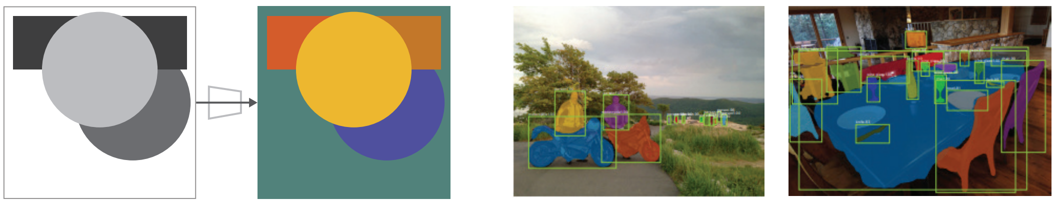


Illustration of Neural Image Segmentation
Example Image Segmentations by Mask R-CNN

Figure 2.13. **Left:** An illustration of neural approaches that learn to directly output an image segmentation. **Right:** Corresponding bounding boxes and instance segments as obtained by He et al. [136].

this graph clustering framework, either via repeated top-down partitioning [347] or bottom-up agglomerative merging [263, 152].

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 [eg. as in 347, 242] has little chance at facilitating the 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, eg. to learn a similarity function between segments [312, 82, 206] or discrete graphical patterns [239], to learn boundary detection [248, 152], or as a means of top-down feedback [263]. Unsupervised approaches (based on self-supervision) provide a more promising alternative. One approach is to learn a similarity function between pairs of pixels, eg. based on their point-wise mutual information using kernel-density estimation [166] or based on self-supervised prediction using a neural network [167]. Alternatively, one can attempt to steer the clustering process based on the unsupervised principle of compressibility (minimum description length; 263).

Notice 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, but see 23).

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

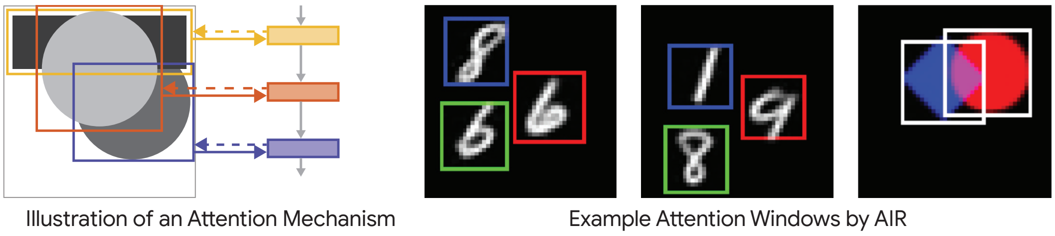


Illustration of an Attention Mechanism
8
6
6
1
9
8
Example Attention Windows by AIR

Figure 2.14. **Left:** An illustration of attention-based approaches, which sequentially attend to individual objects. **Right:** Corresponding attention windows as obtained by Eslami et al. [84].

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 segmentation have produced high-quality instance segmentations of real-world images[12]. For example, approaches based on R-CNN [105] decompose the instance segmentation problem into the discovery of bounding boxes using region-proposal networks [311] and mask prediction [62, 136] to provide instance segmentations. The more recent DEtection TRansformer (DETR; 49) was able to integrate these stages into a single Transformer-based network using a bipartite matching loss. Other approaches output an energy function from which the segmentation is easily derived, eg. based on the Watershed transformation [17]. Instance segmentation has also been phrased as an image-to-image translation problem using conditional generative adversarial networks [262]. Approximate instance segments can also be obtained as a by-product of performing some other task, such as learning to interpolate between multiple images [8] or minimizing mutual information between image segments [422].

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. [177] propose 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 (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, eg. 61), which can for example be learned through unsupervised multi-task learning [302].

Sequential 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 (eg. 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

[12]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 can not segregate multiple objects belonging to the same class.

of an attention window, eg. in the shape of a bounding-box [367] or a fovea [343]. Their strong spatial bias (due to the shape of the attention window) makes them particularly relevant for the domain of images, but 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 for incorporating attention mechanisms is in correctly placing 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 [220, 4, 383]. A classifier can then be trained to determine which window contains an object [322, 394, 133]. 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 [168]. Salient regions can be learned based on bottom-up information, such as the self-information of local image patches [42]. Alternatively, they can be derived by also incorporating top-down information, eg. by highlighting locations that are (maximally) informative with respect to a discriminative task [96, 47, 433]. Recently, there has been renewed interest in saliency-based approaches through the discovery of keypoints [174, 212, 258, 109].

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 [367], trained with Q-learning [288], or via Policy Gradients [44]. Alternatively, it can be incorporated as a separate action in an agent trained to perform some task (eg. classification) or to interact with an environment [260, 15]. *Attend Infer Repeat* (AIR; 84) and its sequential extension SQAIR [209] 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 [172], but see also DRAW [125] for an alternative mechanism. Similarly, [375] 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 *Multi-Object Network* (MONet; 43) [43], GENESIS [83], and ECON [398] 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[13]. A similar soft-attention mechanism has also been used to facilitate supervised learning tasks such as caption generation [420], instance segmentation [310], or (multi-)object tracking [208, 92]. Soft attention mechanisms have also been applied *internally* (self-attention) to support segregation. For example, Mott et al. [265] incorporates a form of dot-product attention [388] 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 [435].

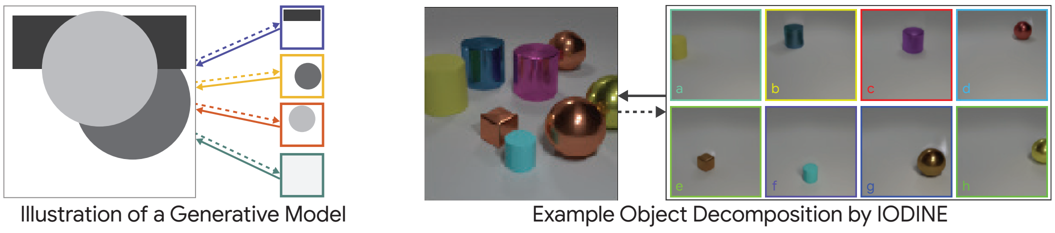


Illustration of a Generative Model
Example Object Decomposition by IODINE
a
b
c
d
e
f
g
h

Figure 2.15. **Left:** An illustration of generative approaches to segregation that model an image as a mixture of components. **Right:** A corresponding decomposition in terms of individual objects as obtained by Greff et al. [119].

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) [14]. 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 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 (i.e. for multistability).

The most basic assumption to incorporate in a generative model, for the purposes 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 [329, 91]. Alternatively, the decomposition can be based on other features such as optical flow [176] or the coefficients of a wavelet transform [128]. Mixture models can also be biased towards spatial coherence to explicitly account for the spatial structure of visual objects [406, 35]. *Independent Component Analysis* (ICA) models the observed data as linear combinations (mixtures) of unobserved random variables (sources) that are statistically independent [163]. This approach has been particularly successful at blind source separation (segregation) in the auditory domain (eg. the cocktail party problem 54), although it has also seen application in the context of images [226].

To more accurately model complex data distributions, it is possible to incorporate domain-specific assumptions 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" [67]. The results that can be obtained by incorporating domain-specific knowledge are impressive [432, 67, 68, 380, 381]. However, performing inference in highly complex generative models of this type is problematic and

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

[14]Human perception is also said to be generative in the sense that we often perceive objects as coherent wholes even when they are only partially observed (amodal completion; 254).

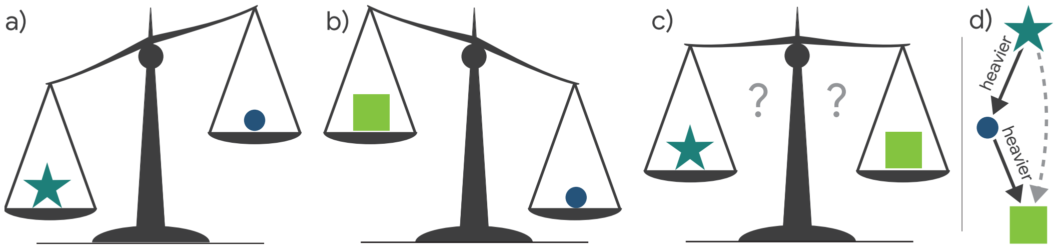


Figure 2.16. Three different objects (🟩, •, ★) appear in different pairings on a scale **(a)** and **(b)**. By evaluating their relationships **(d)**, it can be inferred how the scale will tip in **(c)**.

frequently relies on custom inference methods tailored to this particular task (eg. Markov Chain Monte Carlo using jump moves to remove or add objects or specific initialization strategies). 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 [243] and incorporated deep neural networks to help make inference more tractable [213, 317]. 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 overly relying on supervised labels from a particular domain).

An alternative approach to more accurately modeling complex data distributions is to incorporate fewer assumptions, and rather parameterize the generative model with a neural network that can *learn* a suitable generative process from many different observations. For example, [386] demonstrates 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 a background [see also 277, 80, 279]. 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 [120], generalized expectation-maximization [123], iterative variational inference [119], slot attention [236], or parallel spatial (bounding-box) attention [231, 178]. Further improvements can be made by assuming access to multiple different views to explicitly model 3D structure at a representational level [52, 272]. Even though 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 (eg. common format, disentangled, modular), in a completely unsupervised manner.

2.3 Composition

In this section, we look at the binding problem from the perspective of composition: building structured models of the world that are *compositional*. These structured models leverage the modularity of objects to support different patterns of generalization, and are the means by which more systematic 'human-like' generalization can be accomplished. Here we encounter the need for variable binding: the ability to combine object representations and relations without losing their integrity as constituents (as is needed for compositionality). Compositionality is a core aspect of human cognition and underlies our ability to understand novel situations in terms of existing knowledge. Similarly, in the context of AI, it supports the systematic reuse of familiar

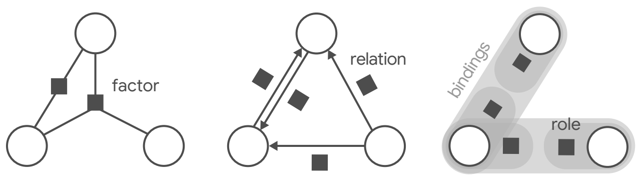


Figure 2.17. Three different ways in which structure can be defined in terms of relations between objects: As a factor graph, a directed graph, or as nested role-filler bindings.

objects and relations to dynamically construct novel inferences, predictions, and behaviors, as well as the ability to efficiently acquire new concepts in relation to existing knowledge. However, this relies on the ability to learn abstract relations that can be arbitrarily and recursively applied to object representations, and requires a form of binding, not unlike the way variables can be bound to placeholder symbols in a mathematical expression. Moreover, the desired structure is often not known in advance and has to be inferred or adapted to a given context or task.

Consider the sequence of observations in Figure 2.16, 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 a *relation* between them. On the other hand, it can still be used to update the properties of the participating objects in response to new information (eg. 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 🟩.

In the following, 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 a compositional structure that combines relations and object representations without undermining their modularity (Section 2.3.1). Next, we consider how a neural network can dynamically infer the appropriate structure and leverage it for the purpose of reasoning (Section 2.3.2). Towards the end, we survey relevant approaches from the literature that address these aspects of composition (Section 2.3.3).

2.3.1 Structure

To implement structured models, a neural network must organize its computations to reflect the desired *structure* in terms of objects and their relations. This structure is generally described by a graph where nodes correspond to objects and edges to relations[15]. By representing relations separately (independent of object 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 (eg. symmetry, 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 (eg. "collides with"), hierarchical relations ("is part of"), or comparative relations (eg. "bigger than"). Moreover, these general relations can often be specialized to include the nature or strength of an interaction (eg. "*elastic* collision", "*much* bigger than"). 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[16]. 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 2.17 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 correspond to the different *roles* that the participating objects play (see Figure 2.17 c). Finally, it is important that relations are represented separate from and independent of the object representations (see also *role-filler-independence*; 161). This enables relations and objects to be composed in arbitrary ways to form a wide variety of (potentially novel) structures.

Variable Binding

To enable a single neural network to implement different structured models, it requires a suitable 'variable binding' mechanism[17] that can *dynamically* combine modular object representations and relations. Consider the classic example of Mary and John adapted from Fodor and Pylyshyn [89]: 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 object representations. This is critical to implement structured models that are *compositional*, which ensures that the neural network generalizes 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 (eg. "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.

[15]In our discussion, we focus mainly on binary relations (eg. 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 (eg. A divides B from C), either by using a higher-order graph (eg. a factor graph) or with the help of auxiliary nodes (eg. by adding a 'division node' with binary relations to A, B, and C).

[16]Doumas et al. [75] even argues that objects and relations should in fact use a *shared* 'feature pool' with which both can be described.

[17]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.

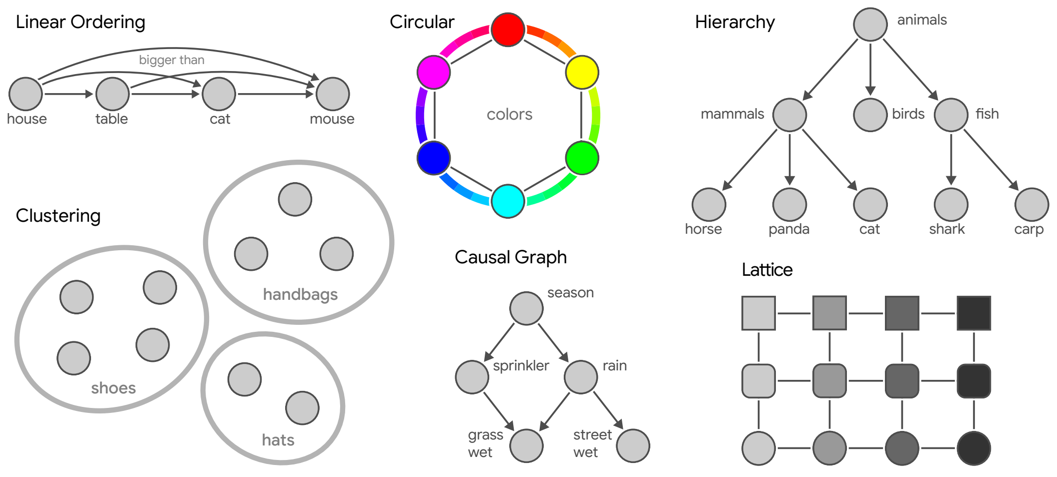


Figure 2.18. Examples of different structural forms [194] that each can be used to define relations among objects and imply different patterns of generalization.

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 2.16, 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; 135), which distinguishes two 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, eg. 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, eg. 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 [159], including 'coordination' (eg. same as) , 'comparison' (eg. larger than), 'hierarchy' (eg. part of) , 'temporal' (eg. after), or 'conditional' (eg. if then). Their corresponding rules for entailment give rise to different *structural forms* [194] among their relations, such as trees, chains, rings, and cliques (see Figure 2.18). 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 (eg. $A = B$ and $B > C$ implies $A > C$).

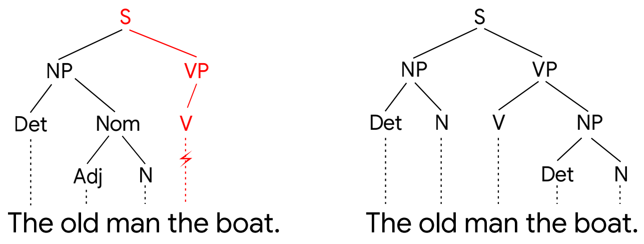


Figure 2.19. Two parse-trees of a garden-path sentence: The intuitive parsing (on the **left**) fails, even though the sentence is grammatically correct (see parse-tree on the **right**).

2.3.2 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, in order to make use of the inferred structure for prediction and behavior.

Relational Responding

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* [89] 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, eg. when applying the distributive law to a given mathematical expression (i.e. turning $a \cdot (b + c)$ into $a \cdot b + a \cdot c$).

A natural choice for facilitating relational responding in a neural network 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 objects representations and relations, which maximally leverages their modularity. These local interactions, which can either be instantaneous (eg. collides with) or persistent (eg. is part of), can facilitate both directed (eg. for causal relations) and bidirectional (eg. 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.

Inferring Structure

Inferring the most desirable structure is an inherently difficult task, which requires making many individual choices at the level of 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 2.19) provides a good example for 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, it is vital that the process of inferring structure is 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 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 2.17 that trigger a comparison of the objects in terms of their masses (as opposed to eg. their relative position or shape). Inferring a relation between objects may also be triggered upon discovering their relation to other objects (eg. 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. relational responding). More generally, the process of inferring structure has to interface closely with the mechanism for variable binding (i.e. for dynamically combining modular object representations and relations in a way that preserves their modularity).

2.3.3 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[18].

Graph Neural Networks

Graph Neural Networks (GNNs; 334, 299, 362) are a promising approach for composition that incorporates the desired structure for relational responding at an architectural level (see Wu et al. 419 for an overview). 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

[18]We note that the problem of inferring structure has also received considerable attention in the causality literature, often specifically focusing on cause-effect discovery (eg. see Hoyer et al. [155], Lopez-Paz et al. [237], Peters et al. [294] or Peters et al. [295] for an overview). 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 [293, 344].

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 incoming relations to update the object representations [20]. 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 a way that follows the dependency structure implied by the relations (relational responding).

Graph Convolutional Networks *Graph Convolutional Networks* (GCNs) are a type of GNN based on a generalization of convolutional neural networks (which operate on grids) to non-Euclidean geometries such as graphs [40]. 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 [130], citation networks [201], 3D surfaces [233], knowledge base completion tasks [337], and bio-chemical modeling [11]. 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 the purpose of 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 [eg. 137, 225] have used a mechanism for coarsening (down-sampling) the graph between layers, to reduce computational complexity, which could provide a mechanism for refining the structure (i.e. structure inference).

Message Passing Neural Networks *Message Passing Neural Networks* (MPNNs; 103) iteratively update the node representations of a given graph by exchanging messages along its edges (until convergence)[19]. 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 *non-linear* function of both adjacent node representations. Hence, edges play a more prominent role in information processing and by explicitly considering pair-wise interactions it is easier to model comparative relations between objects. MPNNs were initially conceived as a generalization of RNNs to graph-structured inputs [362, 110] and have since been adapted to consider modern deep neural networks [228]. A more general framework that accommodates both MPNNs and GCNs was proposed in [20], which additionally includes a global representation of the graph that interacts with all the nodes and edges (and may thereby more easily provide for structure-sensitive operations).

MPNNs have been shown to generalize more systematically (compared to standard neural networks) on many different tasks that require relational responding in terms of objects, including common-sense physical reasoning [50, 21, 175], hierarchical physical reasoning [270, 227, 366], visual question answering [332, 289], abstract visual reasoning [7], natural language processing [373], physical construction [131] or multi-agent interactions [370]. Similar to GCNs, the desired structure may either be specified directly or inferred dynamically based on some heuristic, eg. based on proximity [50, 270] or a language parser [373]. Alternatively, MPNNs have been used to implement a relational inductive bias based on a generic structure,

[19]Recently, *Message Passing Neural Networks* (MPNNs; 103) were extended to allow for continuous updates [70, 234].

eg. by assuming it to be fixed and fully connected (as in Relation Networks; 332). In this case, information can still be exchanged among all the nodes, although the generalization implied by having the correct structural dependencies is lost (eg. 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, eg. using *Variational Autoencoders* (VAEs; 313, 199) [215, 431], or *Generative Adversarial Networks* (GANs) [421]. The other approach is to directly infer the connectivity between nodes iteratively based on message passing, eg. for a fixed number of nodes as in Neural Relational Inference (NRI; 200) or adaptively as in Graph Recurrent Attention Networks (GRANs; 230).

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 soft variable binding) and thereby infer the desired structure for a fixed set of nodes. However, note that this may be computationally inefficient because it still requires computing all possible messages and only affects which of them end up being used in the final summation. Wang et al. [404] makes use of 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 388] may help mitigate this issue and has been successfully applied for relational reasoning about objects [425, 386, 111, 330], citation networks [389], question answering [66], and language modeling [72, 41]. Indeed, Transformers themselves may already be viewed as a kind of graph network [20]. Alternatively, multiple different relations could be learned by also conditioning the message on the receiving object representation when using attention eg. as in *Relational Neural Expectation Maximization* (R-NEM; 385). 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, eg. in pointer networks [393], energy-based models [264], and capsules [326, 210].

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 (eg. 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 agreement 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. A dedicated memory component 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 [349], its dual purpose for representing structure and information processing makes it difficult to learn programs that generalize systematically [216, 60]. Early examples of memory-augmented RNNs [63, 267] use a continuous adaptation of stacks based on the differentiable push and pop operations introduced by Giles et al. [102] (cf. 185 for an alternative implementation). Although a stack-based memory has proven useful for learning about the grammatical structure of language[eg. 63]), 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; 116) offers a more powerful alternative, which can be accessed via generic read and write operations (but see memory networks for a read-only version; 414, 369). 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 (eg. copying or sorting) that generalize to longer input sentences (i.e. more systematically). Additional memory addressing operations, eg. based on the order in which memory locations are accessed (DNC; 117), based on when they were last read [271], or based on a key-value addressing scheme [59] 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 [117]. 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 [138, 186] or incorporate specialized structure for reasoning into the processor, eg. for the purpose of visual question answering using a read-only memory (knowledge base; see 158). Alternatively, certain (Hebbian) forms of fast weights [339] can be viewed as a type of *internal* associative memory based on previous hidden states [13]. TPR-RNN [335] extends this idea by equipping a fast-weight memory with specialized matrix operations inspired by *Tensor Product Representations* (TPRs; 357), which makes it easier to respond to relational queries. In contrast, Reed and de Freitas [307] and Kurach et al. [214] 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.

2.4 Summary

We have analyzed the binding problem of connectionism in terms of three aspects: 1. The ability to simultaneously represent multiple objects in a common format, without interference (representation), 2. the process of forming grounded object representations from raw unstructured inputs (segregation problem), and 3. the capacity to dynamically relate and compose object representations into structured models (composition). In this work we will focus on the first two problems, and mostly ignore the problem of composition. This is because we believe that the problem of segregation is the most neglected and yet also the most critical aspect of the binding problem. It corresponds to the transition from unstructured sensory data to meaningful symbol-like representations, and therefore essentially concerns the symbol-grounding problem [132]. Bridging this gap between physical sensory data and discrete mental representations, may therefore enable the integration and use of many sophisticated symbolic algorithms.

The main challenge tackled in this thesis is segregation, and thus in coping with the immense variability of useful objects that may depend on both task and context. We have argued that this effectively precludes solutions that overly rely on supervision, or domain-specific engineering. This raises the question of how a useful notion of an object can be discovered mainly via unsupervised learning (and later refined based on task specific information). A key part of the answer is to focus on the modularity of objects, which only depends on the statistical structure of the observed data and interfaces directly with the functional role of objects as compositional building blocks. Indeed, evidence suggests that human object perception is based on similar principles [286, 51]. This consideration is the basis for the notion of objects that we propose in Chapter 4, and variations of this theme for the basis for our following methods too. Several other approaches from the machine learning literature have also shown to be able to successfully leverage modularity to learn about objects, for example either by using image segmentation [166] or by attention [43].

Regarding segregation dynamics, we have seen that it is important to provide architectural inductive biases that help with dynamic information routing, eg. in the form of attention or masking specific parts of the input. Consistency and top-down feedback are mostly affected by the interplay between segregation, representation, and composition, and it is difficult to evaluate these properties in isolation. However, in order to facilitate this interaction, it is critical that segregation is part of a fully-differentiable neural approach, which may be most problematic for clustering-based approaches to image segmentation and probabilistic programs based on symbolic models. In the following chapters we adopt the clustering framework in that we rely on iteratively inferred masks at the input level that dynamically route information to a fixed set of (identical) object-slots. And the integration of this clustering-based approach with neural networks is a major theme among Chapters 5 to 7.

Object representations are the product of segregation and the foundation upon which compositional reasoning is built. To effectively connect high-level abstract reasoning with low-level sensory data they must be learned jointly, together with segregation. We have argued that learning object representations requires incorporating architectural inductive biases to ensure a common format and to provide enough flexibility for dynamically separating information. Regarding separation, slot-based approaches offer a simple and minimal approach, while augmentation and TPRs are more difficult to incorporate, yet support more sophisticated use cases. Therefore, we will focus on instance-slot based representations for the work presented in Chapters 4 to 7.

Chapter 3

Background

This chapter reviews some of the mathematical foundations required for the rest of this thesis. Section 3.1 establishes notation and reviews some fundamentals of probability theory and statistics, while Section 3.2 discusses machine learning, and finally Section 3.3 gives a brief introduction to the basics of neural networks. However, a comprehensive treatment of any of these fields is beyond the scope of this thesis. For an excellent introduction to probability theory we refer the reader to [321]. A thorough introduction to machine learning can be found in [34], and regarding neural networks and modern deep learning, we refer the reader to [342] for a detailed survey of its history, or [107] for a general introduction.

3.1 Probability Theory

Probabilty theory is a mathematical framework for reasoning under uncertainty. It plays a central role in machine learning, where sensory information about the world is inherently incomplete and noisy, and the knowledge of any system is limited. We use probability to formally model the task of perceptual grouping, and as a guide for designing systems that operate effectively under uncertainty. This section provides a brief introduction to the relevant concepts and tools.

A *probability space* consists of a sample space (a set Ω of all possible atomic events ω) and a probability measure P. Anything that can happen within our model, corresponds to one element of the sample space, and typically we are interested in a set of outcomes that we call an *event*. The probability measure quantifies how likely each event is by assigning it a real number between 0 (impossible) and 1 (certain). Formally, a *probability space* is a triplet $(\Omega, \mathcal{F}, P)$ consisting of three parts:

1. The *sample space* Ω is an arbitrary non-empty set.
2. A σ-Algebra $\mathcal{F} \subseteq \wp(\Omega)$.
3. A probability measure $P : \mathcal{F} \rightarrow [0, 1]$.

From this definition it immediately follows that the probabilty of Ω is 1 (certain event), and that the probability of $\varnothing$ is 0 (impossible event). The probability of two events A and B happening simultaneously is also called their *joint probability* and we write $P(A, B)$ as a shorthand for $P(A \cap B)$.

3.1.1 Random Variables

A *random variable* is a function $X : \mathcal{F} \to \mathcal{X}$ that maps events to some measurable quantity in $\mathcal{X}$ (eg. a real value). Often this is more useful than working directly with the elements $\omega \in \Omega$. If X takes only finitely or countably many values, then X is called a *discrete random variable*. If, on the other hand, X can take on uncountably many values (eg. if $\mathcal{X} = \mathbb{R}$), then we call X a *continuous random variable*.

We write $P(X = x)$ or $P_X(x)$ as a shorthand for $P(\{\omega \in \Omega \mid X(\omega) = x\})$ and we call the function $P_X : \mathcal{X} \mapsto [0,1]$ the *probability distribution* of X. If X is discret, then P_X is called a *discrete probability distribution* or probability mass function. If X is continuous, then $P_X(x) = 0$ for all $x \in \mathcal{X}$ (which is not very useful). In this case we instead consider the *probability density function* written as $p_X : \mathcal{X} \mapsto \mathbb{R}$ which satisfies $P_X(x \in A) = \int_{x \in A} p_X dx$ for any measurable set A.

For the case of two random variables X and Y we write their joint distribution as $P_{XY}(x,y)$. Restricting a joint distribution to a subset of their variables can be done by summing over all values of the ignored variables:

$$P_X(x) = \sum_i P_{XY}(x, y_i) \quad \text{or} \quad p_X(x) = \int_y p_{XY}(x,y)\,dy.$$

The resulting destribution is also called the *marginal distribution* of X.

3.1.2 Conditional Probabilities

An important concept for dealing with partial information, is that of *conditional probabilities*: The probability of some event A, given that we know another event B has happened. Formally, the *conditional probability of A given B* is defined as:

$$P(A \mid B) := \frac{P(A,B)}{P(B)}.$$

Here $P(A)$ is called the *a priori (or prior) probability* of A, and $P(A \mid B)$ is called the *a posteriori (or posterior) probability* of A given B. Analogously, for two probability distributions P_X and P_Y with $P_Y > 0$, the *conditional distribution of X given Y* is defined as:

$$P_{X|Y}(x \mid y) := \frac{P_{XY}(x,y)}{P_Y(y)}.$$

From this definition we immediately get the important *product rule*:

$$P(A,B) = P(B)P(A \mid B),$$

and its generalization the *chain rule of probabilities*:

$$P(\bigcap_{i=1}^{N} A_i) = P(A_1)P(A_2 \mid A_1)P(A_3 \mid A_1, A_2) \cdots P(A_N \mid \bigcap_{i=1}^{N-1} A_i).$$

Another important corollary is *Bayes' theorem*, which states that for events A and B the following equality holds:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}.$$

Analogously for random variables X and Y:

$$P_{X|Y}(x,y) = \frac{P_{Y|X}(y \mid x)\, P_X(x)}{P_Y(y)} \qquad \text{for all } x, y.$$

An important application of this theorem is to reverse causal knowledge (A causes B) into a belief update based on observed consequences (how does seeing B affect the likelihood of A?).

3.1.3 Expectation and Variance

The *expected value* (or expectation) of a random variable X is the probability-weighted average of all its possible values. It is defined as:

$$\mathbb{E}[X] := \sum_i x_i P_X(x_i),$$

for discrete random variables, and analagously for continuous random variables as:

$$\mathbb{E}[X] := \int_x x\, p_X(x)\, dx.$$

Intuitively the expected value represents the average value X takes over a large number of independent realizations.

It is worth pointing out that the expectation operator $\mathbb{E}$ is linear in the sense that

$$\mathbb{E}[aX + bY] = a\,\mathbb{E}[X] + b\,\mathbb{E}[Y],$$

for any pair of random variables X and Y and constants a and b[1].

The expectation of a random variable X only describes the mean value of a random variable, and does not contain any information about how much its value can vary. To describe the spread of random variable X with expectation $\mu = \mathbb{E}[X]$, we use its variance which is defined as:

$$\mathrm{Var}[X] := \mathbb{E}\left[(X-\mu)^2\right].$$

The variance is always non-negative ($\mathrm{Var}[X] \geq 0$), and invariant to constant offsets:

$$\mathrm{Var}[X + a] = \mathrm{Var}[X].$$

If the values of a random variable X are scaled by a constant amount a, then its variance is scaled by the square of a:

$$\mathrm{Var}[aX] = a^2\,\mathrm{Var}[X].$$

The square root of the variance is also called *standard deviation* and denoted by $\sigma_X := \sqrt{\mathrm{Var}[X]}$.

3.1.4 Correlation and Independence

An important part of probabilistic modeling is to capture the dependencies between events and random variables. Intuitively, this captures the influence that the outcome of one random variable has on another. If two events or random variables do not depend on each other, that

[1]Note that this is a slight abuse of notation because the multiplication and summation on the left operates on functions, while on the right it operates on scalars $\in \mathcal{X}$.

simplifies the model substantially. We say that two events A and B are *independent* (written as $A \perp B$) if their joint probability factorizes:

$$P(A,B) = P(A)\,P(B).$$

Analogously, we say that two random variables X and Y are *independent* iff

$$P_{XY}(x,y) = P_X(x)\,P_Y(y) \qquad \text{for all } x \text{ and } y.$$

If two random variables are independent that implies $P_{X|Y}(x \mid y) = P_X(x)$ provided that $P_Y > 0$. Furthermore the expectation of their product also factorizes $\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]$. Independence between random variables is very useful, but in many cases it is too strong of an assumption. An important and less restrictive notion is that of *conditional independence* (written as $X \perp Y \mid Z$):

$$P_{XY|Z}(x,y \mid z) = P_{X|Z}(x \mid z)\,P_{Y|Z}(y \mid z) \qquad \text{for all } x,\ y, \text{ and } z.$$

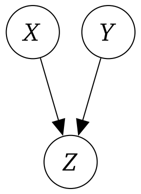
Sometimes, instead of assuming independence, it is more useful to quantify the degree to which they are dependent. To measure the simplest case of linear interactions we generalize the notion of variance to pairs of random variables: Let X and Y be two random variables with expectations $\mu_X = \mathbb{E}[X]$ and $\mu_Y = \mathbb{E}[Y]$. Then their *covariance* is defined as:

$$\mathrm{Cov}[X,Y] := \mathbb{E}[(X-\mu_X)(Y-\mu_Y)].$$

Note that the covariance of a variable with itself is just its variance: $\mathrm{Cov}[X,X] = \mathrm{Var}[X]$.

3.1.5 Graphical Models

Graphical Models are a visual representation of the dependency structure of multiple random variables. This representation offers a clear and intuitive representation and is often helpful for understanding large models. It consists of a directed graph where each node corresponds to a random variable, and every edge represents a dependency. For example, the joint distribution which factorizes as $P_{XYZ}(x,y,z) = P_X(x)P_Y(y)P_{Z|XY}(z \mid x,y)$ could be depicted as:



In general, the distribution corresponding to a graphical model with random variables $\boldsymbol{X} = \{X_i\}_i^N$ is given by:

$$P_{\boldsymbol{X}}(x_1,\ldots,x_N) = \prod_{i=1}^{N} P_{\boldsymbol{X}}(x_i \mid X_{\mathrm{pa}(i)}),$$

where $\mathrm{pa}(i)$ denotes the parents of the node i.

3.1.6 Common Distributions

Bernoulli The *Bernoulli distribution* $\mathcal{B}(\mu)$ is the distribution of a discrete random variable that takes on a value of 1 with probability μ and 0 with probability $1-\mu$. It has an expected value of

μ and a variance of $\mu(1-\mu)$. We write $X \sim \mathcal{B}(\mu)$ as a shorthand for X is distributed according to Bernoulli distribution with parameter μ.

$$\mathcal{B}(x;\mu) = \begin{cases} \mu & x = 1 \\ 1-\mu & x = 0 \end{cases}$$

Categorical A *categorical distribution* $\mathcal{C}(K, \boldsymbol{\mu})$ is the distribution of a discrete random variable X that can take one of K different values. It has two parameters: the number of categories K and their probabilities $\boldsymbol{\mu} = \{\mu_k\}_{k=1}^{K}$ which have to sum to one. The probability distribution of X is thus:

$$\mathcal{C}(x;K,\boldsymbol{\mu}) = \begin{cases} \mu_1 & x = 1 \\ \mu_2 & x = 2 \\ \dots & \\ \mu_K & x = K \end{cases}$$

Uniform A *continuous uniform distribution* describes the distribution of a real-valued random variable that takes only values within a certain interval each with equal probability. It has two parameters $a < b$ which specify the endpoints of the interval, and we write $X \sim \mathcal{U}(a, b)$. Its probability density function is given by:

$$\mathcal{U}(x;a,b) = \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 1-\mu & \text{otherwise} \end{cases}$$

Its expectation is $\mathbb{E}[X] = \frac{1}{2}(a+b)$ and its variance is $\text{Var}[X] = \frac{1}{12}(b-a)^2$.

Normal The *Normal* (or Gaussian) distribution may be the most important continuous distribution for a real-valued random variable. It is written as $\mathcal{N}(\mu, \sigma^2)$ and has two parameters: its mean μ and its variance σ^2. The corresponding probability density function is given by:

$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Mixture Distributions Mixture distributions are probability distributions formed from two or more component distributions. Intuitively, they represent the case where each sample is drawn from one of the component distributions, but it is not observed which one. Formally this consists of a latent (unobserved) categorical random variable $C \sim \mathcal{C}(K, \boldsymbol{\pi})$ and a collection of component random variables $\left\{X^{(k)}\right\}_{k=1}^{K}$. The resulting distribution then corresponds to a weighted mixture of the component distributions:

$$P_X(x) = \sum_{k=1}^{K} P_C(k) P_{X|C}(x \mid k) = \sum_{k=1}^{K} \pi_k P_{X^{(k)}}(x),$$

where the component probabilities π_k are also called *mixing coefficients*. An important example are Gaussian mixture models for which the components are distributed according to Normal distributions:

$$P_X(x) = \sum_{k=1}^{K} \pi_k\, \mathcal{N}(x;\mu_k,\sigma_k^2)$$

3.1.7 Information Theory

Information theory is a branch of probability theory concerned with the quantification, storage and communication of information. Information in this context is defined as the ability to distinguish different outcomes of some (probabilistic) process and is usually measured in *bits*. Formally it is measured as the so-called self-information of an outcome of a discrete random variable X, and can be thought of as the surprise associated with observing the outcome:

$$I_X(x) := -\log_2 P_X(x)$$

Intuitively the self-information corresponds to the number of bits of additional information gained by observing $X = x$ assuming knowledge only about its distribution. This leads to the important concept of *Shannon Entropy* of a random variable, which describes the expected information gain from observing this variable:

$$H[X] := \mathbb{E}[I_X] = -\sum_x P_X(x) \log_2 P_X(x)$$

A useful application of information theory is measuring how different a probability distribution Q is from a reference distribution P: Let P and Q be two separate probability measures defined on the same probability space. Then the *Kullback-Leibler divergence* between P and Q is defined as:

$$D_{\mathrm{KL}}[P \parallel Q] := \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Intuitively, it corresponds to the expected amount of information required to encode outcomes x, if one assumes them to be generated with probabilities $Q(x)$ but in reality they occur with probability $P(x)$. The KL-divergence is non-negative ($D_{\mathrm{KL}}[P \parallel Q] \geq 0$) and is exactly 0 only if P and Q are equal. Note that, in general, the KL-divergence is *not* a metric because it is not symmetric ($D_{\mathrm{KL}}[P \parallel Q] \neq D_{\mathrm{KL}}[Q \parallel P]$) and does not satisfy the triangle inequality ($D_{\mathrm{KL}}[P \parallel Q] + D_{\mathrm{KL}}[Q \parallel R] \ngeq D_{\mathrm{KL}}[P \parallel R]$). A related concept that is often used in machine learning is the so-called *cross-entropy* between two distribution:

$$H[P;Q] = -\sum_x P(x) \log Q(x) = H[P] + D_{\mathrm{KL}}[P \parallel Q]$$

Information can also be used to quantify all the dependencies (including non-linear, unlike covariance) between two random variables X and Y. Given two random variables X and Y, their *mutual-information* corresponds to the KL-divergence between their actual joint distribution, and the factorized distribution which would correspond to them being independent:

$$I[X;Y] := D_{\mathrm{KL}}[P_{XY} \parallel P_X P_Y].$$

The mutual information non-negative and is zero if and only if X and Y are independent:

$$I[X;Y] = 0 \Leftrightarrow X \perp Y$$

We note that there is an important generalization of information theory called *Algorithmic Information Theory* [205]. Rather than defining information in terms of random variables and communication, it centers around computational complexity, and is thus able to define information and randomness for individual observations rather than for distributions. The core concepts of AIT, including Kolmogorov complexity [127] and Solomonoff probability [360] are extremely elegant, but also unfortunately uncomputable, and have to be approximated. For the purpose of this thesis, only probabilistic (Shannon) information theory is relevant, and we defer the extension to algorithmic information theory to future work.

3.2 Machine Learning

An often quoted definition of machine learning is:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." – Tom Mitchell [259]

Machine learning is a large field that spans many very diverse subdisciplines, that each make different assumptions about the domain of experiences, tasks, and performance measures, as well as about the structure of the computer program. Reinforcement Learning (RL), for example, considers the very general case of an agent that observes and acts in an environment and occasionally receives a reward signal. Here, the class of tasks may include any possible sequence of actions in the environment, the experience is considered to be the past sequence of perceptions and actions of the agent, and the performance measure is the accumulated reward. In contrast, the focus of this thesis will be on *supervised learning* and on *unsupervised (or self-supervised) learning*, both from a statistical perspective. This means, that the experience is assumed to consist of a training-set of random samples drawn independent *independent and identically distributed (iid)* from an unknown distribution. This is in sharp contrast to the RL setting, where observations can be influenced by all past actions of the agent, and other changes that may happen in the environment.

3.2.1 Supervised Learning

In supervised learning the task is to map an input x_i to a corresponding output y_i, and the experience takes the form of a training-set of pairs $\{[x_i, y_i]\}_{i=1}^N$, which are assumed to be drawn *independent and identically distributed (iid)* from an unknown distribution P_{XY}:

$$[x_i, y_i] \overset{\text{iid}}{\sim} P_{XY}.$$

A common example of this would be the case of image classifcation, where each $x \in \mathbb{R}^{W*H*3}$ represents an input image, and $y \in \{1, 2, \dots, K\}$ is the corresponding label (one of K classes).

The formulation in terms of a probabilty distribution allows us to use the tools of Bayesian inference to address the learning problem. The idea is to define a parametric class of conditional distributions $P_{Y|X\Theta}$ along with a *prior distribution* over parameters $P_\Theta(\theta)$. The process of learning then corresponds to updating the distribution over parameters based on training examples $[x_i, y_i]$. This new and updated distribution over parameters is called the *posterior distribution* and can be computed using Bayes' theorem:

$$P_{\Theta|YX}(\theta \mid y_i, x_i) = \frac{P_{Y|X\Theta}(y_i \mid x_i, \theta)\, P_\Theta(\theta)}{P_{Y|X}(y_i \mid x_i)} \tag{3.1}$$

$$= \frac{P_{Y|X\Theta}(y_i \mid x_i, \theta)\, P_\Theta(\theta)}{\sum_{\theta'} P_{Y|X\Theta}(y_i \mid x_i, \theta') P_\Theta(\theta')}. \tag{3.2}$$

For a sequence of independent and identically distributed observations $\mathbf{x} = \{x_i\}_{i=1}^N$ and $\mathbf{y} = \{y_i\}^N$ this becomes:

$$P(\theta \mid \mathbf{y}, \mathbf{x}) = \frac{P(\mathbf{y} \mid \mathbf{x}, \theta)\, P_\Theta(\theta)}{\sum_{\theta'} P(\mathbf{y} \mid \mathbf{x}, \theta') P_\Theta(\theta')},$$

where, due to the independence of the training examples, the likelihood term can be written as:

$$P(\mathbf{y} \mid \mathbf{x}, \theta) = \prod_{i=1}^{N} P_{Y|X\Theta}(y_i \mid x_i, \theta).$$

Maximum A-Posteriori Estimate

In general, evaluating the posterior $P_{\Theta|X}$ is difficult and often impossible to do analytically. But in many circumstances, we do not need a full posterior distribution, and it is enough to know the *most likely* value for the parameters θ^* given the observations. This is much easier to compute because the integral term in the denominator of the posterior does not depend on θ and can thus be ignored in the maximization:

$$\theta^* = \operatorname*{argmax}_\theta P(\theta \mid \mathbf{y}, \mathbf{x}) = \operatorname*{argmax}_\theta P(\mathbf{y} \mid \mathbf{x}, \theta)\, P_\Theta(\theta) = \operatorname*{argmax}_\theta \prod_{i=1}^{N} P_{Y|X\Theta}(y_i \mid x_i, \theta) P_\Theta(\theta).$$

This point-estimate θ^* is called a *maximum a-posteriori (MAP) estimate* for θ, the term $P(\mathbf{y} \mid \mathbf{x}, \theta)$ is also known as the likelihood function, and P_Θ is called the prior distribution over parameters. Instead of directly maximizing the posterior, we can equivalently maximize its logarithm without affecting the result, because the logarithm function is monotonically increasing. For independent observations $\{[x_i, y_i]\}_{i=1}^N$ this turns the product of likelihoods into a sum:

$$\theta^* = \operatorname*{argmax}_\theta \sum_{i=1}^{N} \log P_{Y|X\Theta}(y_i \mid x_i, \theta) + \log P_\Theta(\theta),$$

which is much easier to work with in practice and has the added benefit of working with log-likelihood terms, that are also much simpler for many important distributions. Consider, for example, a Gaussian log-likelihood with fixed variance:

$$\log P_{X|\Theta}(x \mid \mu) = \log \mathcal{N}(x; \mu, \sigma^2) = \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) = -\frac{1}{2}\log 2\pi - \log\sigma - \frac{(x-\mu)^2}{2\sigma^2}$$

By also ignoring constant terms the MAP estimate for μ can then be computed simply as:

$$\mu^* = \operatorname*{argmax}_\mu \sum_{i=1}^{N} -\log\sigma - \frac{(x-\mu)^2}{2\sigma^2} + \log P_\Theta(\mu),$$

which, for a sufficiently simple prior P_Θ, corresponds to a quadratic optimization problem, that can be efficiently solved.

Maximum Likelihood Estimate

In the context of machine learning, we often do not wish to, a priori, express any preferences for the parameters. In Bayesian terms this corresponds to an uninformative prior (eg. uniform over all possible values[2]). We can then ignore its contribution when estimating the optimal parameters, resulting in a so-called *maximum likelihood estimate (MLE)*:

$$\theta^* = \operatorname*{argmax}_\theta \log P(\mathbf{y} \mid \mathbf{x}, \theta) = \operatorname*{argmax}_\theta \sum_{i=1}^{N} \log P_{Y|X\Theta}(y_i \mid x_i, \theta).$$

[2]Note that such a prior is technically improper since there is no well-defined uniform distribution over the interval $(-\infty, \infty)$. But this technicality can be ignored as long as the posterior remains well-defined.

This approach of maximum likelihood estimation is an extremely common and powerful paradigm and forms the basis of much of machine learning.

3.2.2 Unsupervised Learning

In the case of unsupervised learning, the training data consists only of inputs $\mathbf{x} = \{x_i\}_{i=1}^N$ which are modelled as *independent and identically distributed (iid)* samples from an (unknown) distribution P_X:

$$x_i \overset{\text{iid}}{\sim} P_X.$$

Usually the goal is to discover interesting structure in the data (eg. clustering) and/or to learn a representation of the data that will be useful to various (though yet unknown) future tasks. Here, we are particularly interested in the generative approach, where the data distribution P_X is assumed to be the marginal of a joint distribution P_{XZ} that includes latent variables Z (unobserved local parameters). Their joint can be specified in terms of a *generative model* $P_{X|Z}$ and a *prior* over latents P_Z:

$$P_X(x) = \sum_z P_{XZ}(x,z) = \sum_z \underbrace{P_{X|Z}(x \mid z)}_{\text{generative model}} \underbrace{P_Z(z)}_{\text{prior}}.$$

The task of representation learning then corresponds to finding a function that maps each x to a suitable value for the latent variable z (i.e. in approximating the posterior $P_{Z|X}$). Superficially, this task looks very similar to the supervised setting, where we wanted to learn the distribution over labels given the input data $P_{Y|X}$. However, the crucial difference here is that the z are unobserved and we thus do not have access to any example values for z.

Variational Inference

Variational inference is a framework for efficiently approximating the posterior distribution $P_{Z|X}$ over the latent variables z given the observations x. The main idea is to introduce a family of distributions Q over latent variables, parametrized by a set of parameters θ: $Q_{Z|\Theta}$. We then find the value for the parameters θ, that makes $Q_{Z|\Theta}$ approximate the desired posterior $P_{Z|X}$ as closely as possible. To measure "closeness" of these two distributions we use their KL divergence (see Section 3.1.7):

$$\begin{aligned}
D_{\text{KL}}[Q_{Z|\Theta} \parallel P_{Z|X}] &= \sum_z Q_{Z|\Theta}(z \mid \theta) \log \frac{Q_{Z|\Theta}(z \mid \theta)}{P_{Z|X}(z \mid x)} && (3.3)\\
&= \mathop{\mathbb{E}}_Q \left[\log \frac{Q_{Z|\Theta}(z \mid \theta)}{P_{Z|X}(z \mid x)} \right] && (3.4)\\
&= \mathop{\mathbb{E}}_Q \left[\log Q_{Z|\Theta}(z \mid \theta) \right] - \mathop{\mathbb{E}}_Q \left[\log P_{Z|X}(z \mid x) \right] && (3.5)\\
&= -\underbrace{\left(\mathop{\mathbb{E}}_Q [\log P_{XZ}(x,z)] - \mathop{\mathbb{E}}_Q \left[\log Q_{Z|\Theta}(z \mid \theta) \right] \right)}_{\text{ELBO}} + \log P_X(x), && (3.6)
\end{aligned}$$

where the first term in the last line is known as the *Evidence Lower BOund (ELBO)*. To see where the name Evidence Lower Bound comes from, one can simply rearange the terms in Equation (3.6) to obtain:

$$\log P_X = \text{ELBO} + D_{\text{KL}}[Q_{Z|\Theta} \parallel P_{Z|X}].$$

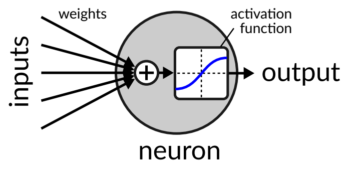


Figure 3.1. Schematic illustration of an artificial neuron.

Remember that the KL-divergence cannot be negative, so the ELBO is indeed a lower bound for $\log P_X(x)$ which is also called the evidence. Since the $\log P_X$ term does not depend on θ, it is enough to maximize the ELBO in order to minimize the KL-divergence between P and Q. This is important, because evaluating $\log P_X(x)$ is usually intractable due to the required summation over all possible values of z. The joint distribution P_{XZ}, on the other hand, avoids this problem and is therefore much easier to compute. The family of distributions $Q_{Z|\Theta}$ can be chosen such that $\mathbb{E}_Q\left[\log Q_{Z|\Theta}(z \mid \theta)\right]$ is tractable too. We have thus turned an intractable Bayesian inference problem, into a tractable optimization problem: namely that of maximizing the ELBO. Variational inference is a powerful framework, which has many important uses in machine learning, including *Expectation Maximization* (EM; 69) and *Variational Autoencoders* (VAEs; 313, 199), both of which are of particular interest in this thesis (see Sections 4.2.3, 5.1.2 and 7.1.2)

3.3 Neural Networks

Artificial Neural Networks (ANNs; or NNs for short) originated as computational models of information processing in biological brains [250, 318, 325] (see [342] for a detailed overview). Broadly speaking, biological brains consist of large network of neurons that communicate electrically via weighted connections called synapses. Similarly, an *Artificial Neural Network* (ANN) consists of many simple computational nodes that are interconnected with weighted connections (also called the weights of the network). The network is activated by stimulating a set of input nodes, and this activation then spreads throughout the network along the weighted connections. However, apart from these superficial similarities, modern ANNs bear little resemblance to their biological counterparts. Importantly, instead of discrete electrical 'spikes', the activation of an ANN node is represented by a continuous numerical value that was originally intended to represent the average firing rate of such spikes over a brief interval. Nonetheless, ANNs enjoy continuing popularity as pattern classifiers and have been proven to work suprisingly well despite, or maye because of, their many simplifications and idealizations. Their focus on learning and parallel computation, have enabled them to benefit optimally from the exponential increase in computational power and available data.

3.3.1 Multi-Layer Perceptrons (MLPs)

Multi-Layer Perceptrons (MLPs; 171, 170) are the simplest type of ANNs and consist of layers of neurons arranged in a linear (feed-forward) topology. Inputs are used to set the activation of the first layer, and this activity is then propagated through the subsequent layers up to the final layer in a process known as the forward pass of the network. Each neuron receives as input all the activations from the previous layer, which it then accumulates as a weighted sum according to the strength of the corresponding connections. The result is then passed through a non-linear activation function (or squashing function) f (see Figure 3.1). Typical choices of activation function include the logistic sigmoid, the hyperbolic tangent, *Rectified Linear*

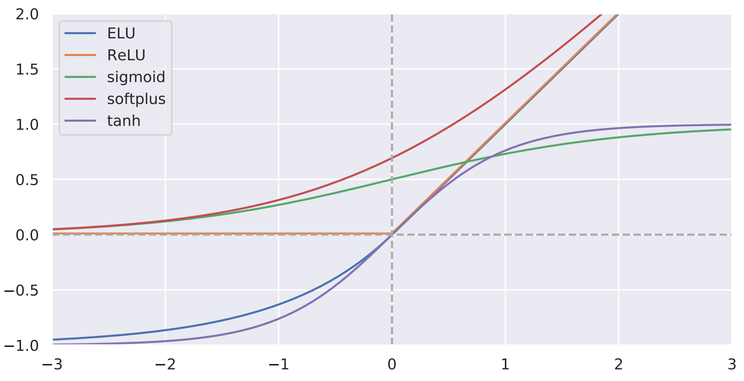


Figure 3.2.
Common activations:

$$\mathrm{sigmoid}(x) = 1/(1+e^{-x})$$

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

$$\mathrm{softplus}(x) = \ln(1+e^{x})$$

$$\mathrm{ReLU}(x) = \max(x,0)$$

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ e^{x}-1 & \text{if } x \leq 0 \end{cases}$$

Unit (ReLU), softplus, and *Exponential Linear Unit* (ELU; 57) (see Figure 3.2). The activation h_i of neuron i is therefore a function of its inputs $\mathbf{x}$ and the corresponding weights $w_{j,i}$ and usually an additive bias term b_i:

$$h_i = f\Big(\sum_j w_{j,i}\, x_j + b_i\Big).$$

Each layer $k > 0$ has its own set of parameters $\theta^{(k)}$ that consists of a matrix of connection weights $W^{(k)}$ and a bias vector $\boldsymbol{b}^{(k)}$. Using matrix notation we can write each layer k as a vector-valued function that maps its inputs $\mathbf{x}^{(k)}$ and parameters $\theta^{(k)}$ to its output activations:

$$\boldsymbol{h}^{(k)} = g^{(k)}(\mathbf{x}^{(k)}; \theta^{(k)}) = f(W^{(k)}\mathbf{x}^{(k)} + \boldsymbol{b}^{(k)}).$$

Here $g^{(0)} = \mathbf{x}$ is called the *input layer*, the intermediate layers $0 < k < K$ are called *hidden layers*, and the final layer $\mathbf{y} = \boldsymbol{g}^{(K)}$ is called the *output layer*. The entire MLP can then be written as the composition of all the layer functions, where the output of each layer becomes the input of the next:

$$\mathbf{y} = \mathrm{MLP}(\mathbf{x}; \boldsymbol{\theta}) = g^{(K)} \circ g^{(K-1)} \circ \ldots \circ g^{(1)}(\mathbf{x}, \boldsymbol{\theta}) = g^{(K)}(g^{(K-1)}(\ldots g^{(1)}(\mathbf{x}, \theta^{(1)}), \theta^{(2)}) \ldots, \theta^{(K)}).$$

Depending on their weights, an MLP can implement a large variety of different functions. In that sense, MLPs are effectively a way to parametrize a rich class of smooth non-linear functions that map input vectors $\mathbf{x}$ to output vectors $\boldsymbol{y} = \boldsymbol{h}^{(K)}$. In fact, it has been shown that they are universal function approximators, in the sense that an MLP with a single (sufficiently large) hidden layer can approximate any continuous function to arbitrary precision [154], given the right set of weights. However, in practice it is often the case that deeper networks are far more efficient in representing the relevant class of functions than shallow networks (eg. [129, 33]).

3.3.2 Training

The fact that neural networks can, in principle, implement any continuous function depending on their weights allows them to be used for a wide variety of tasks. In that way, *Neural Networks* (NNs) convert the problem of searching for a target function, into the problem of finding the right set of weights for a desired task. Typical examples include predicting the (continuous) price of a house based on its features (regression) or categorizing images into cats, dogs, cars etc. (classification). Importantly, for our purposes, they can be used to parametrize families of probability distributions such as the posterior over the latent variables $Q_{Z|X\Theta}$ needed for variational inference. Unfortunately, we cannot directly compute the right set of weights, so we instead cast our search as an optimization problem. Neural Networks are trained by adjusting their weights in a way that it minimizes a objective (or loss) function.

Loss Functions

The loss is a scalar function that plays an important role in the training, because it has to evaluate how well the neural network performs its task in a single number. Often this objective function is based on the *Maximum Likelihood Estimate* (MLE) approach Section 3.2.1 and assumes a probabilistic model parametrized by the neural network. In the case of supervised classification, where the training data consists of pairs of inputs $\mathbf{x}$ (eg. images) and associated categorical labels $t \in 1, \dots, K$ (eg. corresponding to dog, cat, etc.). the underlying probabilistic model is:

$$P(C = k|\mathbf{x}) = y_k = \mathrm{NN}(\mathbf{x})_k.$$

This means that the neural network has K neurons in its output layer and is used to parametrize a conditional categorical distribution over the K possible output classes. In this case a softmax activation function is typically used at the output layer to ensure a valid distribution:

$$y_k = \mathrm{softmax}(h_k) = \frac{e^{h_k}}{\sum_j e^{h_j}}.$$

The corresponding log-likelihood function for classification is given by the average (negative) categorical cross-entropy error:

$$L(\mathbf{y}, \mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N} \log P(C = t_i|x_i) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log(y_{ik}),$$

where the target labels $\mathbf{t}_i$ are assumed to be one-hot vectors with $t_{ik} = 1$ for the correct class k and zero elsewhere. Note that the average cross-entropy error is the same as the KL-divergence between the output by the network $\mathbf{y} = \mathrm{NN}(\mathbf{x})$ and the true distribution $P_{C|\mathbf{x}}$, but with the analytical expectation replaced by an (empirical) average.

For continuous outputs, a common choice is to assume the output of the network to correspond to the mean of a Gaussian distribution $\mu = NN(\mathbf{x})$ with fixed variance σ^2:

$$P(Y = t \mid \mathbf{x}) = \mathcal{N}(t; \mu, \sigma^2) = \mathcal{N}(t; \mathrm{NN}(\mathbf{x}), \sigma^2).$$

The corresponding MLE-based loss function is the well-known Mean-Squared-Error (MSE):

$$L(\boldsymbol{\mu}, \mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{\mu} - \mathbf{t})^2,$$

where σ and other constants have been omitted, since they do not affect the optimization.

Gradient Descent

Given a scalar loss function that measures how well the NN performs on the desired task for set of parameters, we now turn to finding the set of parameters θ^* that minimizes this loss:

$$\theta^* = \underset{\theta}{\mathrm{argmin}}\, L(\mathbf{y}, \mathbf{t}) = \underset{\theta}{\mathrm{argmin}}\, L\left(\mathrm{NN}(\mathbf{x}, \theta), \mathbf{t}\right).$$

This optimization can be performed automatically by many different methods from random search, and evolution, gradient descent, higher order optimization methods such as newtons method, to line search methods, and interior points methods. Here, we will focus on *gradient descent*, an iterative local optimization algorithm for finding a (local) minimum of a loss function. The general idea is to start with a random set of parameters θ, and then repeatedly take small steps in the opposite direction of the gradient of the loss function:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial L(\mathbf{y}, \mathbf{t})}{\partial \theta},$$

either for a fixed number of steps, or until some threshold is reached. For sufficiently small learning rates η, this procedure is guaranteed to converge to a local minimum of the loss function.

For large datasets, gradient descent becomes very inefficient, because it has to processes the entire training set before each update step. Therefore, in practice, a variant called *stochastic gradient descent (SGD)* is much more common. Instead of using the the average gradient for the entire training set, it uses only the gradients from a small (random) subset of examples (a minibatch) for each update. It can be shown that this minibatch-gradient is an unbiased estimate of the full (batch) gradient. In expectation, and with a sufficiently small learning rate, SGD will therefore find the same optimum as batch gradient descent, while being being much faster to compute. Many variations of gradient descent have been developed including SGD with momentum, RPROP [315], AdaGrad [252, 78], and importantly Adam [198].

Backpropagation

Note that the log-likelihood based losses take the form of an average over many per-example-losses, and therefore its derivative too can be written as an average of individual per-example-derivatives:

$$\frac{\partial L(\mathbf{y}, \mathbf{t})}{\partial \theta} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(y_i, t_i)}{\partial \theta}$$

In the following, we will therefore focus on computing the gradient for a single example $[x, t]$ and the corresponding output of the neural network $y = \mathrm{NN}(x, \theta)$.

To efficiently compute the gradient of the loss with respect to the weights of a neural network, we use an approach called backpropagation of error (or backprop for short [409]). It is based on the insight, that both the loss and the NN are compositions of many simple differentiable functions. This makes it possible to use the chain rule of derivatives to compute the partial derivatives of the loss wrt. the parameters θ in terms of many component-wise derivatives. The chainrule has been known for centuries and had been used in earlier work [232, 77, 192], it was Werbos [409] that first developed the neural network specific version of backprop, which later became popular through the work of Rumelhart et al. [324][3].

Applied to the loss, the chain rule gives us:

$$\frac{\partial L(y, t)}{\partial \theta} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial \theta} = \frac{\partial L}{\partial y} \frac{\partial \mathrm{NN}(x, \theta)}{\partial \theta}.$$

Further applications to the neural network result in a long chain of multiplicative terms, one for each constituent function such as the linear projections and activation functions used in MLPs.

[3]See Schmidhuber [342] for a more detailed discussion of the history of backpropagation

Each parameter θ_m may contribute to a different layer and thus require a slightly different chain of derivates, where each of the terms may reappear for many different parameters. The key to the computational efficiency of backpropagation is to systematically store the intermediate values of shared prefixes, instead of recomputing them. In particular, the derivative of the loss with respect to the pre-activations (before the non-linearity) of each layer, play an important role and are known as *deltas*. In the case of an MLP, for example, where the pre-activations $\mathbf{a}^{(l)}$ of the l-th layer are given as:

$$\mathbf{a}^{(l)} = W\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, \tag{3.7}$$

$$\mathbf{h}^{(l)} = f^{(l)}(\mathbf{a}^{(l)}), \tag{3.8}$$

the deltas correspond to $\frac{\partial L}{\partial \mathbf{a}^{(l)}}$ and can be recursively computed from the deltas of layer $l+1$:

$$\boldsymbol{\delta}^{(l)} := \frac{\partial L}{\partial \mathbf{h}^{(l)}} \frac{\partial \mathbf{h}^{(l)}}{\partial \mathbf{a}^{(l)}} = \frac{\partial L}{\partial \mathbf{h}^{(l)}} f'(\mathbf{a}^{(l)}) = \boldsymbol{\delta}^{(l+1)} W^T f'(\mathbf{a}^{(l)}).$$

The gradient wrt. the weights $W^{(l)}$ is then simply:

$$\frac{\partial L}{\partial W^{(l)}} = \boldsymbol{\delta}^{(l)} \frac{\partial \mathbf{a}^{(l)}}{\partial W^{(l)}} = \boldsymbol{\delta}^{(l)} \otimes \mathbf{h}^{(l-1)},$$

where $\otimes$ denotes the outer product. The resulting algorithm has the same computational complexity as a forward-pass that propagates the input-activations through the network to compute its output. As the name suggests, backpropagation operates in a backward fashion, that starts at the loss and propagates the errors backwards through the layers towards the input. Modern neural networks are usually implemented using auto-differentiation software such as Tensorflow [1], Pytorch [291], or JAX [38], which can automatically construct an efficient backward pass for any specified computational graph composed of differentiable building blocks.

3.3.3 Architectures

So far, the only neural network architecture we have discussed is the MLP, which is simply a linear chain of fully connected layers. However, it is worth noting that, in general, ANNs are not restricted to a linear (chain-like) architecture, but can be formed from arbitrary directed acyclic graphs (DAGs). A vast multitude of different architectures have been proposed and studied in the literature. Far too many to list them here, but it is worth mentioning a few important layer types from which architectures can be constructed.

Recurrent Neural Network

Recurrent neural networks (RNNs; 81, 184, 222) are an extension of normal (feed-forward) neural networks for sequential data. They introduce recurrent connections that allow the activations of a particular timestep to depend on the activations of the previous timestep. In its simplest form, which we will call *Simple Recurrent Network* (SRN; 316), it takes the shape of a fully connected recurrent layer:

$$\mathbf{h}^t = f(W\mathbf{x}^t + R\mathbf{h}^{t-1} + \mathbf{b}),$$

where $\mathbf{h}^t$ and $\mathbf{x}^t$ are the hidden activation and input at timestep t respectively, f is the activation function, W and R are weight matrices and $\mathbf{b}$ are the biases. This construction turns out to be far more powerful than a simple MLP layer, due to the fact that it can use **h** as an internal memory. In principle, this allows *Recurrent Neural Networks* (RNNs; 250, 368, 316, 410) to model temporal dependencies of arbitrary length, and in fact, Siegelmann and Sontag [349] have shown that a sufficiently large RNN can perform arbitrary computations (i.e. that RNNs are Turing-complete). However, in practice it is difficult to train them to discover dependencies that span more than a few timesteps, due to the so-called *vanishing gradient problem* [148, 151].

Recurrent neural networks are trained with a temporal extension of backprop called *backprop trough time (BPTT)* [316, 410, 415]. The basic idea is to mentally "unfold the RNN in time", by (virtually) turning each timestep into its own layer and thus converting the RNN into a very deep feed-forward neural network. Backpropagation can then be applied as usual to this new very deep network to compute the gradient of the loss with respect to the weights. The only difference is that each weight now appears at many places in the same network (weight-sharing), and thus the contributions from each of those occurrences have to be summed up to receive the final gradient.

Long Short-Term Memory

Recurrent neural networks with Long Short-Term Memory (*Long Short-Term Memorys* (LSTMs; 149)) have emerged as an effective and scalable alternative to SRNs. LSTMs are both general and effective at capturing long-term temporal dependencies. They do not suffer from the optimization hurdles that plague SRNs [151] and have been used to advance the state of the art for many difficult problems. This includes handwriting recognition [113, 296, 73] and generation [112], language modeling [426] and translation [240], acoustic modeling of speech [327], speech synthesis [85], protein secondary structure prediction [361], analysis of audio [244], and video data [74] among others.

The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell. Most modern studies incorporate many improvements that have been made to the LSTM architecture since its original formulation [149, 150]. The LSTM setup most commonly used in literature was originally described by Graves and Schmidhuber [115]. We refer to it as *vanilla LSTM* and use it a reference for comparison of all the variants. The vanilla LSTM incorporates changes by Gers et al. [101] and Gers and Schmidhuber [100] into the original LSTM [150] and uses full gradient training. A schematic of the vanilla LSTM block can be seen in Figure 3.3. It features three gates (input, forget, output), block input, a single cell (the Constant Error Carousel), an output activation function, and peephole connections. The output of the block is recurrently connected back to the block input and all of the gates. However, LSTMs are now applied to many learning problems which differ significantly in scale and nature from the problems that these improvements were initially tested on. In Greff, Srivastava, Koutník, Steunebrink and Schmidhuber [121] we conducted a systematic study of the utility of various computational components which comprise LSTMs (see Figure 3.3).

Let $\mathbf{x}^t$ be the input vector at time t, N be the number of LSTM blocks and M the number of inputs. Then we get the following weights for an LSTM layer:

- Input weights: $\mathbf{W}_z, \mathbf{W}_s, \mathbf{W}_f, \mathbf{W}_o \in \mathbb{R}^{N\times M}$
- Recurrent weights: $\mathbf{R}_z, \mathbf{R}_s, \mathbf{R}_f, \mathbf{R}_o \in \mathbb{R}^{N\times N}$
- Peephole weights: $\mathbf{p}_s, \mathbf{p}_f, \mathbf{p}_o \in \mathbb{R}^{N}$
- Bias weights: $\mathbf{b}_z, \mathbf{b}_s, \mathbf{b}_f, \mathbf{b}_o \in \mathbb{R}^{N}$

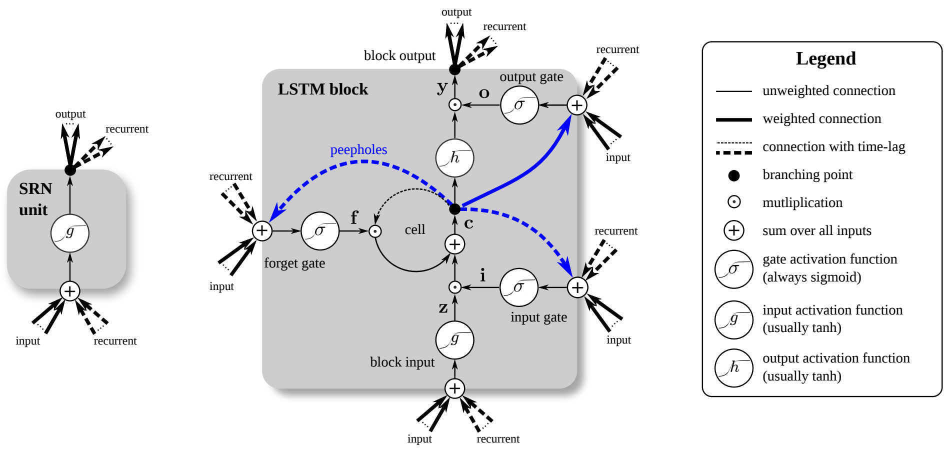


Figure 3.3. Detailed schematic of the SRN unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

Then the vector formulas for a vanilla LSTM layer forward pass can be written as:

$$
\begin{aligned}
\bar{\mathbf{z}}^t &= \mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z \\
\mathbf{z}^t &= g(\bar{\mathbf{z}}^t) && \textit{block input} \\
\bar{\mathbf{i}}^t &= \mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i \\
\mathbf{i}^t &= \sigma(\bar{\mathbf{i}}^t) && \textit{input gate} \\
\bar{\mathbf{f}}^t &= \mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f \\
\mathbf{f}^t &= \sigma(\bar{\mathbf{f}}^t) && \textit{forget gate} \\
\mathbf{c}^t &= \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t && \textit{cell} \\
\bar{\mathbf{o}}^t &= \mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o \\
\mathbf{o}^t &= \sigma(\bar{\mathbf{o}}^t) && \textit{output gate} \\
\mathbf{y}^t &= h(\mathbf{c}^t) \odot \mathbf{o}^t && \textit{block output}
\end{aligned}
$$

Here, functions σ, g and h are point-wise non-linear activation functions. The *logistic sigmoid* ($\sigma(x) = \frac{1}{1+e^{-x}}$) is used as activation function of the gates and the *hyperbolic tangent* ($g(x) = h(x) = \tanh(x)$) is usually used as the block input and output activation function. The operator $\odot$ denotes the point-wise multiplication of two vectors.

Highway and Residual Networks

Training of very deep neural networks with dozens or hundreds of layers, suffers from the same *vanishing gradient problem* as SRNs. Both Highway and Residual networks address this problem by improving the error flow via identity skip connections that allow units to copy their inputs on to the next layer unchanged. This design principle was originally introduced in Long Short-Term Memory (LSTM) recurrent networks [150] and mathematically these architectures correspond to a simplified LSTM network, "unrolled" over time.

In Highway Networks, for each unit there are two additional gating units, which control how much (typically non-linear) transformation is applied (transform gate T) and how much to just copy of the activation from the corresponding unit in the previous layer (carry gate C). Let $H(\mathbf{x})$ be a nonlinear parametric function of the inputs, $\mathbf{x}$, (typically an affine projection followed by pointwise non-linearity). Then a traditional feed-forward network layer can be written as:

$$y(\mathbf{x}) = H(\mathbf{x}). \tag{3.9}$$

By adding two additional units, $T(\mathbf{x})$ and $C(\mathbf{x})$ a Highway layer can be written as:

$$y(\mathbf{x}) = H(\mathbf{x}) \cdot T(\mathbf{x}) + \mathbf{x} \cdot C(\mathbf{x}). \tag{3.10}$$

Usually this is further simplified by coupling the gates, i.e. setting $C(\mathbf{x}) = 1 - T(\mathbf{x})$:

$$y(\mathbf{x}) = H(\mathbf{x}) \cdot T(\mathbf{x}) + \mathbf{x} \cdot (1 - T(\mathbf{x})). \tag{3.11}$$

ResNets simplify the Highway networks approach by reformulating the desired transformation as the input plus a residual $F(\mathbf{x})$. The rationale behind this is that it is easier to optimize the residual form than the original function. For the extreme case where the desired function is the identity, this amounts to the trivial task of pushing the residual to zero:

$$y(\mathbf{x}) = F(\mathbf{x}) + \mathbf{x}. \tag{3.12}$$

As with Highway networks, Residual networks can be viewed as unfolded recurrent neural networks of the particular mathematical form (one with an identity self-connection) of an LSTM cell. This has been explicitly pointed out by [229], who also argue that this could allow Residual networks to emulate recurrent processing in the visual cortex and thus adds to their biological plausibility. Setting $F(\mathbf{x}) = T(\mathbf{x})[H(\mathbf{x}) - \mathbf{x}]$ converts Equation 3.12 to Equation 3.11 showing that both formulations differ only in the precise functional form for F. Alternatively, Residual networks can be seen as a particular case of Highway networks where $C(\mathbf{x}) = T(\mathbf{x}) = \mathbf{1}$ and are not learned.

Chapter 4

Reconstruction Clustering

We are interested in learning object representations in an unsupervised fashion. That is to split a given image into objects, each of which should be described by a distributed representation. This is unlike most work on segmentation, because we do not want to provide any supervision on what or where the objects are. But that raises an important question: What could be an unsupervised notion of an object? And is there even such a thing? In this chapter I argue for an information theoretic notion of objects and develop a simple proof-of-concept system that learns in an unsupervised way to segregate an image into objects, based only on the statistics of the training images. The goal here is to provide some intuition as well as lay the foundation for a formal framework. It is based on the publication [122].

4.1 What is an Object?

Most work on object detection with neural networks relies on supervision through a combination of class labels, bounding boxes, and/or segmentation masks. But humans can detect and segment objects which they have never seen before. Consider, for example the made-up object called *Greeble* shown in Figure 4.1: You can not only segment it from its background, but also say something about its properties (color, shape, texture, etc.). The fact that you are able to generalize your object perception to this unfamiliar object means that it cannot rely on explicit supervision. Rather, you had to rely on general cues like the ones studied in Gestalt Psychology.

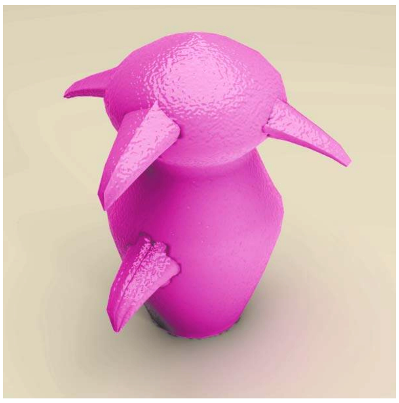


Figure 4.1. Picture of a *"Greeble"* [97].

4.1.1 Gestalt Psychology

Gestalt Psychology was arguably the first systematic investigation of human object perception. Inspired by the seminal work of Wertheimer [412], they argue that the perception of wholes (or Gestalten[1]) can not be described as a simple composition of more primitive percepts. This

[1] *"Gestalten"* is plural of the German word *"Gestalt"* meaning "form" or "shape".

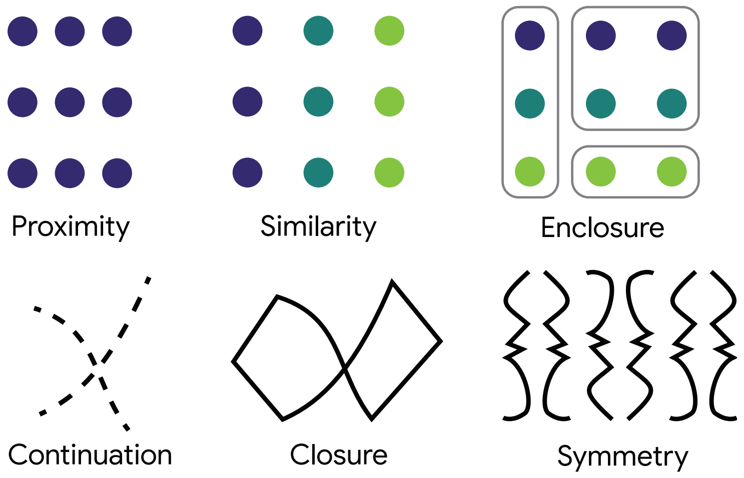


Figure 4.2. Illustration of several Gestalt laws of visual perception. Note how the different cues influence which elements are perceived as belonging together.

holistic view of perception, was later summarized by Kurt Koffka as: "The whole *is other* than the sum of its parts" [202][2], Their concept of a Gestalt closely resembles our notion of object. For an excellent in-depth overview of Gestalt Psychology we encourage the reader to refer to [400, 401].

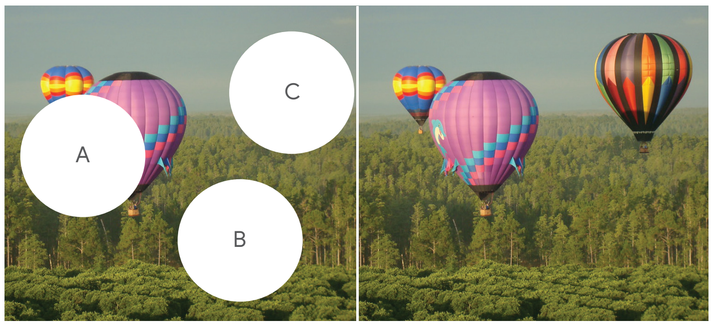
The best-known results of Gestalt research are their principles of perceptual grouping (also known as Gestalt Laws; see Figure 4.2 for an overview). They describe which stimulus cues influence the perceived grouping of a set of discrete elements [413, 400]. They include among others: law of proximity (closeby pieces tend to be grouped together), the law of similarity (similar pieces tend to be grouped together), the law of closure (grouping prefers to form closed contours), the law of symmetry (grouping prefers to form symmetric objects) and the law of common fate (what moves together groups together). Several other Gestalt laws have been found over the years. While Gestalt Psychology initially focussed on the visual domain only, it has later been extended to other sensory modalities including audio, tactile and even olfaction.

Note that the laws of proximity and common fate can be seen as special cases of the law of similarity (with position and movement resp. being the compared attributes). Similarly, some have argued that the Gestalt Laws are all special cases of a single information theoretic grouping principle [134]. The idea is that a *good Gestalt* is one with a lot of internal redundancy [10], and thus that the likelihood of a particular grouping is inversly proportional to the amount of information required to describe it [147]. There is disagreement about how to quantify information and the issue of simplicity vs likelihood has been debated extensively, though they might turn out to be identical [51]. For our purpose the existence of these general principles, and their prevalence in multiple sensory domains is very interesting. It makes plausible the idea of a general segregation mechanism that can generalize to novel objects.

4.1.2 Grouping by Predictability

Here we use mutual predictability (also called pointwise mutual information) of the pixels to formalize this notion of "internal redundancy". Intuitively, knowing about some pixel values that belong to an object helps in predicting the other pixels of the same objects, but not with predicting other objects (see Figure 4.3). Knowledge about the predictability structure is precisely what is

[2]Frequently misquoted as "The whole is *greater* than the sum of its parts".



(a)

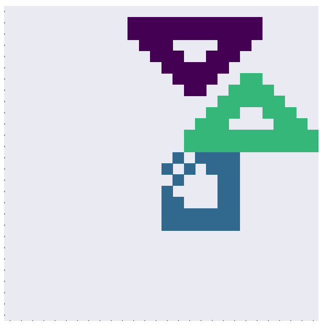

(b)

Figure 4.3. An illustration of intra-object predictability. a) Notice that when a ballon is partially occluded (A) the rest of it can still be inferred, but not when it is fully occluded (B). b) The same holds for corrupted pixels: The missing pixels of the square can easily be predicted using its remaining pixels, but not from pixels constituting other objects.

needed in order to remove corruption from an image. An example can be seen in Figure 4.3b where the corrupted pixels in the bottom left corner of the square can be reconstructed from knowledge about the rest of the square, but not from any of the triangles. So we define an object as a group of pixels that help in predicting each other, but do not carry information about pixels outside of that group.

Based on this insight, we propose to use a *Denoising Autoencoder* (DAE; 27, 391) to measure predictability. The DAE is trained to remove corruption from images of single objects and thus learns a local model of the data generating distribution [391, 31].

4.2 Method

With this unsupervised notion of an object, we are now ready to set up our perceptual grouping system. The idea is to use the predictability of pixels to divide them into several groups. We start from a random split of the input image, and then use an iterative clustering method to group pixels together that help predict each other. At the end the resulting clusters will then (hopefully) correspond to objects. We measure predictability of pixels using a DAE that was trained on images of the desired objects, and formalize the clustering as a variation of *Expectation Maximization* (EM; 69). Thus predictability, as we use it here, is derived from the structure of the underlying data-distribution. The representation computed by the encoder of the DAE serves as the object representation and thus as the parameters of the corresponding cluster.

4.2.1 Denoising Autoencoder

Let g be the encoder and f be the decoder of a *Denoising Autoencoder* (DAE; 27, 391), such that $\mathbf{z} = g(\mathbf{x})$ is the encoded representation of input $\mathbf{x}$. The decoded output $\boldsymbol{\mu} = f(\mathbf{z})$ corresponds to the means of a pixel-wise Bernoulli distribution $\mathcal{B}$. The DAE is trained to remove corruption from images of single objects and thus learns a local model of the data generating distribution [391, 31]. After training the same DAE is used for each of the clusters to get predictions μ_i for pixel i, where the object is represented by $\mathbf{z}$. This corresponds to the following likelihood:

$$P_{X|Z}(\mathbf{x} \mid \mathbf{z}) = \prod_{i=1}^{D} P_{X|Z}(x_i \mid \mathbf{z}) = \prod_{i=1}^{D} \mathcal{B}(x_i; \mu_i) = \prod_{i=1}^{D} \mu_i^{x_i}(1-\mu_i)^{1-x_i} \tag{4.1}$$

Here the x_i's are assumed to be independent given $\boldsymbol{\mu}$.

4.2.2 Spatial Mixture Model

We model the input as a *spatial mixture model*: Each pixel x_i of the input image $\mathbf{x}$ is generated by one of K independent objects with representations $\mathbf{z}_k$ according to Equation (4.1). The pixels are independent (given $\mathbf{z}$), but not identically distributed (unlike a regular mixture model). We introduce a latent pixel-wise categorical random variable $C_i \in \{1, 2, \ldots, K\}$ determines which of the K objects the pixel belongs to. Let the C_i's be independent and identically distributed and its prior distribution be given by[3]:

$$\pi_k := P(C_i = k) = \frac{1}{K}, \quad \text{for all } 1 \le k \le K \text{ and } 1 \le i \le D. \tag{4.2}$$

The (complete data) log likelihood for the image $\mathbf{x}$ is then

$$\mathcal{L}(\mathbf{z}; \mathbf{x}, \mathbf{c}) := \log P_{X|Z,C}(\mathbf{x} \mid \mathbf{z}, \mathbf{c}) = \sum_{i=1}^{D} \log P_{X|Z,C}(x_i \mid \mathbf{z}, c_i) = \sum_{i=1}^{D}\sum_{k=1}^{K} \delta_{c_i=k} \log P_{X|Z}(x_i \mid \mathbf{z}_k), \tag{4.3}$$

where δ is the indicator function:

$$\delta_{c_i=k} = \begin{cases} 1 & c_i = k \\ 0 & \text{else} \end{cases}.$$

Given an image $\mathbf{x}$, we would like to to use maximum likelihood estimation to infer the object representations $\mathbf{z}_k$. Unfortunatly the assignment to pixels $\mathbf{c}$ is unknown, so we need to marginalize over all possible values of $\mathbf{c}$. With that we get that the (incomplete) data log-likelihood $\mathcal{L}(\mathbf{z} \mid \mathbf{x})$ is a pixel-wise mixture model with mixing coefficients π_k:

$$\begin{aligned}
\mathcal{L}(\mathbf{z} \mid \mathbf{x}) &= \log P_{X|Z}(\mathbf{x} \mid \mathbf{z}) && (4.4)\\
&= \sum_{i=1}^{D} \log P_{X|Z}(x_i \mid \mathbf{z}) && (4.5)\\
&= \sum_{i=1}^{D} \log \sum_{k=1}^{K} P(C_i = k) P_{X|Z}(x_i \mid \mathbf{z}_k) && (4.6)\\
&= \sum_{i=1}^{D} \log \sum_{k=1}^{K} \pi_k P_{X|Z}(x_i \mid \mathbf{z}_k) && (4.7)\\
& && (4.8)
\end{aligned}$$

[3] For the sake of simplicity we are assuming a fixed uniform prior for the C_i's (mixing coefficients), but it is straightforward to adjust the framework to other choices and even iteratively updating the mixing coefficients.

4.2.3 Expectation Maximization

Direct maximization of $\mathcal{L}$ wrt. $\mathbf{z}$ is difficult due to the summation inside the logarithm, so we turn to the iterative EM algorithm. It is based on the observation that if we knew the values of either $\mathbf{z}$ or $\boldsymbol{c}$, optimizing the complete log likelihood from Equation (4.3) with respect to the other would be straight-forward. So it divides the optimization into two steps that alternate between optimizing $\mathbf{z}$ (M-Step) and $\boldsymbol{c}$ (E-Step) where in ech step we use our current best estimate for the other quantity.

E-Step In the E-Step we assume a given estimate for $\boldsymbol{z}$ and use it to calculate the posterior for the assignment:

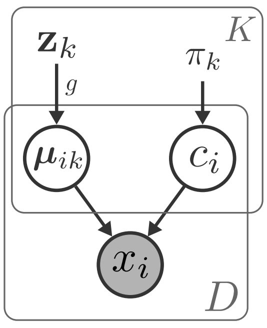
$$\begin{aligned}
m_{i,k} &:= P_{C|X,Z}(k \mid x_i, \mathbf{z}) && (4.9)\\
&= \frac{P_{X|Z,C}(x_i \mid \mathbf{z}, k) P_{C|Z}(k \mid \mathbf{z})}{P_{X|Z}(x_i \mid \mathbf{z})} && (4.10)\\
&= \frac{P_{X|Z}(x_i \mid \mathbf{z}_k)\pi_k}{\sum_{k=1}^{K} \pi_k P_{X|Z}(x_i \mid \mathbf{z}_k)} && (4.11)\\
&= \frac{P_{X|Z}(x_i \mid \mathbf{z}_k)}{\sum_{k=1}^{K} P_{X|Z}(x_i \mid \mathbf{z}_k)} && (4.12)\\
& && (4.13)
\end{aligned}$$

Here, the second line is just an application of Bayes' rule, the third line follows from the fact that C is assumed to be independent of Z, and the final line is a simplification using the fact that all $\pi_k = 1/K$.

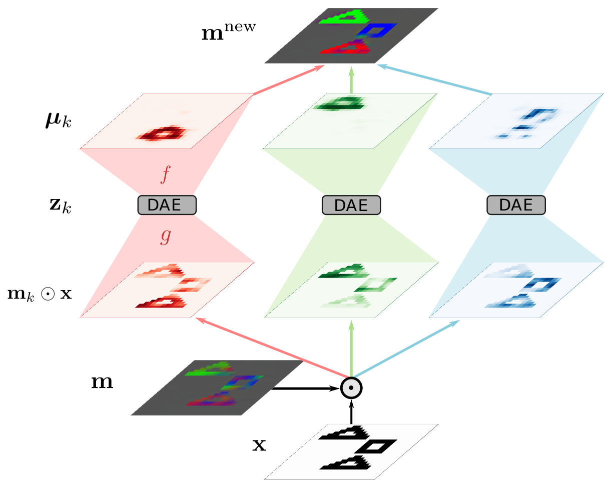
M-Step Now we need the $\mathcal{Q}$ value used in EM which is defined as the expectation of the complete data log-likelihood with respect to the posterior of $\mathbf{C}$ given the data and a previous estimate for the parameters $\mathbf{z}'$:

$$\begin{aligned}
\mathcal{Q}(\mathbf{z}, \mathbf{z}') &= \mathop{\mathbb{E}}_{\mathbf{C}|X,Z'} [\mathcal{L}(\mathbf{z}; \mathbf{x}, \boldsymbol{c})] && (4.14)\\
&= \sum_{i=1}^{D} \sum_{k=1}^{K} \mathop{\mathbb{E}}_{C|X,Z'} [\delta_{c_i=k} \log P_{X|Z}(x_i \mid \mathbf{z}_k)] && (4.15)\\
&= \sum_{i=1}^{D} \sum_{k=1}^{K} P_{C|X,Z}(c_i \mid \mathbf{x}, \mathbf{z}'_k) \log P_{X|Z}(x_i \mid \mathbf{z}_k) && (4.16)\\
&= \sum_{i=1}^{D} \sum_{k=1}^{K} m_{i,k} \log P_{X|Z}(x_i \mid \mathbf{z}_k) && (4.17)
\end{aligned}$$

The maximum of $\mathcal{Q}$ with respect to $\boldsymbol{\mu}$ would be trivially obtained by setting $\mu_{i,k} = x_i$ for all k. This is due to the fact that the problem is actually ill-posed in the sense that we have K



(a)



(b)

Figure 4.4. (a) The assumed probabilistic structure. (b) A schematic illustration of one iteration of the RC algorithm.

parameters to fit for each pixel. So there are infinitely many settings of $\boldsymbol{\mu}$ which achieve the optimal log likelihood of the data, and we depend on the encoder f to sufficiently restrict our modelling capacity.

So in the M-step of EM we aim to maximize $\mathcal{Q}(\mathbf{z}, \mathbf{z}')$ over all choices of $\mathbf{z}$:

$$\mathbf{z}^{new} = \underset{\mathbf{z}}{\operatorname{argmax}}\, \mathcal{Q}(\mathbf{z}, \mathbf{z}') \tag{4.18}$$

Unfortunately this maximization is intractable due to the non-linear dependence of $\boldsymbol{\mu}$ on $\mathbf{z}$.

4.2.4 Reconstruction Step

At this point we deviate from the EM formulation, and instead of maximizing $\mathcal{Q}$ wrt. $\mathbf{z}$, we use the encoder g of the DAE. We compute a partial image $\tilde{\mathbf{x}}_k = \boldsymbol{m}_k \odot \mathbf{x}$ for each cluster k by (soft) masking out all pixels that are not assigned to it. The R-step then applies the encoder g to this partial image to infer an updated object representation for each cluster. The intuition is that DAE then denoises the “corruption” caused by the cluster assignments. The R-Step is thus given by the following formula, where $\odot$ denotes point-wise multiplication:

$$\mathbf{z}_k^{new} = g(\boldsymbol{m}_k \odot \mathbf{x}), \tag{4.19}$$

Unfortunately this step can not be guaranteed to increase the expected log-likelihood, because only in expectation does the DAE map from regions of low likelihood to regions of higher likelihood. Moreover, this property only holds for the whole image and not for all subsets of pixels. Thus, convergence can’t be proven and *Reconstruction Clustering* (RC; 122) is not a proper *Expectation Maximization* (EM; 69) algorithm. Nevertheless, empirical results show that convergence does occur reliably (Section 4.4.2).

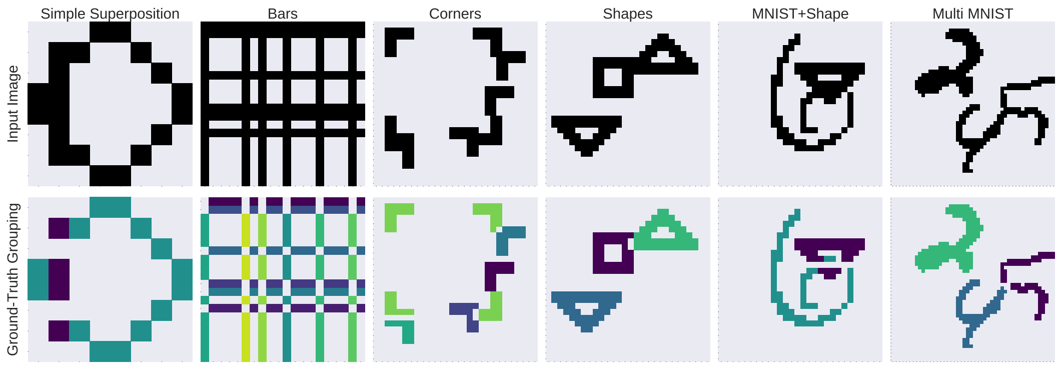


Figure 4.5. One example from each of the six datasets. The input images are shown on the top row with the corresponding ground-truth grouping below.

4.2.5 Putting it together

The full *Reconstruction Clustering* (RC; 122) thus starts by training a DAE to denoise corrupted images of single objects on a dataset. Then, for each multi-object image we perform the following steps:

1. Randomly initialize $\boldsymbol{m}$
2. *(R-step)* Apply the autoencoder to the each of the K images that are assigned to the clusters to get a new estimate of $\mathbf{z}_k$ object representations.
3. *(E-step)* Re-assign the pixels to the clusters according to their reconstruction accuracy.
4. repeat steps 2 and 3 until convergence

4.3 Experiments

We evaluated RC on a series of artificially generated datasets consisting of binary images of varying complexity. For each dataset, a DAE was trained to remove salt&pepper noise on images with single objects. The autoencoders used were fully-connected feed-forward *Neural Networks* (NNs) with a single hidden layer and sigmoid output units. A random search was used to select appropriate hyperparameters (see Appendix for details). The best DAE obtained for each dataset was used for reconstruction clustering on 1000 test images containing multiple objects, and the binding performance was evaluated based on groud-truth object identities. All the code for these experiments (including the creation of the datasets and figures) is available online at GitHub.com/Qwlouse/Binding.

4.3.1 Datasets

We evaluate RC using six simple datasets (see Figure 4.5):

Simple Superposition A collection of simple pixel patterns two of which are superimposed. Taken from [304]. This is a simple dataset with no translations, but significant overlap between patterns.

Shapes Taken from [309]. Three shapes (△▽□) are randomly placed in an image (possibly with overlap). This dataset tests binding of shapes under translation invariance and varying overlap.

Bars Introduced by [90] to demonstrate unsupervised learning of independent components of an image. We use the variant from [309] which employs 6 horizontal, and 6 vertical lines placed in random positions in the image.

Corners This dataset consists of 8 corner shapes placed in random orientations and positions, such that 4 of them align to form a square. It was introduced by [309] to demonstrate that spatial connected-ness is not a requirement for binding.

MNIST+Shape Another dataset from [309], which combines a random shape from the shapes dataset with a single MNIST digit. This dataset is useful to investigate binding multiple types of objects.

Multi-MNIST Three random MNIST digits are randomly placed in a 48×48 image. It provides a more challenging setup with multiple complex objects.

4.3.2 Evaluation

Since the data is generated, a ground-truth segmentation for each image is available. We evaluated performance by measuring the *Adjusted Mutual Information* (AMI; 392) score between the true segmentation and the result of RC. This score measures how well two cluster assignments agree and takes a value of 1 when they are equivalent, and 0 when their agreement corresponds to chance level. For evaluation we count only those pixels that unambiguously belong to a single object and ignore background pixels and regions of overlap.

4.3.3 Training Details

All experiments have been performed with the brainstorm library and were organized and logged using sacred. The code for this paper can be found on GitHub.

For the DAEs we use a simple 3 layer fully connected neural network, with sigmoid activation at the output layer. The network was trained with *Stochastic Gradient Descent* (SGD) at a minibatch size of 100 to remove salt& pepper noise added to inputs. Training was stopped when the validation loss didn't decrease for more than 10 consequtive epochs. We performed a random search with 100 runs for each dataset to determine a good set of hyperparameters. For each run we randomly sampled from the following parameters:

- learning rate log-uniform from $[10^{-3}, 1]$
- Amount of Salt& Pepper Noise from $[0.0, 0.1, \dots, 0.9]$
- hidden layer size from $[100, 250, 500, 1000]$
- hidden layer activation function from [rel, sigmoid, tanh]

The best network configurations found by that search can be found in Table 4.1.

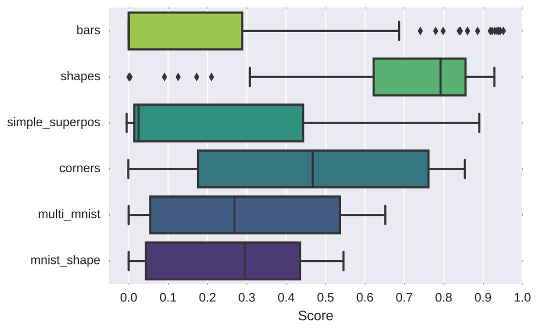


Figure 4.6. Summary of the scores achieved during the random search

Dataset	learning rate	# hidden units	activation	salt&pepper	score
bars	0.768015	100	ReL	0.0	0.951809
corners	0.001920	100	ReL	0.0	0.853866
multi_mnist	0.011362	1000	ReL	0.6	0.651657
mnist_shape	0.031685	250	sigmoid	0.6	0.545559
shapes	0.083147	500	tanh	0.4	0.928792
simple_superpos	0.366627	100	ReL	0.1	0.890472

Table 4.1. Configuration of the best network for each dataset as found by the random search.

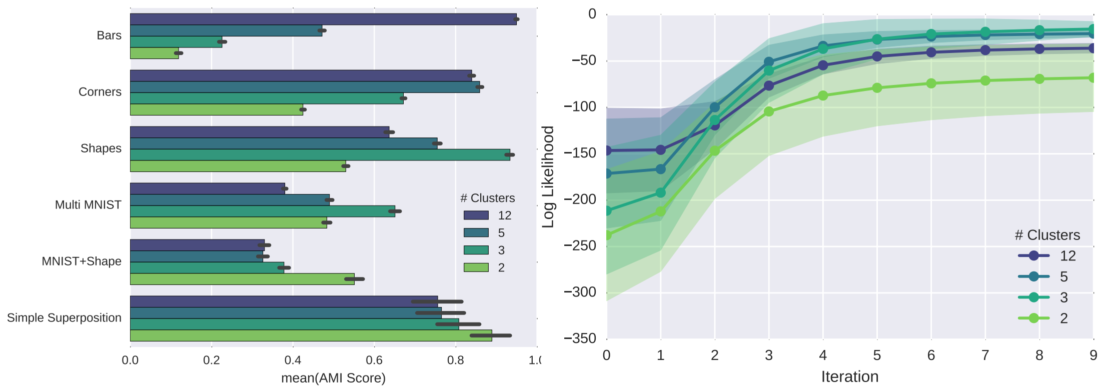
4.4 Results

4.4.1 Scores

Figure 4.7a shows the mean scores obtained using RC for each dataset averaged over 100 runs. Scores obtained with different choices of the number of clusters K. Results are consistent across runs, hence the standard deviations are very low and barely visible. The optimal number of clusters is two for *Simple Superposition* and *MNIST+Shape*, three for *Multi MNIST* and *Shapes*, five for *Corners*, and 12 for *Bars*. Scores are higher than 0.5 for all datasets and higher than 0.8 for four out of the six datasets demonstrating the ability of RC to successfully bind objects together.

4.4.2 Convergence

Figure 4.7b shows the convergence of the mean log-likelihood over RC iterations on the *shapes* dataset. Convergence is quick, typically within 5-10 iterations, depending on the chosen number of clusters K and the dataset (not shown). As expected, the final likelihood is highest when the number of clusters equals the number of objects in the shapes dataset (3), matching the results from Figure 4.7a. The likelihood is much lower for $k = 2$ than for $k = 3$ and drops again slightly if we choose $k = 5$. The likelihood for $k = 12$ is significantly lower. In some cases the correct choice of k did not result in the highest likelihood, but in general this correspondence appeared to hold. If the number of objects is unknown, this trend can be used to determine the correct number of clusters.



(a) Overall Scores (b) Convergence

Figure 4.7. **Left:** Mean AMI score over 1000 test samples for all datasets and various number of clusters K. **Right:** Convergence of the log-likelihood on the *shapes* dataset for different numbers of clusters, showing test set mean (line) and standard deviation (shaded) over the test set.

4.4.3 Qualitative Analysis

Figures 4.8 to 4.10 show a few example RC runs for qualitative evaluation. The initial cluster assignments are random, therefore all observed structure is due to the clustering process. The final clustering corresponds well to the ground truth even for cases with significant overlap. Again, it is notable that RC converges quickly (within 5 iterations).

4.4.4 Loss vs Score

RC utilizes autoencoders trained with the denoising objective for binding. Therefore, it is instructive to examine the relationship between denoising performance and the final RC binding score. For this purpose, we trained 100 DAEs with the same architecture on each dataset with random learning rates and initializations, and then performed RC using each of them. Figure 4.11 shows the relationship between the denoising loss and binding score for each dataset. It can be observed that lower loss correlates positively with higher score for all datasets, indicating that denoising is a suitable surrogate training objective. We added a regression line to indicate that relation for each dataset, even though for *MNIST+Shape* and *Multi MNIST* it doesn't look even remotely linear. Instead, the individual points are approximately arranged on a curve. This suggests that there is a direct but complex interplay between the denoising performance and the score.

4.4.5 Training on Multiple Objects

So far the DAEs were trained on single-object images, then used to bind objects in multi-object images. In general it is desirable to not *require* single-object images for training, and be able to directly use any image without this restriction. This would remove the last bit of supervision and make RC a truly unsupervised method. To test this we performed a separate random search to tune DAE hyperparameters for the case of multi-object training. The only difference is the training data and that for determining the final score we use K-means-like (hard) cluster

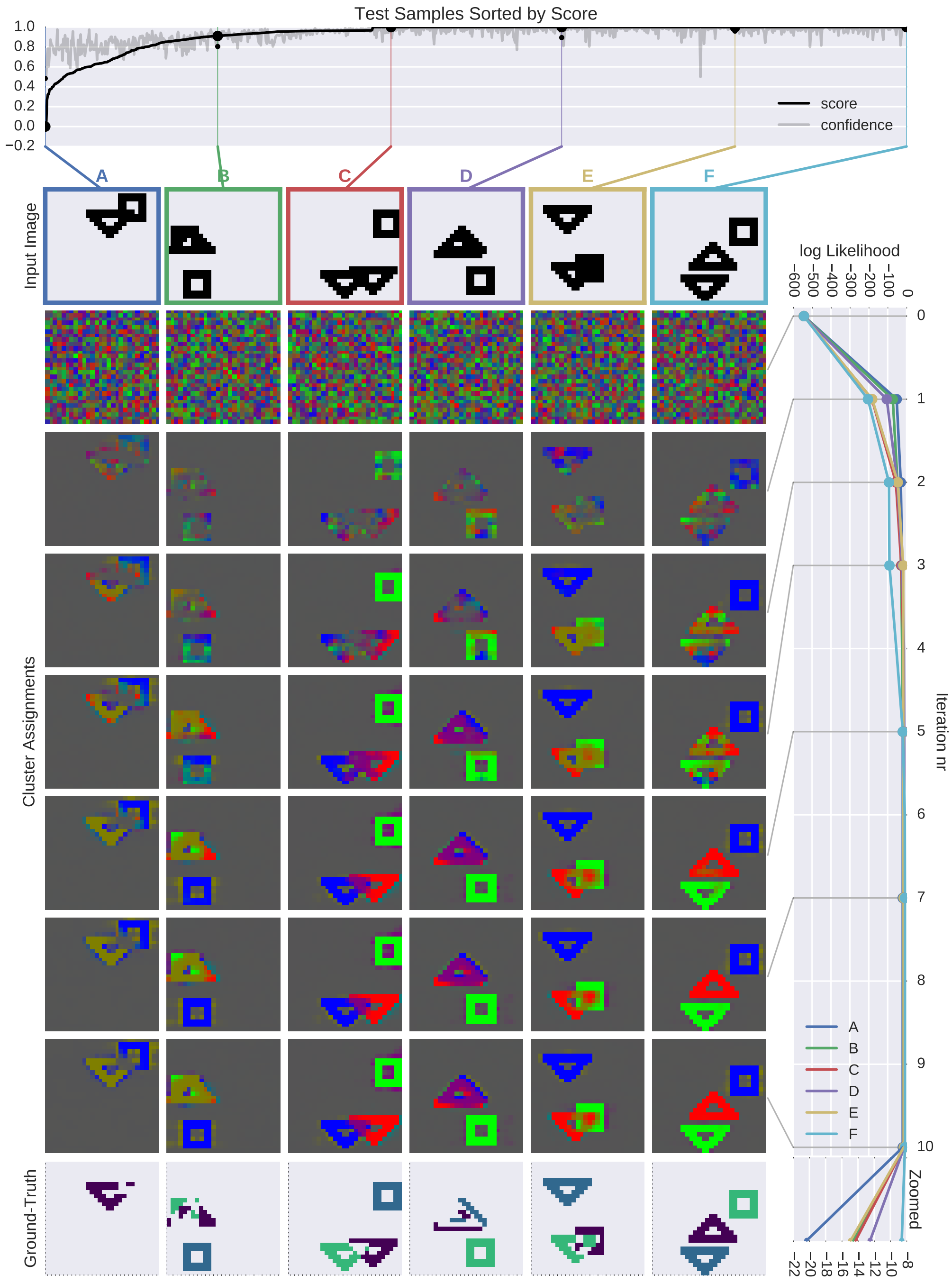


Figure 4.8. The top plot shows the score and confidence for each of the 1000 test images from the shapes dataset, sorted by score. The confidence is the average value of $\max_k m_{i,k}$ for each evaluated pixel (non-background, non-overlap). The central part of the figure shows six examples (columns) along with the cluster assignments (indicated by different colors) over RC iterations. The corresponding ground-truth is shown at the bottom. The right vertical plot shows the log-likelihood over the RC iterations corresponding to the displayed cluster assignments.

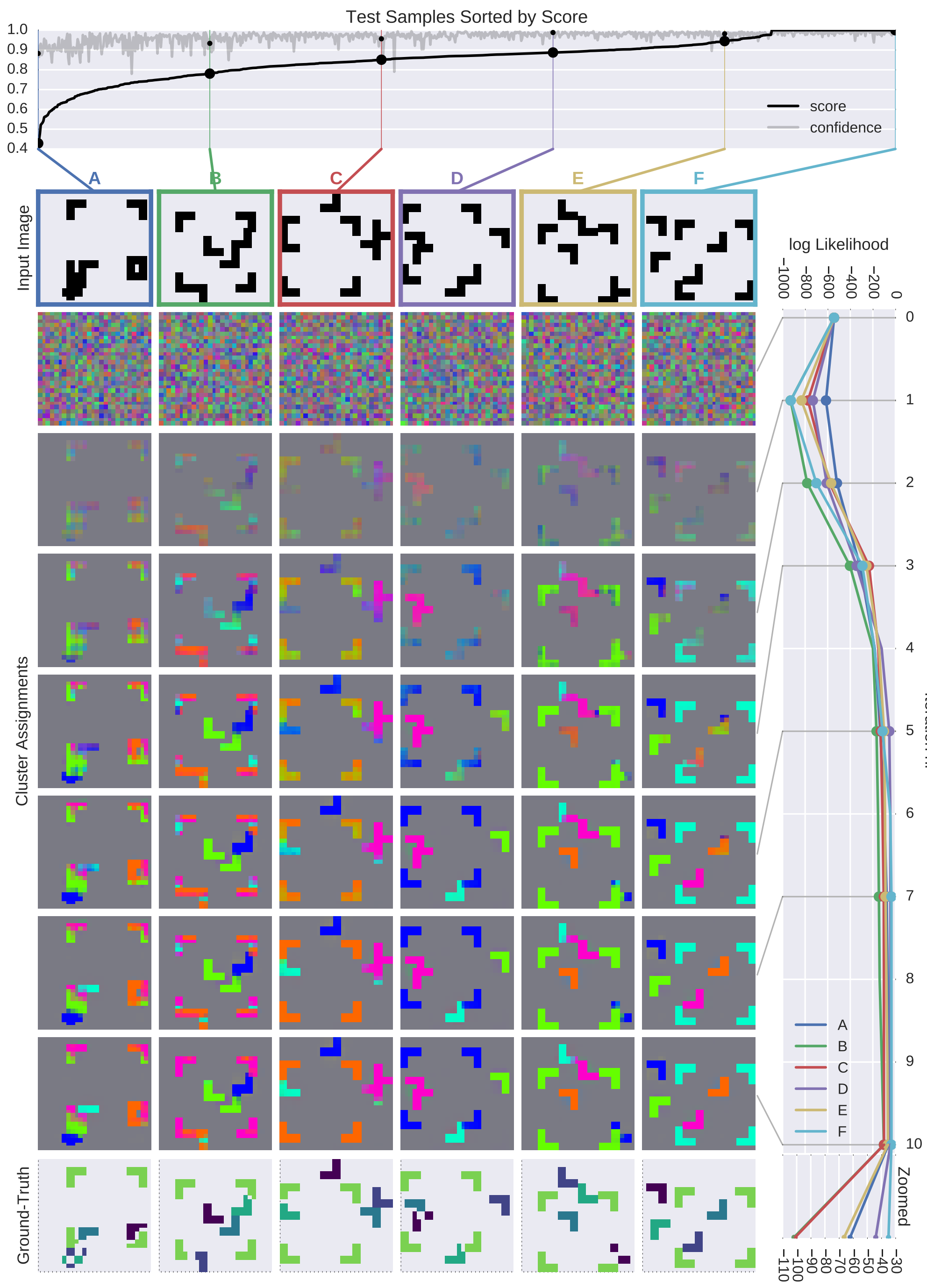


Figure 4.9. The top plot shows the score and confidence for each of the 1000 test images from the *corners dataset*, sorted by score. The central part of the figure shows six examples (columns) along with the cluster assignments (indicated by different colors) over RC iterations. The corresponding ground-truth is shown at the bottom. The right vertical plot shows the log-likelihood over the RC iterations corresponding to the displayed cluster assignments.

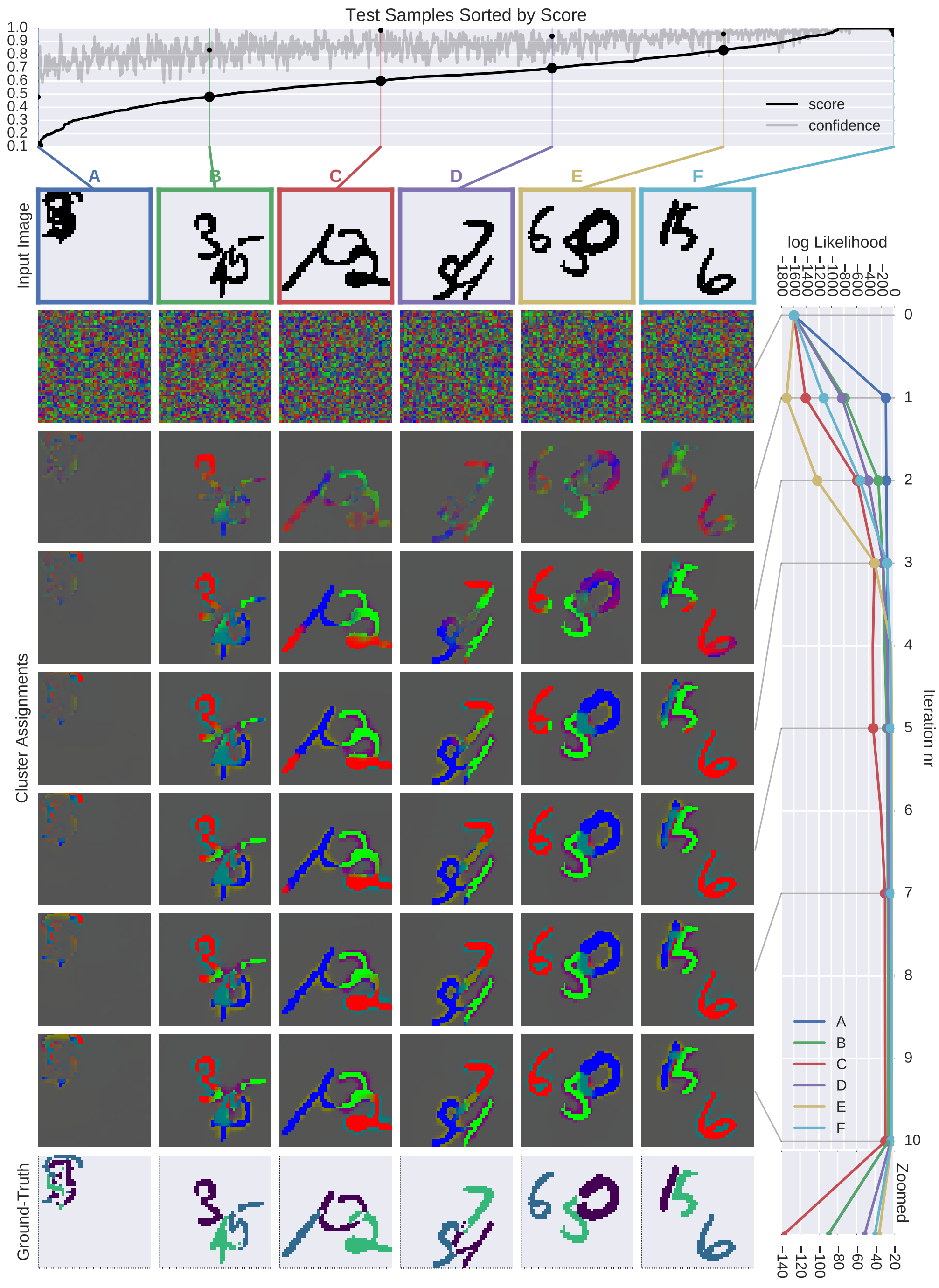


Figure 4.10. The top plot shows the score and confidence for each of the 1000 test images from the *Multi-MNIST dataset*, sorted by score. The central part of the figure shows six examples (columns) along with the cluster assignments (indicated by different colors) over RC iterations. The corresponding ground-truth is shown at the bottom. The right vertical plot shows the log-likelihood over the RC iterations corresponding to the displayed cluster assignments.

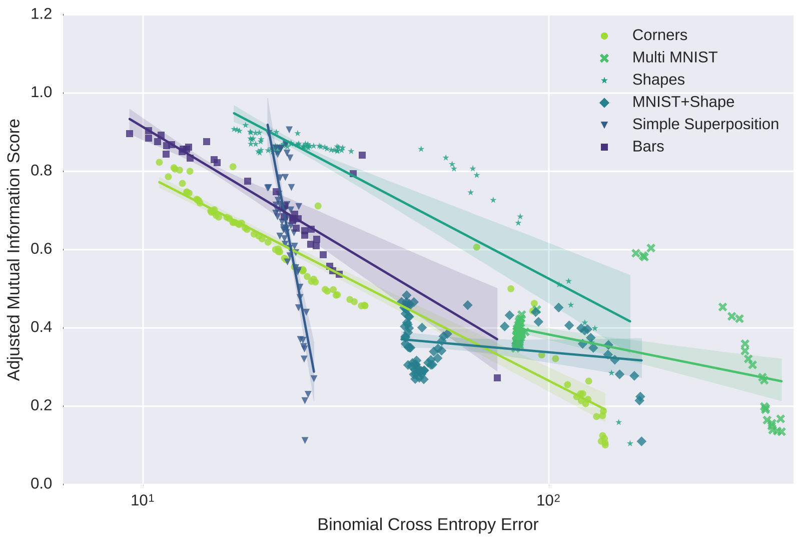
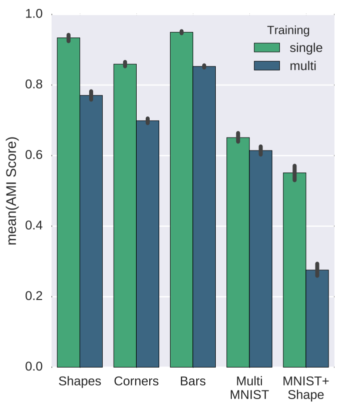
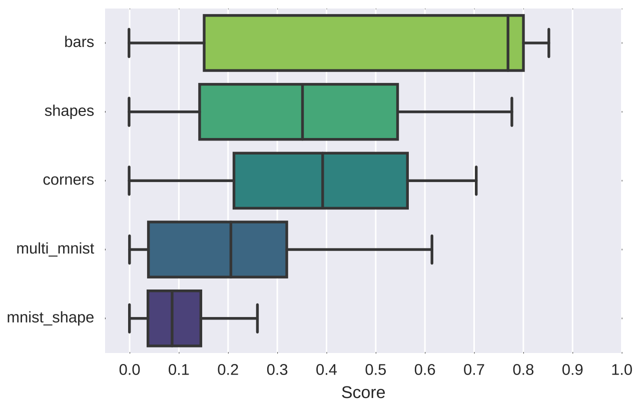


Figure 4.11. Relationship between the DAE loss and the AMI score. All networks have 250 hidden units and were trained with random learning rates and initializations. A few networks that failed to train were removed from the plot for better visualization.



(a)



(b)

Figure 4.12. **(a)** RC scores obtained when training DAEs on multi-object images vs. single object images. **(b)** Summary of the scores achieved during the random search for training with multiple objects.

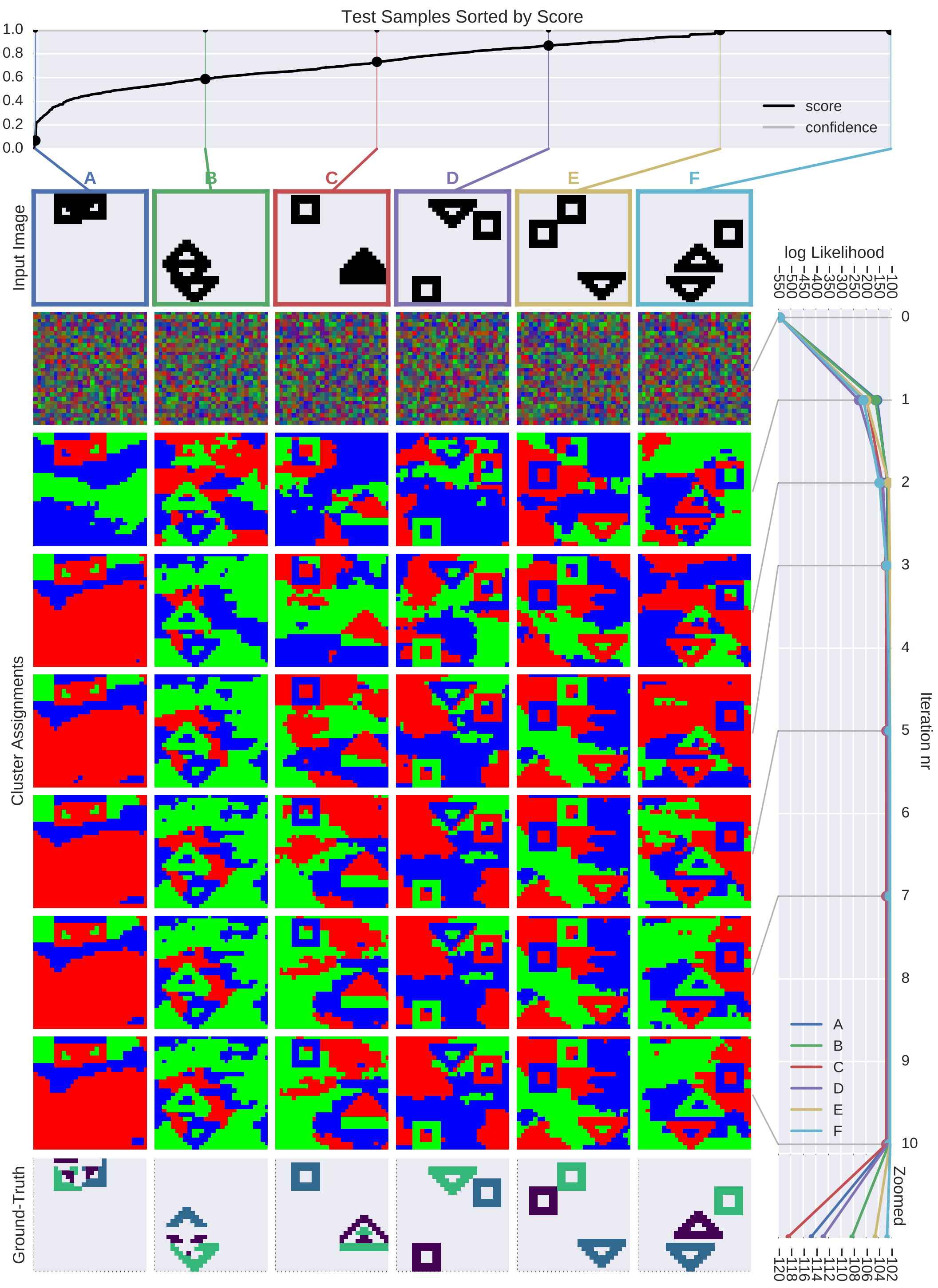


Figure 4.13. Example iterations of RC when using hard assignments and a DAE that has been trained only on images with multiple objects.

Dataset	learning rate	# hidden units	activation	salt&pepper	score
bars	0.012192	100	sigmoid	0.8	0.851777
corners	0.026035	100	ReL	0.7	0.704285
mnist_shape	0.033200	1000	ReL	0.6	0.259646
multi_mnist	0.001786	250	sigmoid	0.9	0.614277
shapes	0.049402	100	sigmoid	0.9	0.776656

Table 4.2. Configuration of the best network trained on *multiple objects* for each dataset as found by the random search.

assignments in RC. Note also that we didn't include the Simple Superposition dataset, since it only consists of 120 images with multiple objects available, and no separate test set. Similar to the single-object case, we then used the best obtained DAEs to perform RC on test examples.

When training the DAEs on images with multiple objects, it is less obvious why running RC should lead to a segregation of the objects. Intuitively it seems that the autoencoder should always try to reconstruct the whole image including all the objects. However, even if each cluster tries to reconstruct every object, there will be small asymmetries due to the difference in inputs they see. Since no object carries any information about the shape and position of another object in our datasets, this will lead to differences in prediction quality of the objects. We found that with soft-assignments to the clusters, the differences were too small and would even out over several iterations, leading to uniform cluster assignments. By changing the E-step to hard (K-Means-like) assignments, we were able to amplify these changes enough to eliminate this stable state, and force the clusters to compete more for the pixels. Together with the fact that in our datasets objects don't carry any information about other objects this leads to a stronger amplification of the initial differences in reconstruction quality. Figure 4.12a shows that DAEs trained on multi-object images can indeed be used for binding via RC with hard assignments, although they lead to lower scores in comparison. In Figure 4.13 this process can be seen on the shapes dataset. Note that the hard RC converges even faster, but generally leads to worse performance.

4.4.6 Generalization to Unfamiliar Images

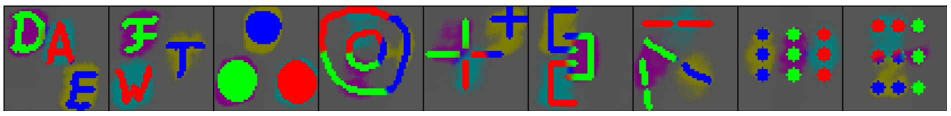


Figure 4.14. Binding novel objects via RC. The DAE used was trained on the *Multi MNIST* dataset.

A central intuition behind our approach to binding is that the low-level structures learned by the model will generalize to new and unseen configurations. Evaluation on unseen test sets demonstrated this to be true, but we can take it one step further. We can test what happens when we confront our method with novel objects that the auto-encoders have not been trained on.

We ran RC on several images with non-digits using a DAE trained on the *Multi-MNIST* dataset. Figure 4.14 shows that RC “correctly” binds letters and circles together. We also show images for which the resulting binding differs from our expectation. It appears that the network has mainly learned to bind based on spatial proximity with a slight bias towards vertical proximity. This can be expected since that it has only seen digits of roughly the same size so far, and because the used autoencoder is very limited. Nevertheless, it is very interesting that a fully-connected network which is permutation invariant learns the preference for spatial proximity entirely from data. It is reasonable to speculate that it in the future it may be possible to recover other *Gestalt Principles* such as continuity and similarity with a similar procedure.

4.5 Relationship to other Methods

Recently, the ideas of neural synchronization for dynamic binding were implemented using complex valued activations in neural networks to jointly encode firing rate and phase [304, 309]. Such binding mechanisms are close to their biological inspiration, clustering only implicitly through synchronization. In contrast, RC is based on a mathematical framework which explicitly incorporates binding.

The core ideas of RC are similar to *Masked Restricted Boltzman Machine* (MRBM; 223). They model an image as being composed of multiple layers (clusters) and also add a corresponding latent variable for each pixel. Similar to the DAEs used for RC each layer is modelled by a separate *Restricted Boltzman Machine* (RBM) all of which share weights. Le Roux et al. [223] further parameterize the model for the shape of masks (cluster assignments) and explicitly model occlusion. The main difference is that for RC we use DAEs together with a simple clustering mechanism instead of RBMs to perform inference. This simplifies training and seems to speed up convergence.[4]

Structurally, RC resembles a model introduced by [408]. They too split the image representation into competing feature layers (objects). Inter-object predictability is modelled by lateral connections that represent compatibility between features. These connections are trained from labelled samples using Hebbian learning. After training, the clustering for an input is obtained by finding a fixed point of the energy function defined over the layers using the Gauss-Seidel method.

In some aspects, RC is also similar to segmentation algorithms. The main difference is that RC learns the segmentation from the data in a largely unsupervised manner. In this sense, it is more similar to *superpixel* methods (see eg. [2] for an overview). However, these methods impose a handcrafted similarity measure over pixels or pixel regions, whereas RC learns a non-linear similarity measure from the data, parameterized by a DAE.

4.6 Conclusion and Future Work

We adopted a notion of object-ness based on mutual predictability of their constituent pixels. Based on that we introduced the Reconstruction Clustering framework to explicitly model data as a composition of objects. This work should be considered a proof-of-concept, since it still

[4]Since the datasets used for evaluation differ this comparison isn’t very informative. That being said, Le Roux et al. [223] report results for 5000 steps of Gibbs sampling, while our model typically converges within 10 iterations. This speedup is also in line with the observations of [309], whose model also builds upon RBMs and typically takes 100 steps to converge.

has many limitations, including the simplistic datasets, and missing integration between the DAE object models and the segregation process. But compared to previous approaches to perceptual grouping, our framework is completely unsupervised and integrates well with neural representation learning methods. While a typical representation learning method (such as a DAE) learns a *static binding* of features, Reconstruction Clustering utilizes it to iteratively perform *dynamic binding* for every input example by introducing interaction between the statically bound features extracted by the autoencoder. In particular, this interaction enables dynamic binding of feature combinations never seen before by the autoencoder. It demonstrates that the task of perceptual grouping can plausibly be learned in a completely unsupervised fashion.

Chapter 5

Neural Expectation Maximization

In the previous chapter we have introduced *Reconstruction Clustering* (RC; 122): a proof-of-concept system which learns to segregate an image into objects in an unsupervised manner. It showed that this challenging task can plausibly be tackled with a clustering approach built around a neural network. But RC has several shortcomings: Firstly, its *Denoising Autoencoder* (DAE; 27, 391) is trained separatly outside of the clustering iterations which means that the object model is ignorant of the perceptual grouping process. Secondly, the mathematical basis of RC is shaky due to the ad-hoc substitution of the M-Step with a pass through the encoder of the DAE. Finally, the networks used are not very powerful and show poor denoising performance even on the simplistic toy datasets that we investigated.

In this chapter, we develop a formal framework that mitigates these issues. It is still based on *Expectation Maximization* (EM; 69) to do inference in a (pixel-wise) spatial mixture, where each component is parametrized by a *Neural Network* (NN). But unlike in Chapter 4, we integrate them into a unified differentiable clustering method which can be trained end-to-end. We call this framework *Neural Expectation Maximization* (N-EM; 123), and show that it is able to simultaneously learns how to group and represent individual entities. We evaluate our method on the (sequential) perceptual grouping task and find that it is accurately able to recover the constituent objects. We demonstrate that the learned representations are useful for predictive coding. This chapter is based on the publication Greff et al. [123], and is joint work with my shared first-author Sjoerd van Steenkiste. Due to the nature of our close collaboration it is impossible to attribute parts of the work to either of us individually.

5.1 Method

Our goal is to train a system in an unsupervised fashion to produce separate representations for the individual conceptual entities (objects) contained in a given input (here: image). We will use the same predictability-based notion of objects as in Chapter 4. Moreover, we are interested in representing each such entity (object) k with some vector $\mathbf{z}_k$ that captures all the structure of the affected pixels, but carries no information about the rest of the image. This modularity is a powerful invariant, since it allows the same representation to be reused in different contexts, which enables generalization to novel combinations of known objects. Having all possible objects represented in the same format makes it easier to work with these representations. Finally, having a separate $\mathbf{z}_k$ for each object (as opposed to for the entire image) allows $\mathbf{z}_k$ to be distributed and disentangled without running into the binding problem.

We treat each image as a composition of K objects, where each pixel is determined by exactly one object. Which objects are present, as well as the corresponding assignment of pixels to objects, varies from input to input. Assuming that we have access to the family of distributions $P_{X|Z}(\mathbf{x} \mid \mathbf{z}_k)$ that corresponds to an object level representation as described above, we can model each image as a mixture model. Then EM can be used to simultaneously compute a *Maximum Likelihood Estimate* (MLE) for the individual $\mathbf{z}_k$-s and the grouping we are interested in.

The central problem we consider in this work is therefore how to learn such a $P_{X|Z}(\mathbf{x} \mid \mathbf{z}_k)$ in a completely unsupervised fashion. We accomplish this by parametrizing this family of distributions by a differentiable function $f_\phi(\mathbf{z})$ (a neural network with weights ϕ). We show that in that case, the corresponding EM procedure also becomes fully differentiable, which allows us to backpropagate an appropriate outer loss into the weights of the neural network. In the rest of this section we formalize and derive this method which we call N-EM.

5.1.1 Parametrized Spatial Mixture Model

We model each image $\mathbf{x} \in \mathbb{R}^D$ as a spatial mixture of K components parametrized by vectors $\mathbf{z}_1, \ldots, \mathbf{z}_K \in \mathbb{R}^H$. A differentiable non-linear function f_ϕ (a neural network) is used to transform these representations $\mathbf{z}_k$ into parameters $\psi_{i,k} = f_\phi(\mathbf{z}_k)_i$ for separate pixel-wise distributions. These distributions are typically Bernoulli or Gaussian in which case $\psi_{i,k}$ would be a single probability or a mean and variance respectively. This parametrization assumes that given the representation, the pixels are independent but *not* identically distributed (unlike in standard mixture models). A set of categorical latent variables $\boldsymbol{C} \in \{0, \ldots K\}^D$ encodes the unknown true pixel assignments, such that $c_i = k$ iff pixel i was generated by component k. A graphical representation of this model can be seen in Figure 5.1, where $\boldsymbol{\pi} = (\pi_1, \ldots \pi_K)$ are the mixing coefficients (or prior for $\boldsymbol{C}$). The full likelihood for $\mathbf{x}$ given $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_K)$ is given by:

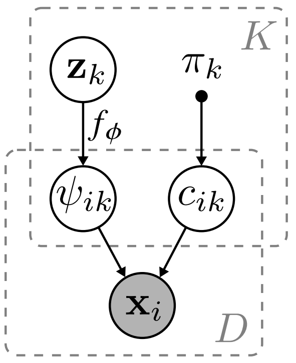


Figure 5.1. The probabilistic graphical model for N-EM.

$$P_{X|Z}(\mathbf{x} \mid \mathbf{z}) = \prod_{i=1}^{D} \sum_{\mathbf{c}_i} P_{X,C|\Psi}(x_i, \mathbf{c}_i \mid \boldsymbol{\psi}_i) \tag{5.1}$$

$$= \prod_{i=1}^{D} \sum_{k=1}^{K} \underbrace{P_C(c_i = k)}_{\pi_k} P_{X|\Psi,C}(x_i \mid \psi_{i,k}, C_i = k). \tag{5.2}$$

5.1.2 Expectation Maximization

Directly optimizing $\log P_{X|\Psi}(\mathbf{x} \mid \boldsymbol{\psi})$ wrt. $\mathbf{z}$ is difficult due to marginalization over $\mathbf{c}$, while for many distributions optimizing $\log P_{X,C|\Psi}(\mathbf{x}, \mathbf{c} \mid \boldsymbol{\psi})$ is much easier. EM takes advantage of this and instead optimizes a lower bound given by the expected log likelihood:

$$\mathcal{Q}(\mathbf{z}, \mathbf{z}^{\text{old}}) = \mathop{\mathbb{E}}_{C|X,\Psi^{\text{old}}} \left[\log P_{X,C|\Psi}(\mathbf{x}, \mathbf{c} \mid \boldsymbol{\psi})\right]. \tag{5.3}$$

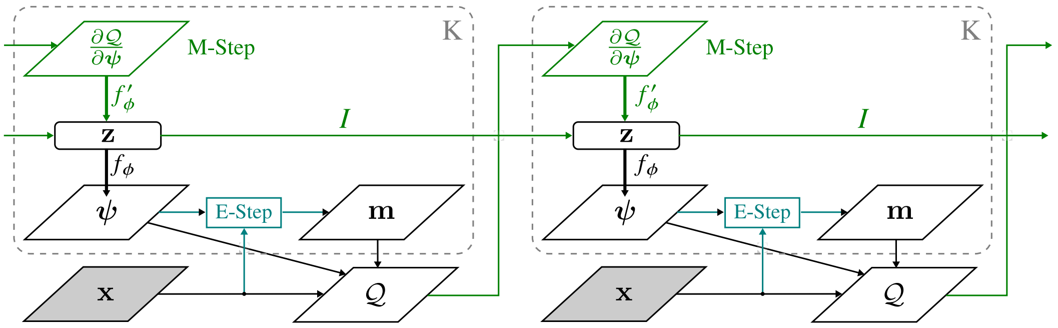


Figure 5.2. Illustration of the computations for two steps of N-EM.

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 assignments given $\mathbf{z}^{old}$ from the previous iteration, yielding a new soft-assignment of the pixels to the components (clusters):

$$m_{i,k} := P_{C|X,\Psi^{\text{old}}}(c_i = k \mid x_i, \psi_i^{\text{old}}). \tag{5.4}$$

In the M-step we then aim to find the configuration of $\mathbf{z}$ that would maximize the expected log-likelihood using the posteriors computed in the E-step. In our case, there exists no analytical solution to $\text{argmax}_{\mathbf{z}}\, \mathcal{Q}(\mathbf{z}, \mathbf{z}^{\text{old}})$ due to the non-linearity of f_ϕ. However, since f_ϕ is differentiable, we can improve $\mathcal{Q}(\mathbf{z}, \mathbf{z}^{\text{old}})$ by taking a gradient ascent step [1]:

$$\mathbf{z}^{\text{new}} = \mathbf{z}^{\text{old}} + \eta \frac{\partial \mathcal{Q}}{\partial \mathbf{z}} \qquad \text{where} \qquad \frac{\partial \mathcal{Q}}{\partial \mathbf{z}_k} \propto \sum_{i=1}^{D} m_{i,k}(\psi_{i,k} - x_i)\frac{\partial \psi_{i,k}}{\partial \mathbf{z}_k}. \tag{5.5}$$

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 [417].

5.1.3 Unrolling

In our model the information about statistical regularities that is required for clustering the pixels into objects is encoded in the neural network f_ϕ with weights ϕ. So far we have considered f_ϕ to be fixed and have shown how we can compute a MLE for $\mathbf{z}$ alongside the appropriate clustering. We now observe that by unrolling the iterations of the presented generalized EM, we obtain an end-to-end differentiable clustering procedure based on the statistical model implemented by f_ϕ. This enables us to train the weights ϕ by means of *Backpropagation Through Time* (BPTT; eg. 410). We refer to this trainable procedure as N-EM, an overview of which can be seen in Figure 5.2.

Upon inspection of the structure of N-EM we find that it resembles K copies of a recurrent neural network with hidden states $\mathbf{z}_k$ that, at each timestep, receive $\mathbf{m}_{:,k} \odot (\boldsymbol{\psi}_{:,k} - \mathbf{x})$ as their input. Each copy generates a new $\boldsymbol{\psi}_{:,k}$, which is then used by the E-step to re-estimate the soft-assignments $\mathbf{m}$. In order to accurately mimic the M-Step (5.5) with an *Recurrent Neural Network* (RNN; 250, 368, 316, 410), we must impose several restrictions on its weights and

[1]Here we assume that $P_{X|C,\Psi}(x_i \mid C_i = k, \psi_{i,k})$ is given by $\mathcal{N}(x_i; \mu = \psi_{i,k}, \sigma^2)$, yet a similar update arises for many typical parametrizations of the pixel distributions.

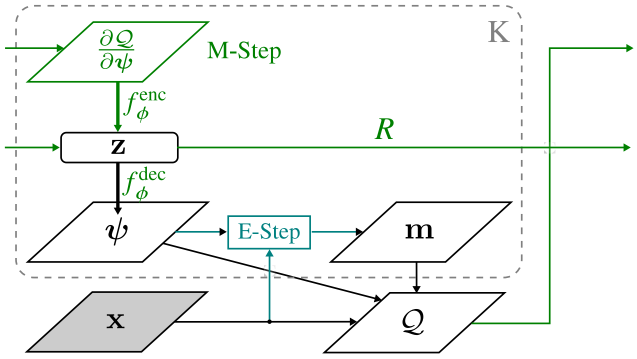


Figure 5.3. Illustration of the computations for a single step of RNN-EM.

structure. Instead we introduce a new algorithm, named *Recurrent Neural Network Expectation Maximization* (RNN-EM; 123), when substituting that part of the computational graph of N-EM with an actual RNN (without imposing any restrictions). Although RNN-EM can no longer guarantee convergence of the data log likelihood, its recurrent weights increase the flexibility of the clustering procedure. Moreover, by using a fully parametrized recurrent weight matrix RNN-EM naturally extends to sequential data. Figure 5.3 presents the computational graph of a single RNN-EM (time) step.

5.1.4 Training Objective

N-EM is a differentiable clustering procedure, whose outcome relies on the statistical model f_ϕ. We are interested in a particular unsupervised clustering that corresponds to grouping entities based on the statistical regularities in the data. In order to train our system we therefore require a loss function that teaches f_ϕ to map from representations $\mathbf{z}$ to parameters $\boldsymbol{\psi}$ that correspond to pixelwise distributions for such objects.

We accomplish this with a two-term loss function that guides each of the K networks to model the structure of a single object independently of any other information in the image:

$$L(\mathbf{x}) = -\sum_{i=1}^{D}\sum_{k=1}^{K} \underbrace{m_{i,k} \log P_{X,C|\Psi}(x_i, C_i = k \mid \psi_{i,k})}_{\text{Term 1}} + \underbrace{(1-m_{i,k}) D_{\mathrm{KL}}[P_X(x_i) \parallel P_{X|\Psi,C}(x_i \mid \psi_{i,k}, C_i = k)]]}_{\text{Term 2}}. \tag{5.6}$$

The first term corresponds to the same expected data log-likelihood $\mathcal{Q}$ as is optimized by N-EM. It is analogous to a standard reconstruction loss used for training autoencoders, weighted by the cluster assignment. Similar to autoencoders, this objective is prone to trivial solutions in case of overcapacity, which prevent the network from modelling the statistical regularities that we are interested in.

Standard techniques can be used to overcome this problem, such as making $\mathbf{z}$ a bottleneck or using a noisy version of $\mathbf{x}$ to compute the inputs to the network. Furthermore, when RNN-EM is used on sequential data we can use next-step prediction loss.

Weighing the loss pixelwise is crucial since it allows for specializing the predictions of each $\mathbf{z}$ to an individual object. However, it also introduces a problem: the loss for out-of-cluster pixels ($m_{i,k} = 0$) vanishes. This leaves the network free to predict anything and does not yield specialized representations. Therefore, we add a second term which penalizes the KL divergence between out-of-cluster predictions and the pixelwise prior of the data. Intuitively this tells each representation $\mathbf{z}_k$ to contain no information regarding non-assigned pixels x_i:

$$P_{X|\Psi,C}(x_i \mid \psi_{i,k}, C_i = k) = P_X(x_i).$$

A disadvantage of the interaction between m and ψ in (5.6) is that it may yield conflicting gradients. For any $\mathbf{z}_k$ the loss for a given pixel i can be reduced by better predicting x_i, or by decreasing $m_{i,k}$ (i.e. taking less responsibility) which is (due to the E-step) realized by being worse at predicting x_i. A practical solution to this problem is obtained by stopping the $\mathbf{m}$ gradients, i.e. by setting $\frac{\partial L}{\partial m} = 0$ during backpropagation.

5.2 Related work

The Binding problem was first considered in the context of Neuroscience [257, 395] and has sparked some early work in oscillatory neural networks that use synchronization as a grouping mechanism [397, 403, 304]. Later, complex valued activations have been used to replace the explicit simulation of oscillation [303, 309]. By virtue of being general computers, any RNN can in principle learn a suitable mechanism. In practice however it seems hard to learn, and adding a suitable mechanism like competition [411], fast weights [339], or our N-EM seems necessary.

Unsupervised Segmentation has been studied in several different contexts [339], from random vectors [164] over texture segmentation [128] to images [188, 167]. Early work in unsupervised video segmentation [183] used generalized Expectation Maximization (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 [292, 390]. A related line of work under the term of multi-causal modelling [333] has formalized perceptual grouping as inference in a generative compositional model of images. *Masked Restricted Boltzman Machines* (MRBMs; 223) for example extend *Restricted Boltzman Machines* (RBMs) with a latent mask inferred through Block-Gibbs sampling.

Gradient backpropagation through inference updates has previously been addressed in the context of sparse coding with (Fast) Iterative Shrinkage/Tresholding Algorithms ((F)ISTA; [64, 323, 24]). 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 [126]. 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 [363]. An alternative to further empower the network by untying its weights across iterations [139] was not considered for flexibility reasons.

5.3 Experiments

We evaluate our approach on the 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 “correct” grouping and learns representations corresponding to these objects. In particular we are interested in studying the role of next-step prediction as a unsupervised objective for perceptual grouping, the effect of hyperparameter K, and the usefulness of the learned representations.

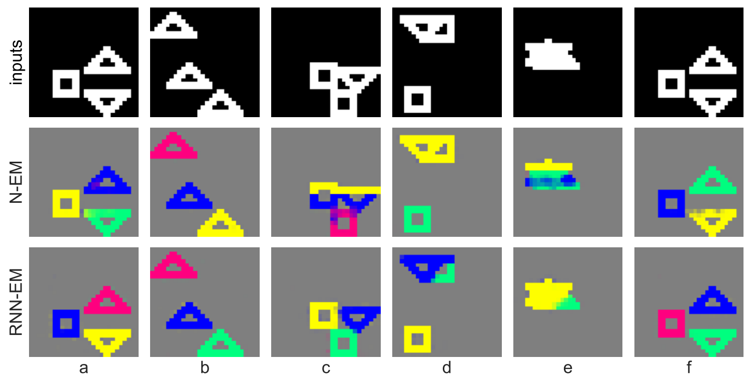


Figure 5.4. Groupings by RNN-EM (bottom row), N-EM (middle row) for six input images (top row). Both methods recover the individual shapes accurately when they are separated (a, b, f), even when confronted with the same shape (b). RNN-EM is able to handle most occlusion (c, d) but sometimes fails (d). The exact assignments are permutation invariant and depend on γ initialization compare (a) and (f).

In all experiments we train the networks using *Adaptive Moment Estimation* (Adam; 198) with default parameters, a batch size of 64 and 50 000 train + 10 000 validation + 10 000 test inputs. Consistent with earlier work [122, 120], 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; 392) score, which provides a measure of clustering similarity between 0 (random) and 1 (perfect match). We use early stopping when the validation loss[2] has not improved for 10 epochs. All reported results are averages computed over five different runs.[3]

5.3.1 Static Shapes

To validate that our approach yields the intended behavior we consider a the simple perceptual grouping task of grouping three randomly chosen regular shapes (△▽□) located in random positions [309] of a 28×28 binary image. The goal is to cluster the pixels in each image into the individual shapes and in doing so obtain a separate representation for each shape. 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 by means of a single layer fully connected neural network with a sigmoid output for each pixel that corresponds to a Bernoulli distribution with mean $\psi_{i,k}$. The representation $\mathbf{z}_k$ is a real-valued 250-dimensional vector squashed to the $(0, 1)$ range by a sigmoid function before being fed into the network. Similarly for RNN-EM we use a recurrent neural network with 250 sigmoidal hidden units and an equivalent output layer. Both networks are trained with $K = 3$ for 15 EM steps, and the outer-loss is only injected at the final EM-step.

As shown in Figure 5.4, we observe that both approaches are able to recover the individual shapes as long as they are separated, even when confronted with identical shapes. N-EM performs

[2]Note that we do not stop on the AMI score as this is not part of our objective function and only measured to evaluate the performance *after* training.

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

Type	Size	Act. Func.	Comment
Input: $\mathbf{x}$	$28 \times 28 \times 1$		
Conv 4×4	$14 \times 14 \times 32$	ELU	stride 2 & layernorm
Conv 4×4	$7 \times 7 \times 64$	ELU	stride 2 & layernorm
MLP	512	ELU	layernorm
recurrent:	100	sigmoid	layernorm on the output
MLP	512	ReLU	layernorm
MLP	$7 \times 7 \times 64$	ReLU	layernorm
rescale	$14 \times 14 \times 64$		nearest-neighbour
Conv 4×4	$14 \times 14 \times 32$	ReLU	layernorm
rescale	$28 \times 28 \times 32$		nearest-neighbour
Conv 4×4	$28 \times 28 \times 1$	sigmoid	

Table 5.1. Architecture for the network used for flying shapes.

worse if the image contains occlusion, and in 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 [126]. Similarly we conclude that the tunable degrees of freedom in RNN-EM help speed-up the optimization process resulting in a more powerful approach that requires fewer iterations. The benefit is reflected in the large score difference between the two: 0.826 ± 0.005 AMI compared to 0.475 ± 0.043 AMI for N-EM.

5.3.2 Flying Shapes

We consider a sequential extension of the *static shapes* dataset. Each input consists of a sequence of binary 28×28 images containing a fixed number of shapes (△▽□) that start in random positions and "fly" across randomly sampled trajectories and bounce off the walls of the image for 20 steps. An example sequence with 5 shapes can be seen in the bottom row of Figure 5.5. We use a convolutional encoder and decoder with *Rectified Linear Unit* (ReLU) and *Exponential Linear Unit* (ELU; 57) activation functions inspired by the discriminator and generator networks of infoGAN [53], with a recurrent neural network of 100 sigmoidal units as seen in Table 5.1

Instead of using transposed convolutions (to implement the "de-convolution") we first reshape the image using the default nearest-neighbour interpolation followed by a normal convolution in order to avoid frequency artifacts [282]. Note that we do not add layer norm on the recurrent connection.

At each timestep t we feed $m_{:,k}(\psi_{:,k}^{(t-1)} - \tilde{\mathbf{x}}^{(t)})$ as input to the network, where $\tilde{\mathbf{x}}$ is the input with added bitflip noise ($p = 0.2$). RNN-EM is trained with a next-step prediction objective implemented by replacing $\mathbf{x}$ with $\mathbf{x}^{(t+1)}$ in (5.6), which we evaluate at each time-step. A single RNN-EM step is used for each timestep. The prior for each pixel in the data is set to a Bernoulli distribution with $p = 0$. We prevent conflicting gradient updates by not back-propagating any gradients through $\mathbf{m}$.

For three shapes we observe that the produced groupings are close to perfect (AMI: 0.970 ± 0.005). Even in the very cluttered case of five shapes the network is able to separate the individual objects in almost all cases (AMI: 0.878 ± 0.003).

These results demonstrate the adequacy of the next step prediction task for perceptual grouping. However, we also find that the converse holds: the corresponding representations

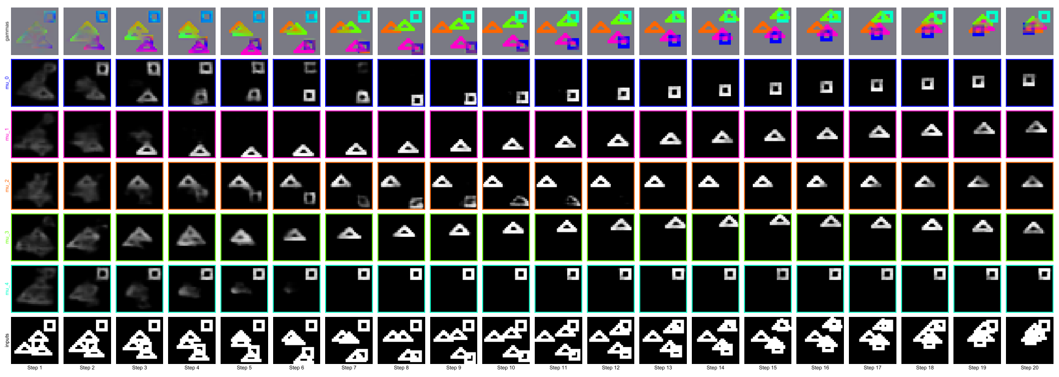


Figure 5.5. A sequence of 5 shapes flying across random trajectories in the image (bottom row 1). The next-step prediction of each copy of the network (rows 2 to 5) and the soft-assignment of the pixels to each of the copies (top row). Observe that the network learns to separate the individual shapes as a means to efficiently solve next-step prediction. Even when many of the shapes are overlapping, as can be seen in time-steps 18-20, the network is still able to disentangle the individual shapes from the clutter.

are useful for the prediction task. In Figure 5.6 we compare the next-step prediction error 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. Note that RNN-EM produces significantly lower errors, especially when the number of objects increases.[4]

Finally, in Table 5.3 we also provide insight into the impact of choosing the hyper-parameter K, which is unknown for many real-world scenarios.

Surprisingly we observe that training with too large K is in fact favourable, 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 choices of K, and specifically that choosing it to be too high is not detrimental.

5.3.3 Flying MNIST

In order to incorporate greater variability among the objects we consider a sequential extension of MNIST. Here each sequence consists of gray-scale 24×24 images containing two down-sampled MNIST digits that start in random positions and "fly" across randomly sampled trajectories within the image for T timesteps. An example sequence can be seen in the bottom row of Figure 5.8. We deploy a slightly deeper version of the architecture shown in Table 5.2.

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 copy of the network. The training procedure is identical to flying shapes except that we replace bitflip noise with masked uniform noise: we first sample a binary mask from a multi-variate Bernoulli distribution with $p = 0.2$ and then use this mask to interpolate between the original image and samples from a Uniform distribution

[4]To evaluate RNN-EM on next-step prediction we computed its loss using $P_{X|\Psi}(x_i \mid \psi_i) = P_{X|\Psi}(x_i \mid \max_k \psi_{i,k})$ as opposed to $P_{X|\Psi}(x_i \mid \psi_i) = \sum_k m_{i,k} P_{X|\Psi}(x_i \mid \psi_{i,k})$ to avoid including information from the next timestep.

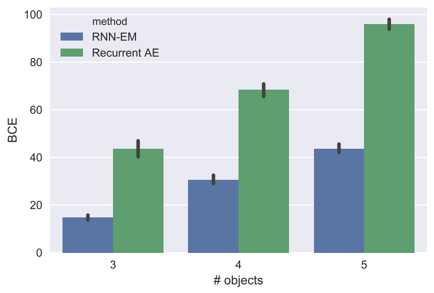


Figure 5.6. Binomial Cross Entropy Error obtained by RNN-EM and a recurrent autoencoder (RNN-EM with $K = 1$) on the denoising and next-step prediction task. RNN-EM produces significantly lower BCE across different numbers of objects.

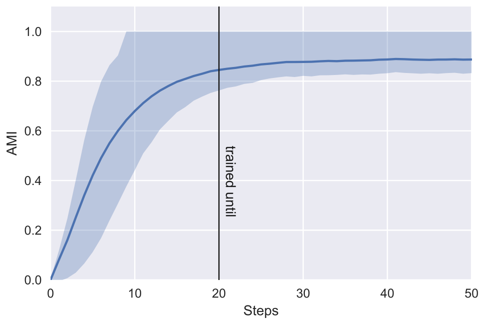


Figure 5.7. Average AMI score (blue line) measured for RNN-EM (trained for 20 steps) across the test-set and corresponding quartiles (shaded areas), computed for each of 50 time-steps. The learned grouping dynamics generalize to longer sequences and even further improve the AMI score.

Type	Size	Act. Func.	Comment
Input: $\mathbf{x}$	$24 \times 24 \times 1$		
Conv 4×4	$12 \times 12 \times 32$	ELU	stride 2 & layernorm
Conv 4×4	$6 \times 6 \times 64$	ELU	stride 2 & layernorm
Conv 4×4	$3 \times 3 \times 128$	ELU	stride 2 & layernorm
MLP	512	ELU	layernorm
recurrent:	250	sigmoid	layernorm on the output
MLP	512	ReLU	layernorm
MLP	$3 \times 3 \times 128$	ReLU	layernorm
rescale	$6 \times 6 \times 128$		nearest-neighbour
Conv 4×4	$6 \times 6 \times 64$	ReLU	layernorm
rescale	$12 \times 12 \times 64$		nearest-neighbour
Conv 4×4	$12 \times 12 \times 32$	ReLU	layernorm
rescale	$24 \times 24 \times 32$		nearest-neighbour
Conv 4×4	$24 \times 24 \times 1$	linear	

Table 5.2. Architecture for the network used for flying shapes.

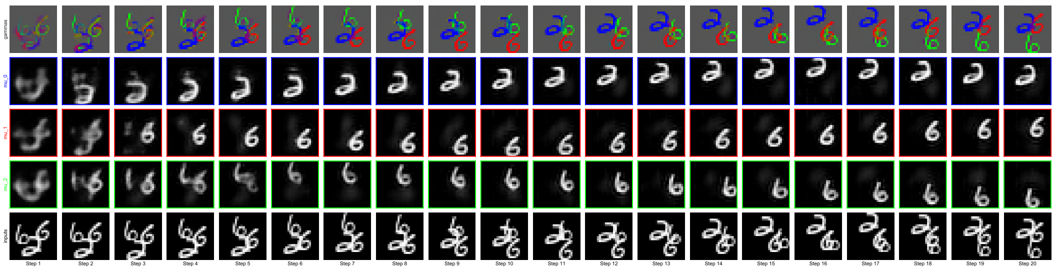


Figure 5.8. A sequence of 3 MNIST digits flying across random trajectories in the image (bottom row). The next-step prediction of each copy of the network (rows 2 to 4) and the soft-assignment of the pixels to each of the copies (top row). Although the network was trained (stage-wise) on sequences with two digits, it is accurately able to separate three digits.

between the minimum and maximum values of the 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.

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, measured across 5 runs. In early experiments we observed that, given the large variability among the 50 000 unique digits, we can boost the model performance by training in stages using 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 hyper-parameter configuration as before and obtain an AMI score of 0.917 ± 0.005 on the test set, measured across 5 runs.

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 increase K to three. Even without further training we are able to maintain 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 5.8. In the second experiment we are interested in whether the grouping mechanism that has been learned can be transferred to static images. We find that using 50 RNN-EM steps we are able to transfer a large part of the learned grouping dynamics and obtain an AMI score of 0.619 ± 0.023 (stage-wise: 0.772 ± 0.008). As a final experiment we evaluate the directly trained network on the same dataset for a larger number of timesteps. Figure 5.7 displays the average AMI score across the test set as well as the range of the upper and lower quartile for each timestep.

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

5.4 Discussion

The experimental results indicate that the proposed Neural Expectation Maximization framework can indeed learn how to group pixels according to constituent objects. In doing so the network learns a useful and localized representation for individual entities, which encodes only the

Train			Test			Test Generalization		
# obj.	K	AMI	# obj.	K	AMI	# obj.	K	AMI
3	3	0.969 ± 0.006	3	3	0.970 ± 0.005	3	5	0.972 ± 0.007
3	5	0.997 ± 0.001	3	5	0.997 ± 0.002	3	3	0.914 ± 0.015
5	3	0.614 ± 0.003	5	3	0.614 ± 0.003	3	3	0.886 ± 0.010
5	5	0.878 ± 0.003	5	5	0.878 ± 0.003	5	3	0.981 ± 0.003

Table 5.3. AMI scores obtained by RNN-EM on *flying shapes* when varying the number of objects (# obj.) and number of components K, during training and at test time.

information relevant to it. Each entity is represented separately in the same space, which avoids the binding problem and makes the representations usable as efficient symbols for arbitrary entities in the dataset. We believe that this is useful for reasoning in particular, and a potentially wide range of other tasks that depend on interaction between multiple entities. Empirically we find that the learned representations are already beneficial in predictive coding with multiple objects, a task in which overlapping objects are problematic for standard approaches but can be handled efficiently when learning a separate representation for each object.

As is typical in clustering methods, in N-EM there is no preferred assignment of object 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. RNN-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 ill-defined and task dependent. We envision future work to alleviate this by extending unsupervised segmentation to hierarchical groupings that are dynamically conditioned on the task at hand.

5.5 Conclusion

We have argued for the importance of separately representing conceptual entities contained in the input, and suggested clustering based on statistical regularities as an appropriate unsupervised approach for separating them. We formalized this notion and derived a novel framework that combines neural networks and generalized EM into a trainable clustering algorithm. Unlike the RC algorithm presented in Chapter 4 this framework fully integrates the neural object model with the clustering process into a system that can be trained end-to-end. The datasets we used are still very simplistic, but we have shown how this method can be trained in a fully unsupervised fashion to segment its inputs into entities, and to represent them individually. Using synthetic images and video, we have empirically verified that our method can recover the objects underlying the data, and represent them in a useful way. N-EM framework still assumes that the latent assignments are independent of the object representations, which is clearly unrealistic and may be an important limitation.

Chapter 6

Iterative Amortized Grouping

In the previous chapters we have introduced an unsupervised notion of objects based on mutual predictability of their pixels. We used it to build a proof-of-concept system for unsupervised perceptual grouping called *Reconstruction Clustering* (RC; 122) in Chapter 4. And in Chapter 5 we showed how to integrate all the parts into a unified differentiable clustering system that can be trained end-to-end. However, thus far we were only able to successfully apply these methods to simplistic datasets that consist of white objects on a black background. One reason for this limitation is the assumption that the segmentation of the pixels is independent of the object representations.

In this chapter we pursue a slightly different approach, and instead of explicitly modelling objects and assignments as latent random variables, we leave them implicit and up to a neural network to learn. By directly integrating more powerful denoising networks with a differentiable grouping mechanism, we obtain a system that can be trained end-to-end and learns to efficiently perform perceptual grouping as a byproduct. We achieve very fast convergence by allowing the system to amortize the joint iterative inference of the groupings and their representations. We call this framework *iTerative Amortized Grouping* (TAG; 120), and while it works completely unsupervised, it can also be combined with supervised learning for classification or segmentation. We use this to evaluate the usefulness of perceptual grouping on multi-digit classification of very cluttered images that require texture segmentation. Remarkably our method achieves improved classification performance over convolutional networks despite being fully connected, by making use of the grouping mechanism. Furthermore, we observe that our system greatly improves upon the semi-supervised result of a baseline Ladder network on our dataset. These results provide important evidence that grouping is useful and a powerful tool that can help to improve sample efficiency. This chapter is based on the publication [120].

6.1 Method

Our goal is to enable neural networks to split inputs and internal representations into coherent groups that can be processed separately. Like before, each pixel of the input is assigned to one of K different groups which are processed and represented separately by a neural network. The task of perceptual grouping then again amounts to simultaneous inference over two sets of variables: the latent group assignments and the individual group representations. But unlike in RC we want our model to learn to infer not only the representation of each group but also the the group assignments. As before we tackle the resulting mixture model using a iterative inference, but rather than deriving and then running an inference algorithm, we now train a parametric mapping to arrive at the end result of inference as efficiently as possible [126].

This approach of leaving latent variables implicit and directly learning the inference is known as *amortized* inference [365]. This situation is analogous to normal supervised deep learning, which can also be viewed as amortized inference [30]. Rather than specifying all the hidden variables that are related to the inputs and labels and then deriving and running an inference algorithm, a supervised deep model is trained to arrive at an approximation $q(\text{class} \mid \text{input})$ of the true posterior $P(\text{class} \mid \text{input})$ without the user specifying or typically even knowing the underlying generative model. It is also used for representation learning in variational autoencoders where the encoder learns to amortize the posterior inference required by the generative model represented by the decoder. Here we instead build on the framework of denoising autoencoders [94, 224, 391] which are trained to reconstruct original inputs $\mathbf{x}$ from corrupted versions $\tilde{\mathbf{x}}$. By designing the network architecture in a way that mimics iterative inference in a spatial mixture model, we bias the model towards performing perceptual grouping as a by-product of approximating the (unknown) data-generating distribution $P(X)$. This encourages the network to implement useful amortized posterior inference without ever having to specify or even know the underlying generative model.

A high-level illustration of the TAG framework is presented in Figure 6.1: We train a network with a learnable grouping mechanism to iteratively denoise corrupted inputs $\tilde{\mathbf{x}}$. The output at each iteration is an approximation $q^{(t)}(\mathbf{x})$ of the true probability $P(\mathbf{x} \mid \tilde{\mathbf{x}})$, which is refined over iterations indexed by t. If the model can improve the estimates in each step, then it will converge to an approximate solution.

6.1.1 Group Structure

Internally, we divide the network into K separate but identical subnetworks each of which independently processes one of the groups. This can be thought of as running K separate copies of the same network, where each network only sees a subset of the inputs. At the image level, we again introduce a mask $\mathbf{m}$ to encode which of the K groups each input element x_i is assigned to. It mimics the structure of a spatial mixture model, where the mask element $m_{i,k}$ corresponds to the (approximate) probability $q(C_i = k)$ of pixel x_i being assigned to group k. Each network then outputs $\boldsymbol{\mu}_k = q(\mathbf{x}|C = k)$ (the expected value of the input for that group), and a $\mathbf{m}_k = q(C = k)$ (the group assignment probabilities). Each $\boldsymbol{\mu}_k$ and $\mathbf{m}_k$ has the same dimensionality as the input, and they are both iteratively updated. The group assignment probabilities $\mathbf{m}_k$ are forced to be non-negative and sum up to one over k (enforced by using a softmax over groups on the outputs):

$$m_{k,i} \geq 0, \qquad \sum_{k=1}^{K} m_{k,i} = 1. \tag{6.1}$$

In the binary case we assume $q_(x_i = \mathbf{1} \mid C = k) = \text{sigmoid}(\mu_{k,i})$, and in the continuous case we take $\mu_{k,i}$ to represent the mean of a Gaussian distribution with variance σ^2. This way inference is split into K groups, and we can write the approximate posterior assuming a Gaussian input distribution as follows:

$$q^{(t)}(\mathbf{x}) = \sum_k q^{(t)}(\mathbf{x}|C = k)\, q^{(t)}(C = k) = \sum_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k^{(t)}, \sigma^2 \boldsymbol{I})\, \mathbf{m}_k^{(t)}. \tag{6.2}$$

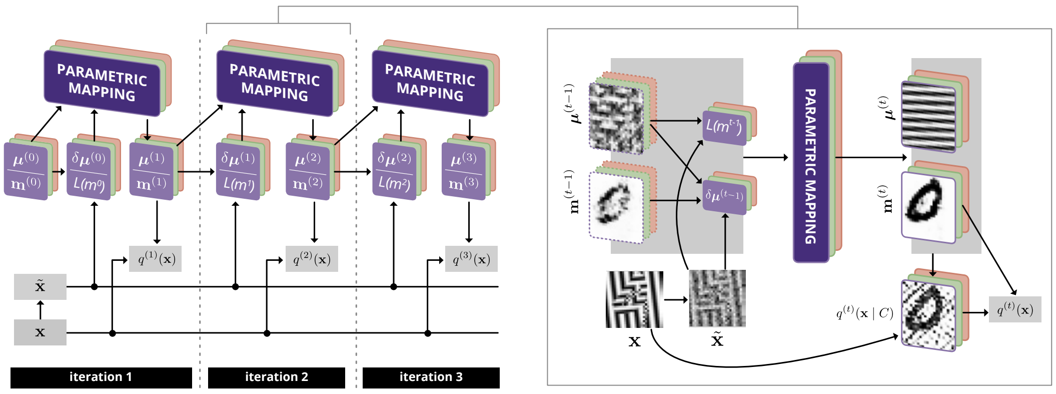


Figure 6.1. Illustration of the TAG framework used for training. Left: The system learns by denoising its input over iterations using several groups to distribute the representation. Each group, represented by several panels of the same color, maintains its own estimate of reconstructions $\boldsymbol{\mu}^{(t)}$ of the input, and corresponding masks $\mathbf{m}^{(t)}$, which encode the parts of the input that this group is responsible for representing. These estimates are updated over iterations by the same network, that is, each group and iteration share the weights of the network and only the inputs to the network differ. In the case of images, $\boldsymbol{\mu}$ contains pixel-values. Right: In each iteration $\boldsymbol{\mu}^{(t-1)}$ and $\mathbf{m}^{(t-1)}$ from the previous iteration, are used to compute a likelihood term $L(\mathbf{m}^{(t-1)})$ and modeling error $\delta\boldsymbol{\mu}^{(t-1)}$. These four quantities are fed to the parametric mapping to produce $\boldsymbol{\mu}^{(t)}$ and $\mathbf{m}^{(t)}$ for the next iteration. During learning, all inputs to the network are derived from the corrupted input as shown here. The unsupervised task for the network is to learn to denoise, i.e. output an estimate $q(\mathbf{x})$ of the original clean input.

Throughout this work we assume a constant global variance of the Gaussian distribution but which is learned from the data.

6.1.2 Inputs

In contrast to a normal denoising autoencoder which receives the corrupted $\tilde{\mathbf{x}}$, we instead feed the output estimates $\mathbf{m}_k^{(t-1)}$ and $\boldsymbol{\mu}_k^{(t-1)}$ from the previous iteration along with two additional quantities: the remaining modeling error $\delta\boldsymbol{\mu}_k^{(t-1)}$ and the group assignment likelihood ratio $L(\mathbf{m}_k^{(t-1)})$ which carry information about how the estimates can be improved: The latter two are functions of the previous groupwise $\mathbf{m}_k^{(t-1)}$ and $\boldsymbol{\mu}^{(t-1)_k}$ and the corrupted $\tilde{\mathbf{x}}$ They provide additional information that is easy to compute, but very useful for improving the estimates. A parametric mapping (here a neural network) then produces the new estimates $\mathbf{m}_k^{(t)}$ and $\boldsymbol{\mu}_k^{(t)}$.

Corrupted Image. For each input image $\mathbf{x}$ we produce a corrupted version $\tilde{\mathbf{x}}$ by applying a certain kind of noise. In the case of binary inputs we use bitflip noise for corruption:

$$\tilde{\mathbf{x}} = \mathbf{x} \oplus \mathcal{B}(\beta),$$

where $\oplus$ denotes componentwise XOR, and $\mathcal{B}(\beta)$ is Bernoulli distributed noise with probability β which we set to $\beta = 0.2$. In the case of real-valued inputs add Gaussian noise:

$$\tilde{\mathbf{x}} = \mathbf{x} + \mathcal{N}(0, \sigma_{\mathbf{x}}^2),$$

where $\sigma_{\mathbf{x}}$ is the standard deviation of the input noise which we also set to $\sigma_{\mathbf{x}} = 0.2$.

Initialization. Similar to expectation maximization, the initial values for $\mathbf{m}_k^{(0)}$ are randomized, and $\boldsymbol{\mu}_k^{(0)}$ is set to the data mean for all k:

$$m_{k,i} = \frac{e^{\hat{m}_{k,i}}}{\sum_{h=1}^{K} e^{\hat{m}_{h,i}}}, \qquad \text{where} \quad \hat{m}_{k,i} \sim \mathcal{N}(0,1). \tag{6.3}$$

$$\mu_{k,i} = \mu_{\mathbf{x}} = \mathbb{E}\left[\frac{1}{D}\sum_{i=1}^{D} x_i\right]. \tag{6.4}$$

$$\tag{6.5}$$

In our experiments $\mu_{\mathbf{x}} = 0.5$ for the TextureMNIST datasets and $\mu_{\mathbf{x}} = 0.26$ for the Shapes dataset.

Modelling Error. Intuitively, the auxiliary input $\delta\boldsymbol{\mu}_k$ carries information about how far away the reconstruction $\boldsymbol{\mu}_k$ is from the input image $\mathbf{x}$. During training as a denoiser, we can only allow information about the corrupted $\tilde{\mathbf{x}}$ as inputs but not about the original clean $\mathbf{x}$. Therefore, we use the derivative of the the gradient of the negative log likelihood of the corrupted input wrt. $\boldsymbol{\mu}_k$ as information about the remaining modelling error. Assuming a Gaussian input distribution this amounts to [1]:

$$\delta\mu_{k,i} = m_{k,i}(\tilde{x}_i - \mu_{k,i}) \propto \frac{\partial}{\partial \mu_{k,i}}\left[-\log(\sum_h q(\tilde{x}_i \mid \mu_{h,i}, C = h))\right]. \tag{6.6}$$

[1]Refer to the appendix of [120] for the case of binary inputs.

Assignment Likelihood. Intuitively, the term $L(\mathbf{m}_k)$ describes how well each group reconstructs the individual input elements relative to the other groups. Again, to avoid leaking information of the clean input $\mathbf{x}$ we look at the group assignment likelihood of the corrupted input. Unlike $\mu_{k,i}$ the $\mathbf{m}_{:,i}$'s correspond to categorical distributions, so we treat them slightly differently:

$$L(\mathbf{m}_{k,i}^{(t)}) \propto \frac{q^{(t)}(\tilde{x}_i \mid C = k)}{\sum_h q^{(t)}(\tilde{x}_i \mid C = h)}. \tag{6.7}$$

Note that we normalize $L(m_{k,i})$ over k such that it sums up to one for each value of i. This amounts to providing each group information about how likely each input element belongs to them rather than some other group. In other words, this is equivalent to likelihood ratio rather than the raw likelihood.

6.1.3 Parametric mapping

The final component needed in the TAG framework is the parametric model, which does all the heavy lifting of inference. This model has a dual task: first, to update the denoising estimates $\boldsymbol{\mu}_k$ of what each group predict about the input, and second, to update the group assignment probabilities $\mathbf{m}_k$ of each input element. The the remaining modeling error input is based on the corrupted input $\tilde{\mathbf{x}}$; thus, the parametric network has to denoise this and in effect implement posterior inference for the estimated quantities. The mapping function is the same for each group k and for each iteration. In other words, we share weights and in effect have only a single function approximator that we reuse.

The denoising task encourages the network to iteratively group its inputs into coherent groups that can be modeled efficiently. The trained network can be useful for a real-world denoising application, but typically, the idea is to encourage the network to learn interesting internal representations. Therefore, it is not $q(\mathbf{x})$ but rather $\mathbf{m}_k$, $\boldsymbol{\mu}_k$ and the internal representations of the parametric mapping that we are typically concerned with.

6.1.4 Training

An overview of the whole system is given in Figure 6.1. By using the log likelihood as the objective function, we train our system to compute an approximation $q^{(t)}(\mathbf{x})$ of the true denoising posterior $P_{X|\tilde{X}}(\mathbf{x}|\tilde{\mathbf{x}})$ at each iteration t:

$$\mathcal{L}(\mathbf{x}) = \sum_{t=0}^{T} \log q_X^{(t)}(\mathbf{x}), \tag{6.8}$$

where the summation is over iterations t. From here on we mostly omit t from the equations for readability. Since this cost function does not require any class labels or intended grouping information, training can be completely unsupervised, though additional terms for supervised tasks can be added too.

The trainable part of the TAG framework is given by a parametric mapping that operates independently on each group k and is used to compute both $\boldsymbol{\mu}_k^{(t)}$ and $\mathbf{m}_k^{(t)}$ (which is afterwards normalized using an elementwise softmax over the groups). This parametric mapping is usually implemented by a neural network and the whole system is trained end-to-end using standard backpropagation through time.

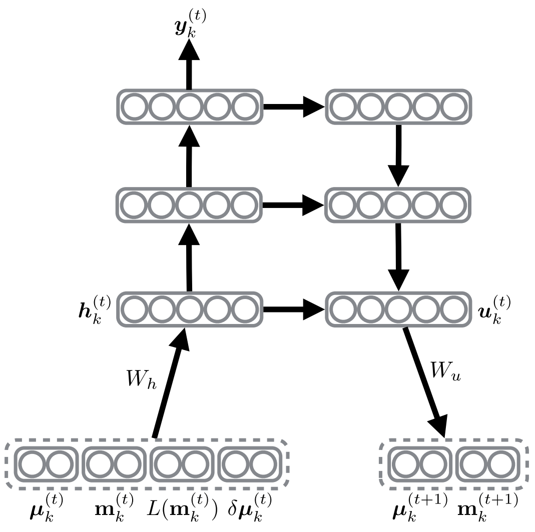

$y_k^{(t)}$
$h_k^{(t)}$
$u_k^{(t)}$
W_h
W_u
$\mu_k^{(t)}$ $\mathbf{m}_k^{(t)}$ $L(\mathbf{m}_k^{(t)})$ $\delta\mu_k^{(t)}$
$\mu_k^{(t+1)}$ $\mathbf{m}_k^{(t+1)}$

Figure 6.2. An example of how Tagger would use a 3-layer-deep Ladder Network as its parametric mapping to perform its iteration $i+1$. Note the optional class prediction output $y_k^{(t)}$ for classification tasks.

6.1.5 The Tagger: Combining TAG and Ladder Network

We chose the Ladder network [305] as the parametric mapping because its structure reflects the computations required for posterior inference in hierarchical latent variable models. This means that the network should be well equipped to handle the hierarchical structure one might expect to find in many domains. We call this Ladder network wrapped in the TAG framework *Tagger*. This is illustrated in Figure 6.2 and the corresponding pseudocode can be found in algorithm 1. Only the forward pass for a single example is shown, but derivatives of the cost $\mathcal{L}$ wrt. parameters σ, W_h, W_u and Θ are computed using regular backpropagation through time.

We mostly used the specifications of the Ladder network as described by Rasmus et al. [305], but there are some minor modifications we made to fit it to the TAG framework. We found that the model becomes more stable during iterations when we added a sigmoid function to the gating variable v [305, Equation 2] used in all the decoder layers with continuous outputs. None of the noise sources or denoising costs were in use (i.e., $\lambda_l = 0$ for all l in Eq. 3 of Rasmus et al. [305]), but Ladder's classification cost (C_c in Rasmus et al. [305]) was added to the Tagger's cost for the semi-supervised tasks.

All four inputs ($\boldsymbol{\mu}_k^{(t)}$, $\mathbf{m}_k^{(t)}$, $\delta\boldsymbol{\mu}_k^{(t)}$, and $L(\mathbf{m}_k^{(t)})$) were concatenated and projected to a hidden representation that served as the input layer of the Ladder Network. Subsequently, the values for the next iteration were simply read from the reconstruction ($\hat{\mathbf{x}}$ in Rasmus et al. [305]) and projected linearly into $\boldsymbol{\mu}_k^{(t+1)}$[2] and via softmax to $\mathbf{m}_k^{(t+1)}$ to enforce that they correspond to a categorical distribution.

6.2 Experiments and Results

We explore the properties and evaluate the performance of Tagger both in fully unsupervised settings and in semi-supervised tasks in two datasets[3]. Although both datasets consist of images

[2]For the binary case, we used a logistic sigmoid activation for $\boldsymbol{\mu}_k^{(t+1)}$.

[3]The datasets and a Theano [376] reference implementation of Tagger are available at `http://github.com/CuriousAI/tagger`

```
**Data:** $\mathbf{x}, K, T, \sigma_{\mathbf{x}}, \mu_{\mathbf{x}}$
**Trainable Parameters:** $W_h, W_u, \Theta, \sigma$
**Result:** $\boldsymbol{\mu}^{(T)}, \mathbf{m}^{(T)}, \mathcal{L}$
**begin** Initialization:
    $\tilde{\mathbf{x}} \leftarrow \mathbf{x} + \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{x}}^2 \boldsymbol{I})$;
    $\mathbf{m}^0 \leftarrow \mathrm{softmax}(\mathcal{N}(\mathbf{0}, \boldsymbol{I}))$;
    $\boldsymbol{\mu}^0 \leftarrow \mu_{\mathbf{x}}$;
**end**
**for** $t = 0 \ldots T-1$ **do**
    $\tilde{\boldsymbol{\mu}}_k \overset{k}{\leftarrow} \mathcal{N}(\tilde{\mathbf{x}}; \boldsymbol{\mu}_k^{(t)}, (\sigma^2 + \sigma_{\mathbf{x}}^2)\boldsymbol{I})$;
    $\delta\boldsymbol{\mu}_k^{(t)} \overset{k}{\leftarrow} (\tilde{\mathbf{x}} - \boldsymbol{\mu}_k^{(t)})\mathbf{m}_k^{(t)}\tilde{\boldsymbol{\mu}}_k$;
    $L(\mathbf{m}_k^{(t)}) \overset{k}{\leftarrow} \frac{\tilde{\boldsymbol{\mu}}_k}{\sum_h \tilde{\boldsymbol{\mu}}_h}$ ;
    $\boldsymbol{h}_k^{(t)} \overset{k}{\leftarrow} f(W_h \left[\boldsymbol{\mu}_k^{(t)}, \mathbf{m}_k^{(t)}, \delta\boldsymbol{\mu}_k^{(t)}, L(\mathbf{m}_k^{(t)})\right])$;
    $\boldsymbol{u}_k^{(t)} \overset{k}{\leftarrow} \mathrm{Ladder}(\boldsymbol{h}_k^{(t)}, \Theta)$;
    $[\boldsymbol{\mu}_k^{(t+1)}, \hat{\mathbf{m}}_k^{(t+1)}] \overset{k}{\leftarrow} W_u \boldsymbol{u}_k^{(t)}$;
    $\mathbf{m}^{(t+1)} \leftarrow \mathrm{softmax}(\hat{\mathbf{m}}^{(t+1)})$;
    $q^{(t+1)}(\mathbf{x}) \leftarrow \sum_{k=1}^{K} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k^{(t+1)}, \sigma^2 \boldsymbol{I})\, \mathbf{m}^{(t+1)}$;
**end**
$\mathcal{L} \leftarrow \sum_{t=1}^{T} \log q^{(t)}(\mathbf{x})$;
```

Algorithm 1: Pseudocode for running Tagger on a single real-valued example $\mathbf{x}$, where $f(x) = \max(x, 0)$ is the rectified linear function followed by BN. Note that $\overset{k}{\leftarrow}$ refers to a parallel assignment using all values of $1 \leq k \leq K$. For a binary-input version please refer to supplementary material of [120].

and grouping is intuitively similar to image segmentation, there is no prior in the Tagger model for images: our results (unlike the ConvNet baseline) generalize even if we permute all the pixels .

6.2.1 Datasets

Shapes. We use the simple Shapes dataset [308] to examine the basic properties of our system. It consists of 60,000 (train) + 10,000 (test) binary images of size 20x20. Each image contains three randomly chosen shapes (△▽□) composed together at random positions with possible overlap.

Textured MNIST. We generated a two-object supervised dataset (TextureMNIST2) by sequentially stacking two textured 28x28 MNIST-digits, shifted two pixels left and up, and right and down, respectively, on top of a background texture. The textures for the digits and background are different randomly shifted samples from a bank of 20 sinusoidal textures with different frequencies and orientations. Some examples from this dataset are presented in the column of Figure 6.4. We use a 50k training set, 10k validation set, and 10k test set to report the results. The dataset is assumed to be difficult due to the heavy overlap of the objects in addition to the clutter due to the textures. We also use a textured single-digit version (TextureMNIST1) without a shift to isolate the impact of texturing from multiple objects.

6.2.2 Training and evaluation

We train Tagger in an unsupervised manner by only showing the network the raw input example $\mathbf{x}$, not ground truth masks or any class labels, using 4 groups and 3 iterations. We average the cost over iterations and use *Adaptive Moment Estimation* (Adam; 198) with a batch-size of 100 for optimization. On the Shapes dataset we trained for 100 epochs with a bit-flip probability of 0.2, and on the TextureMNIST dataset for 200 epochs with a corruption-noise standard deviation of 0.2. The models reported in this paper took approximately 3 and 11 hours in wall clock time on a single Nvidia Titan X GPU for Shapes and TextureMNIST2 datasets respectively.

To understand how model size, length of the iterative inference, and the number of groups affect the modeling performance, we evaluate the trained models using two metrics: First, the denoising cost on the validation set, and second we evaluate the segmentation into objects using the *Adjusted Mutual Information* (AMI; 392) score and ignore the background and overlap regions in the Shapes dataset (consistent with Greff, Srivastava and Schmidhuber [122]). Evaluations of the AMI score and classification results in semi-supervised tasks were performed using uncorrupted input. The system has no restrictions regarding the number of groups and iterations used for training and evaluation. The results improved in terms of both denoising cost and AMI score when iterating further, so we used 5 iterations for testing. Even if the system was trained with 4 groups and 3 shapes per training example, we could test the evaluation with, for example, 2 groups and 3 shapes, or 4 groups and 4 shapes.

6.2.3 Unsupervised Perceptual Grouping

Table 6.1 shows the median performance of Tagger on the Shapes dataset over 20 seeds. Tagger is able to achieve very fast convergences, as shown in Table 6.1a. Through iterations, the network improves its denoising performances by grouping different objects into different groups.

	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5
Denoising cost	0.094	0.068	0.063	0.063	0.063
AMI	0.58	0.73	0.77	0.79	0.79
Denoising cost*	0.100	0.069	0.057	**0.054**	**0.054**
AMI*	0.70	0.90	0.95	0.96	**0.97**

(a) Convergence of Tagger over iterative inference

	AMI
RC [122]	0.61 ± 0.005
Tagger	0.79 ± 0.034
Tagger*	**0.97 ± 0.009**

(b) Method comparison

Table 6.1. Table (a) shows how quickly the algorithm evaluation converges over inference iterations with the Shapes dataset. Table (b) compares segmentation quality to previous work on the Shapes dataset. The AMI score is defined in the range from 0 (guessing) to 1 (perfect match). The results with a star (*) are using *LayerNorm* (LN; 14) instead of BN.

Comparing to Greff, Srivastava and Schmidhuber [122], Tagger performs significantly better in terms of AMI score (see Table 6.1b).

Figure 6.3 and Figure 6.4 qualitatively show the learned unsupervised groupings for the Shapes and textured MNIST datasets. Tagger uses its TAG mechanism slightly differently for the two datasets. For Shapes, $\boldsymbol{\mu}_k$ represents filled-in objects and masks $\mathbf{m}_k$ show which part of the object is actually visible. For textured MNIST, $\boldsymbol{\mu}_k$ represents the textures and masks $\mathbf{m}_k$ texture segments. In the case of the same digit or two identical shapes, Tagger can segment them into separate groups, and hence, it performs instance segmentation. We used 4 groups for training even though there are only 3 objects in the Shapes dataset and 3 segments in the TexturedMNIST2 dataset. The excess group is left empty by the trained system but its presence seems to speed up the learning process.

The hand-picked examples A-C in Figure 6.3 illustrate the robustness of the system when the number of objects changes in the evaluation dataset or when evaluation is performed using fewer groups.

Example E is particularly interesting; E_1 shows how the normal evaluation looks like but E_2 demonstrates how we can remove the topmost digit from the scene and let the system fill in digit below and the background. We do this by setting the corresponding group assignment probabilities $\mathbf{m}_k$ to a large negative number just before the final softmax over groups in the last iteration.

To solve the textured two-digit MNIST task, the system has to combine texture cues with high-level shape information. The system first infers the background texture and mask which are finalized on the first iteration. Then the second iteration typically fixes the texture used for topmost digit, while subsequent iterations clarify the occluded digit and its texture. This demonstrates the need for iterative inference of the grouping. Another evidence to support that the system has the high-level information and not just local cues is visible in hand-picked example D which shows how the system can use long-distance correlations to construct the occluded digit from three disjoint parts, even though it would have a fourth group available to represent them separately. This is further supported by semi-supervised experiments.

6.2.4 Classification

To investigate the role of grouping for the task of classification, we evaluate Tagger against four baseline models on the textured MNIST task. As our first baseline we use a fully connected

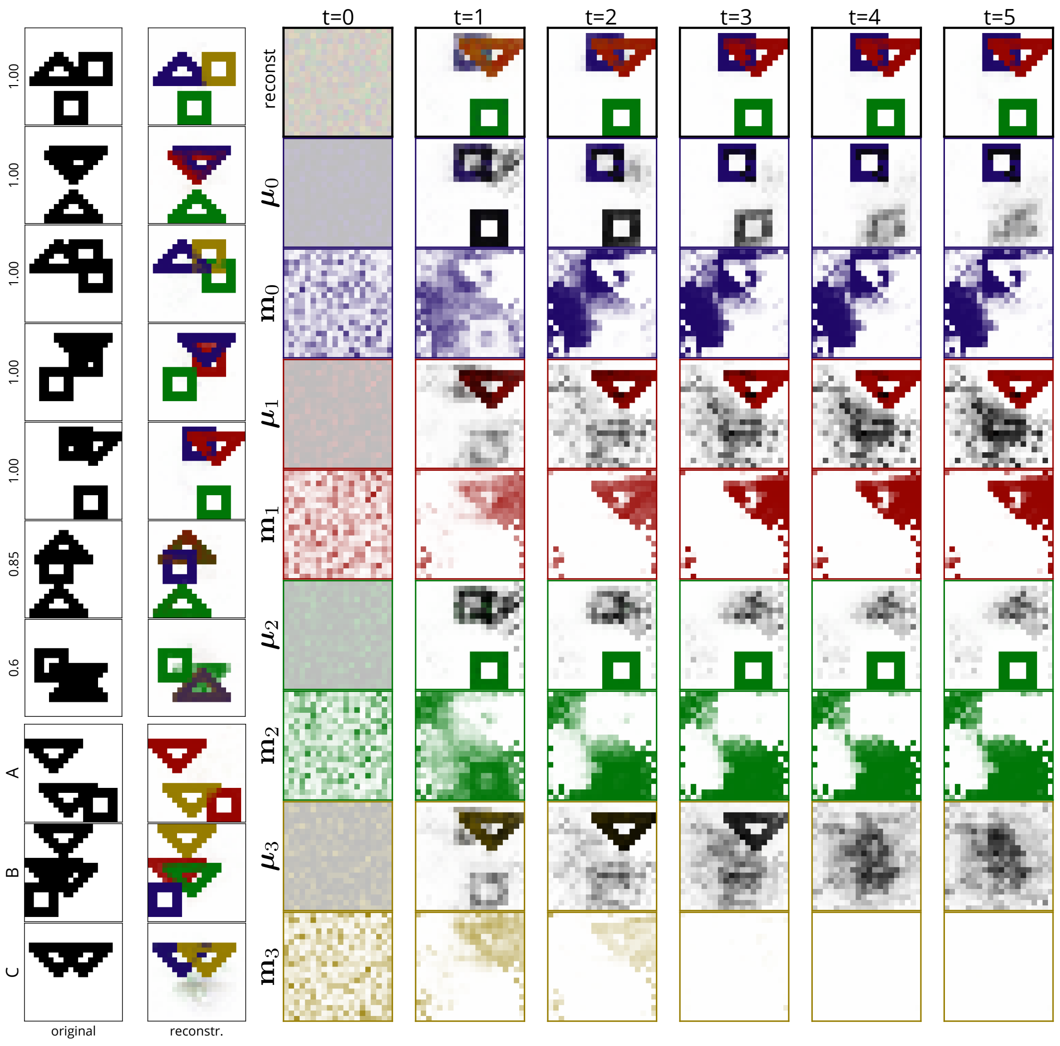


Figure 6.3. Results for Shapes dataset. Left column: 7 examples from the test set along with their resulting groupings in descending AMI score order and 3 hand-picked examples (A, B, and C) to demonstrate generalization. A: Testing 2-group model on 3 object data. B: Testing a 4-group model trained with 3-object data on 4 objects. C: Testing 4-group model trained with 3-object data on 2 objects. Right column: Illustration of the inference process over iterations for four color-coded groups; $\mathbf{m}_k$ and μ_k.

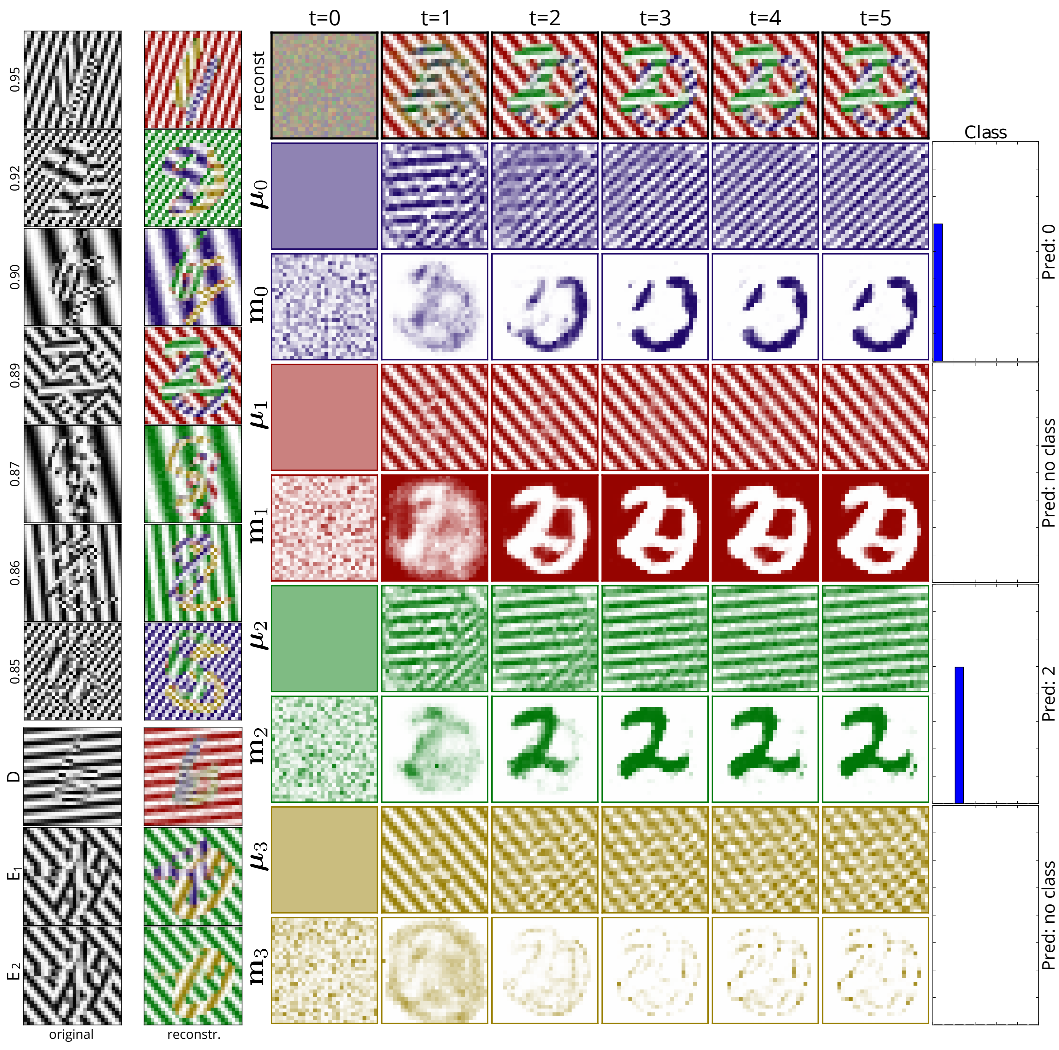


Figure 6.4. Results for the TextureMNIST2 dataset. Left column: 7 examples from the test set along with their resulting groupings in descending AMI score order and 3 hand-picked examples (D, E1, E2). D: An example from the TextureMNIST1 dataset. E1-2: A hand-picked example from TextureMNIST2. E1 demonstrates typical inference, and E2 demonstrates how the system is able to estimate the input when a certain group (topmost digit 4) is removed. Right column: Illustration of the inference process over iterations for four color-coded groups; $\mathbf{m_k}$ and $\boldsymbol{\mu}_k$.

network (*FC*) with *Rectified Linear Unit* (ReLU) activations and BN after each layer. Our second baseline is a ConvNet (*Conv*) based on Model C from [364], which has close to state-of-the-art results on CIFAR-10. We removed dropout, added BN after each layer and replaced the final pooling by a fully connected layer to improve its performance for the task. Furthermore, we compare with a fully connected Ladder [305] (FC Ladder) network.

All models use a softmax output and are trained with 50,000 samples to minimize the categorical cross entropy error. In case there are two different digits in the image (most examples in the TextureMNIST2 dataset), the target is $p = 0.5$ for both classes. We evaluate the models based on classification errors, which we compute based on the two highest predicted classes (top 2) for the two-digit case.

For Tagger, we first train the system in an unsupervised phase for 150 epochs and then add two fresh randomly initialized layers on top and continue training the entire system end to end using the sum of unsupervised and supervised cost terms for 50 epochs. Furthermore, the topmost layer has a per-group softmax activation that includes an added "no class" neuron for groups that do not contain any digit. The final classification is then performed by summing the softmax output over all groups for the true 10 classes and renormalizing it.

As shown in Table 6.2, Tagger performs significantly better than all the fully connected baseline models on both variants, but the improvement is more pronounced for the two-digit case. This result is expected because for cases with multi-object overlap, grouping becomes more important. Moreover, it confirms the hypothesis that grouping can help classification and is particularly beneficial for complex inputs. Remarkably, Tagger is on par with the convolutional baseline for the TexturedMNIST1 dataset and even outperforms it in the two-digit case, despite being fully connected itself. We hypothesize that one reason for this result is that grouping allows for the construction of efficient invariant features already in the low layers without losing information about the assignment of features to objects. Convolutional networks solve this problem to some degree by grouping features locally through the use of receptive fields, but that strategy is expensive and can break down in cases of heavy overlap.

6.2.5 Semi-Supervised Learning

The TAG framework does not rely on labels and is therefore directly usable in a semi-supervised context. For semi-supervised learning, the Ladder [305] is arguably one of the strongest baselines with SOTA results on 1,000 MNIST and 60,000 permutation invariant MNIST classification. We follow the common practice of using 1,000 labeled samples and 49,000 unlabeled samples for training Tagger and the Ladder baselines. For completeness, we also report results of the convolutional (*ConvNet*) and fully-connected (*FC*) baselines trained fully supervised on only 1,000 samples.

From the results in Table 6.2, it is obvious that all the fully supervised methods fail on this task with 1,000 labels. The best result of approximately 52 % error for the single-digit case is achieved by ConvNet, which still performs only at chance level for two-digit classification. The best baseline result is achieved by the *FC Ladder*, which reaches 30.5 % error for one digit but 68.5 % for TextureMNIST2.

For both datasets, Tagger achieves by far the lowest error rates: 10.5 % and 24.9 %, respectively. Again, this difference is amplified for the two-digit case, where the Tagger with 1,000 labels even outperforms the Ladder baseline with all 50k labels. This result matches our intuition that grouping can often segment out objects even of an unknown class and thus help select the relevant features for learning. This is particularly important in semi-supervised learning where

Dataset	Method	Error 50k	Error 1k	Model details
TextureMNIST1	FC MLP	31.1 ± 2.2	89.0 ± 0.2	2000-2000-2000 / 1000-1000
	FC Ladder	7.2 ± 0.1	30.5 ± 0.5	3000-2000-1000-500-250
	FC Tagger (ours)	**4.0 ± 0.3**	**10.5 ± 0.9**	3000-2000-1000-500-250
	ConvNet	**3.9 ± 0.3**	52.4 ± 5.3	based on Model C [364]
TextureMNIST2	FC MLP	55.2 ± 1.0	79.4 ± 0.3	2000-2000-2000 / 1000-1000
	FC Ladder	41.1 ± 0.2	68.5 ± 0.2	3000-2000-1000-500-250
	FC Tagger (ours)	**7.9 ± 0.3**	**24.9 ± 1.8**	3000-2000-1000-500-250
	ConvNet	12.6 ± 0.4	79.1 ± 0.8	based on Model C [364]

Table 6.2. Test-set classification errors for textured one-digit MNIST (chance level: 90 %) and top-2 error on the textured two-digit MNIST dataset (chance level: 80 %). We report mean and sample standard deviation over 5 runs. FC = Fully Connected

the inability to self-classify unlabeled samples can easily mean that the network fails to learn from them at all.

To put these results in context, we performed informal tests with five human subjects. The task turned out to be quite difficult and the subjects needed to have regular breaks to be able to maintain focus. The subjects improved significantly over training for a few days but there were also significant individual differences. The best performing subjects scored around 10 % error for TextureMNIST1 and 30 % error for TextureMNIST2. For the latter task, the test subject took over 30 seconds per sample.

6.3 Related work

Attention models have recently become very popular, and similar to perceptual grouping they help in dealing with complex structured inputs. These approaches are not, however, mutually exclusive and can benefit from each other. Overt attention models [343, 84] control a window (fovea) to focus on relevant parts of the inputs. Two of their limitations are that they are mostly tailored to the visual domain and are usually only suited to objects that are roughly the same shape as the window. But their ability to limit the field of view can help to reduce the complexity of the target problem and thus also help segmentation. Soft attention mechanisms [340, 65, 423] on the other hand use some form of top-down feedback to suppress inputs that are irrelevant for a given task. These mechanisms have recently gained popularity, first in machine translation [16] and then for many other problems such as image caption generation [420]. Because they re-weigh all the inputs based on their relevance, they could benefit from a perceptual grouping process that can refine the precise boundaries of attention.

Our work is primarily built upon a line of research based on the concept that the brain uses synchronization of neuronal firing to bind object representations together. This view was introduced by [395] and has inspired many early works on oscillations in neural networks (see the survey [395] for a summary). Simulating the oscillations explicitly is costly and does not mesh well with modern neural network architectures (but see [253]). Rather, complex values have been used to model oscillating activations using the phase as soft tags for synchronization [304, 309].

In our model, we further abstract them by using discretized synchronization slots (our groups). It is most similar to the models of Wersing et al. [411], Hyvärinen and Perkiö [164] and Greff, Srivastava and Schmidhuber [122]. However, our work is the first to combine this with denoising autoencoders in an end-to-end trainable fashion.

Another closely related line of research [333, 320] has focused on multi-causal modeling of the inputs. Many of the works in that area [223, 374, 358, 156] build upon Restricted Boltzmann Machines. Each input is modeled as a mixture model with a separate latent variable for each object. Because exact inference is intractable, these models approximate the posterior with some form of expectation maximization [69] or sampling procedure. Our assumptions are very similar to these approaches, but we allow the model to learn the amortized inference directly (more in line with Goodfellow et al. [108]).

Since *Recurrent Neural Networks* (RNNs; 250, 368, 316, 410) are general purpose computers, they can in principle implement arbitrary computable types of temporary variable binding [339, 340], unsupervised segmentation [339], and internal [340] and external attention [343]. For example, an RNN with fast weights [340] can rapidly associate or bind the patterns to which the RNN currently attends. Similar approaches even allow for metalearning [341], that is, learning a learning algorithm. Hochreiter et al. [151], for example, learned fast online learning algorithms for the class of all quadratic functions of two variables. Unsupervised segmentation could therefore in principle be learned by any RNN as a by-product of data compression or any other given task. That does not, however, imply that every RNN will, through learning, easily discover and implement this tool. From that perspective, TAG can be seen as a way of helping an RNN to quickly learn and efficiently implement a grouping mechanism.

The recurrent architecture most similar to the Tagger is the *Neural Abstraction Pyramid* (NAP; [26]) – a convolutional neural network augmented with lateral connections which help resolve local ambiguities and feedback connections that allow incorporation of high-level information. In early pioneering work the NAP was trained for iterative image binarization [28] and iterative image denoising [27], much akin to the setup we use. Being recurrent, the NAP layers too, could in principle learn a perceptual grouping as a byproduct. That does not, however, imply that every RNN will, through learning, easily discover and implement this tool. The main improvement that our framework adds is an explicit mechanism for the network to split the input into multiple representations and thus quickly and efficiently learn a grouping mechanism. We believe this special case of computation to be important enough for many real-world tasks to justify this added complexity.

6.4 Conclusion

In this chapter we have developed another approach for unsupervised perceptual grouping based on the idea of amortized inference. The Tagger learns the iterative inference required for grouping directly as a byproduct of trying to denoise the inputs. We have demonstrated the benefits of this mechanism for a heavily cluttered classification task, in which our fully connected Tagger even significantly outperformed a state-of-the-art convolutional network. More impressively, we have shown that our mechanism can greatly improve semi-supervised learning, exceeding conventional Ladder networks by a large margin. This is evidence, that the ability to group input elements and internal representations is a powerful tool that can improve a system's ability to handle complex multi-object inputs. A key contributor to this success compared to the previous chapters is the fact that the group assignments are also inferred by the network. This allows the system to model dependencies between the representation of an object and its segmentation, and to learn an efficient inference for both. One downside of the Tagger is that objects representations are no longer conveniently localized. While the Ladder-like structure of the network is very beneficial for denoising, it also disperses the information about the image throughout all the layers.

Chapter 7

Variational Iterative Multi-Object Representation Learning

Human perception is structured around objects which form the basis for our higher-level cognition and impressive systematic generalization abilities. Yet recent breakthroughs in unsupervised representation learning [141, 241, 53] tend to focus on data where a single object of interest is placed in front of some background (eg. dSprites, 3D Chairs, CelebA) without even considering multiple objects. In contrast, most visual scenes contain a variable number of objects arranged in various spatial configurations, and often with partial occlusions (eg., CLEVR, Johnson et al. 181; see Figure 7.1). Here we argue for the importance of learning to segment and represent objects *jointly*, maintain that discovery of objects in a scene should be considered a crucial aspect of representation learning, rather than treated as a separate problem. Similar to Chapters 4 to 6, we approach the problem from a spatial mixture model perspective [123] and use iterative inference to learn perceptual grouping. As with the *iTerative Amortized Grouping* (TAG; 120) framework we construct a network that amortizes an iterative inference of latent object representations and groupings simultaneously. However, here we will focus on the object representation aspect and use the successful *Variational Autoencoder* (VAE; 313, 199) framework [313, 199] to learn meaningful and disentangled representations for each object. We name the resulting architecture IODINE, and demonstrate that, starting from the simple assumption that a scene is composed of multiple entities, it is possible to learn to segment images into interpretable objects with disentangled representations. Our method learns – without supervision – to inpaint occluded parts, and extrapolates to scenes with more objects and to unseen objects with novel feature combinations on datasets like CLEVR [181], Objects Room [43]. This highlights the benefits of multi-object representation learning by comparison to a VAEs single-slot representations. We also show that, due to the use of iterative variational inference, our system is able to learn multi-modal posteriors for ambiguous inputs and extends naturally to sequences.

7.1 Method

We first express multi-object representation learning within the framework of generative modelling (Section 7.1.1). Then, building upon the successful VAE framework, we leverage variational inference to jointly learn both the generative and inference model (Section 7.1.2). There we also

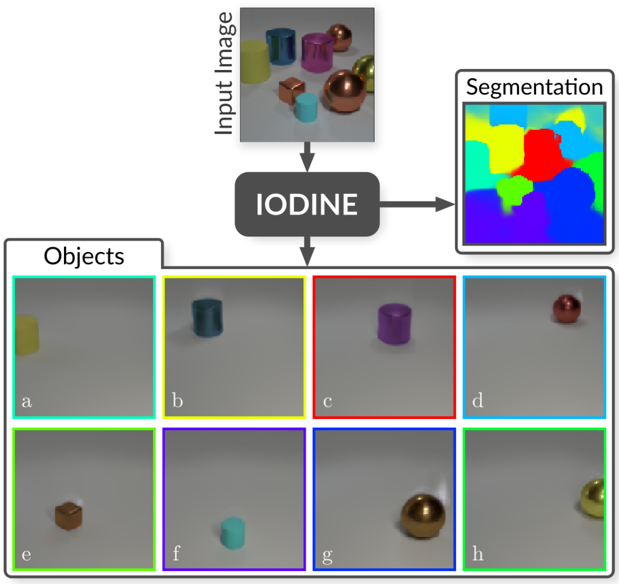


Figure 7.1. Object decomposition of an image from the CLEVR dataset by *Iterative Object Decomposition Inference NEtwork* (IODINE; 119). The model is able to decompose the image into separate objects in an unsupervised manner, inpainting occluded objects in the process (see slots (d), (e) and (h)).

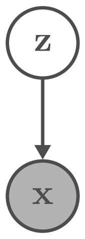
discuss the particular challenges that arise for inference in a multi-object context and show how they can be solved using iterative amortization. Finally, in Section 7.1.3 we bring all elements together and show how the complete system can be trained end-to-end.

7.1.1 Multi-Object Representations

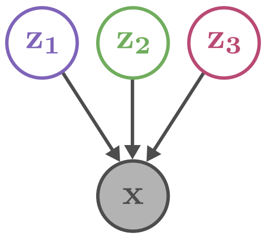
Flat vector representations as used by standard VAEs are inadequate for capturing the combinatorial object structure that many datasets exhibit. To achieve the kind of systematic generalization that is so natural for humans, we propose employing a *multi-slot* representation where each slot shares the underlying representation format, and each would ideally describe an independent part of the input. Consider the example in Figure 7.1: by construction, the scene consists of 8 objects, each with its own properties such as shape, size, position, color and material. To split objects, a flat representation would have to represent each object using separate feature dimensions. But this neglects the simple and (to us) trivial fact that they are interchangeable objects with common properties.

Generative Model We represent each scene with K latent object representations $\mathbf{z}_k \in \mathbb{R}^H$ that collaborate to generate the input image $\mathbf{x} \in \mathbb{R}^D$ (see Figure 7.2b). The $\mathbf{z}_k$ are assumed to be independent and their generative mechanism is shared such that any ordering of them produces the same image (i.e. entailing permutation invariance). Objects distinguished in this way can easily be compared, reused and recombined, thus facilitating combinatorial generalization.

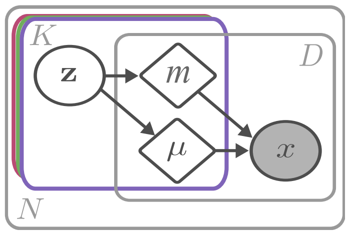
The image $\mathbf{x}$ is modeled with a spatial Gaussian mixture model where each mixing component (slot) corresponds to a single object. That means each object vector $\mathbf{z}_k$ is decoded into a pixel-wise mean μ_{ik} (the appearance of the object) and a pixel-wise assignment $m_{ik} = P_{C|Z}(C_i = k \mid \mathbf{z}_k)$ (the segmentation mask; see Figure 7.2c). Assuming that the pixels i are independent conditioned



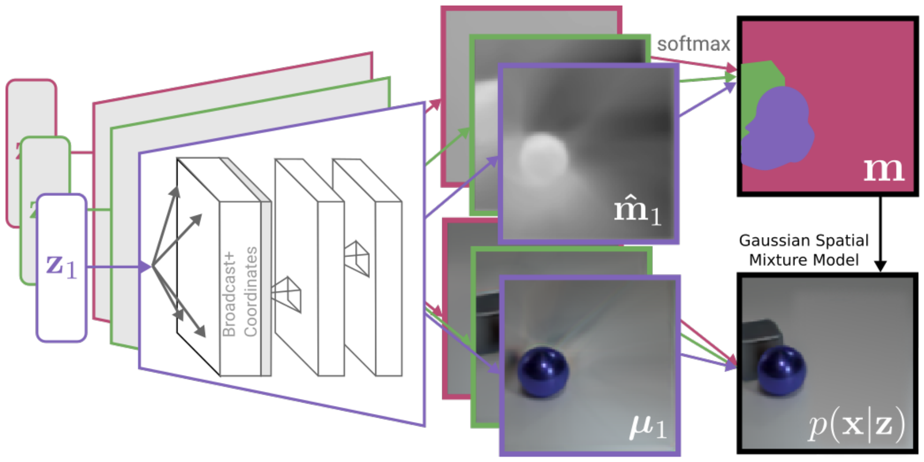
(a) VAE



(b) Multi-object VAE



(c) IODINE



(d) IODINE neural architecture.

Figure 7.2. Generative model illustrations. (a) A regular VAE decoder. (b) A hypothetical multi-object VAE decoder that recomposes the scene from three objects. (c) IODINEs multi-object decoder showing latent vectors (denoted $\boldsymbol{z}$) corresponding to K objects refined over T iterations from images of dimension D. The deterministic pixel-wise means and masks are denoted $\boldsymbol{\mu}$ and $\mathbf{m}$ respectively. (d) The neural architecture of the IODINEs multi-object spatial mixture decoder.

on $\mathbf{z}$, the likelihood thus becomes:

$$p_{X|Z}(\mathbf{x} \mid \mathbf{z}) = \prod_{i=1}^{D} \sum_{k=1}^{K} m_{ik} \mathcal{N}(x_i; \mu_{ik}, \sigma^2), \tag{7.1}$$

where we use a global fixed variance σ^2 for all pixels.

Decoder Structure Our decoder network structure directly reflects the structure of the generative model. See Figure 7.2d for an illustration. Each object latent $\mathbf{z}_k$ is decoded separately into pixel-wise means $\boldsymbol{\mu}_k$ and mask-logits $\hat{\mathbf{m}}_k$, which we then normalize using a softmax operation applied across slots such that the masks $\mathbf{m}_k$ for each pixel sum to 1. Together, $\boldsymbol{\mu}$ and $\mathbf{m}$ parameterize the spatial mixture distribution as defined in Equation (7.1). For the network architecture we use a broadcast decoder [405], which spatially replicates the latent vector $\mathbf{z}_k$, appends two coordinate channels (ranging from −1 to 1 horizontally and vertically), and applies a series of size-preserving convolutional layers. This structure encourages disentangling the position across the image from other features such as color or texture, and generally supports disentangling. All slots k share weights to ensure a common format, and are independently decoded, up until the mask normalization.

7.1.2 Inference

Similar to VAEs, we use amortized variational inference to get an approximate posterior $q_\lambda(\mathbf{z} \mid \mathbf{x})$ parameterized as a Gaussian with parameters $\boldsymbol{\lambda} = \{\boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\sigma}_\mathbf{z}\}$. However, our object-oriented generative model poses a few specific challenges for the inference process: Firstly, being a (spatial) mixture model, we need to infer both the components (i.e. object appearance) and the mixing (i.e. object segmentation). This type of problem is well known, for example in clustering and image segmentation, and is traditionally tackled as an iterative procedure, because there are no efficient direct solutions. A related second problem is that any slot can, in principle, explain any pixel. Once a pixel is explained by one of the slots, the others don't need to account for it anymore. This explaining-away property complicates the inference by strongly coupling it across the individual slots. Finally, slot permutation invariance induces a multimodal posterior with at least one mode per slot permutation. This is problematic, since our approximate posterior $q_\lambda(\mathbf{z} \mid \mathbf{x})$ is parameterized as a unimodal distribution. For all the above reasons, the standard feed-forward VAE inference model is inadequate for our case, so we consider a more powerful method for inference.

Iterative Inference The basic idea of iterative inference is to start with an arbitrary guess for the posterior parameters $\boldsymbol{\lambda}$, and then iteratively refine them using the input and samples from the current posterior estimate. We build on the framework of iterative amortized inference [247], which uses a trained refinement network f_ϕ. Unlike Marino et al., we consider only additive updates to the posterior and use several salient auxiliary inputs $\mathbf{a}$ to the refinement network (instead of just $\nabla_\lambda \mathcal{L}$). We update the posterior of the K slots independently and in parallel (indicated by $\overset{k}{\leftarrow}$ and $\overset{k}{\sim}$), as follows:

$$\mathbf{z}_k^{(t)} \overset{k}{\sim} q_\lambda(\mathbf{z}_k^{(t)} \mid \mathbf{x}) \tag{7.2}$$

$$\boldsymbol{\lambda}_k^{(t+1)} \overset{k}{\leftarrow} \boldsymbol{\lambda}_k^{(t)} + f_\phi(\mathbf{z}_k^{(t)}, \mathbf{x}, \mathbf{a}_k), \tag{7.3}$$

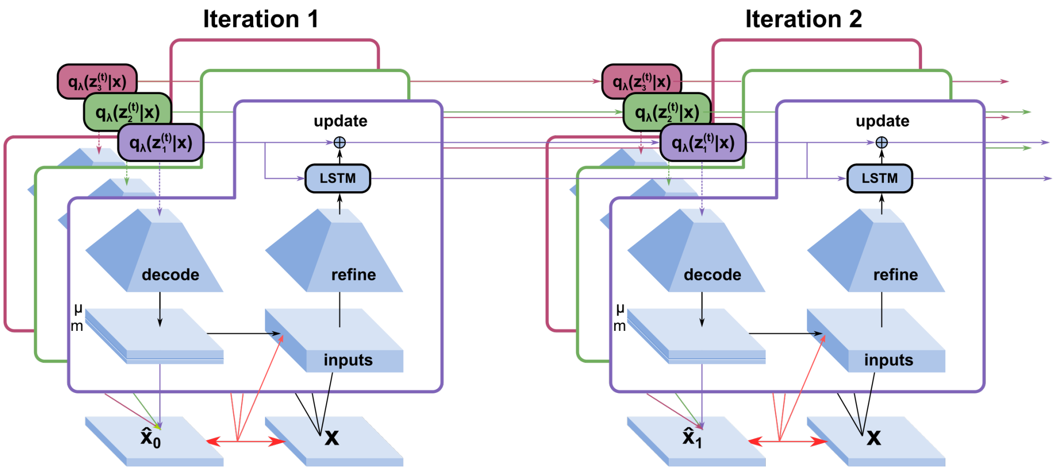


Figure 7.3. Illustration of the iterative inference procedure.

Thus the only place where the slots interact are at the input level. As refinement network f_ϕ we use a convolutional network followed by an LSTM. Instead of amortizing the posterior directly (as in a regular VAE encoder), the refinement network can be thought of as amortizing the gradient of the posterior [246]. The alternating updates to $q_\lambda(\mathbf{z} \mid \mathbf{x})$ and $P_{X|Z}(\mathbf{x} \mid \mathbf{z})$ are also akin to message passing.

```
Input: image $\mathbf{x}$, hyperparamters $K$, $T$, $\sigma^2$
Input: trainable parameters $\boldsymbol{\lambda}^{(1)}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$
Initialize: $\mathbf{h}_k^{(1)} \overset{k}{\leftarrow} \mathbf{0}$
for $t = 1$ to $T$ do
    $\mathbf{z}_k^{(t)} \overset{k}{\sim} q_\lambda(\mathbf{z}_k^{(t)} \mid \mathbf{x})$                          // Sample
    $\boldsymbol{\mu}_k^{(t)}, \hat{\mathbf{m}}_k^{(t)} \overset{k}{\leftarrow} g_\theta(\mathbf{z}_k^{(t)})$                          // Decode
    $\mathbf{m}^{(t)} \leftarrow \mathrm{softmax}_k(\hat{\mathbf{m}}_k^{(t)})$                          // Masks
    $P_{X|Z}(\mathbf{x} \mid \mathbf{z}^{(t)}) \leftarrow \sum_k \mathbf{m}_k^{(t)} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k^{(t)}, \sigma^2)$                          // Likelihood
    $\mathcal{L}^{(t)} \leftarrow D_{\mathrm{KL}}[q_\lambda(\mathbf{z}^{(t)} \mid \mathbf{x}) \,||\, P(\mathbf{z})] - \log P(\mathbf{x} \mid \mathbf{z}^{(t)})$
    $\mathbf{a}_k \overset{k}{\leftarrow} \mathrm{inputs}(\mathbf{x}, \mathbf{z}_k^{(t)}, \boldsymbol{\lambda}_k^{(t)})$                          // Inputs
    $\boldsymbol{\lambda}_k^{(t+1)}, \mathbf{h}_k^{(t+1)} \overset{k}{\leftarrow} f_\phi(\mathbf{a}_k, \mathbf{h}_k^{(t)})$                          // Refinement
end for
```

Algorithm 2: IODINE Pseudocode.

Inputs For each slot k we feed a set of auxiliary inputs $\boldsymbol{a}_k$ to the refinement network f_ϕ which then computes an update for the posterior $\boldsymbol{\lambda}_k$. Crucially, we include gradient information about the ELBO in the inputs, as it conveys information about what is not yet explained by other slots. Omitting the superscript (t) for clarity, we divide the auxiliary inputs $\mathbf{a}_k$ into image-sized inputs which are concatenated and fed to the corresponding convolutional network, and flat vector inputs which serve as inputs to the refinement LSTM. The following are the image-sized inputs where LN means Layernorm and SG means stop gradients:

Description	Formula	LN	SG	Ch.
image	$\mathbf{x}$			3
means	$\boldsymbol{\mu}$			3
mask	$\mathbf{m}_k$			1
mask-logits	$\hat{\mathbf{m}}_k$			1
mask posterior	$P(\mathbf{m}_k \mid \mathbf{x}, \boldsymbol{\mu})$			1
gradient of means	$\nabla_{\mu_k}\mathcal{L}$	✓	✓	3
gradient of mask	$\nabla_{\mathbf{m}_k}\mathcal{L}$	✓	✓	1
pixelwise likelihood	$P(\mathbf{x} \mid \mathbf{z})$	✓	✓	1
leave-one-out likelih.	$P(\mathbf{x} \mid \mathbf{z}_{i \neq k})$	✓	✓	1
coordinate channels				2
			total:	17

The posterior parameters $\boldsymbol{\lambda}$ and their gradients are flat vectors, and as such we concatenate them with the output of the convolutional part of the refinement network and use the result as input to the refinement LSTM. The approximate gradient $\nabla_{\lambda_k}\mathcal{L}$ is computed using the reparameterization trick by a backward pass through the generator network. This is computationally quite expensive, but we found that this information helps to significantly improve training of the refinement network. Like Marino, Yue and Mandt [247] we found it beneficial to normalize the gradient-based inputs with LayerNorm [14]. See Section 7.3.5 for an ablation study on these auxiliary inputs.

Description	Formula	LN	SG
gradient of posterior	$\nabla_{\lambda_k}\mathcal{L}$	✓	✓
posterior	$\boldsymbol{\lambda}_k$		

7.1.3 Training

We train the parameters of the decoder ($\boldsymbol{\theta}$), of the refinement network ($\boldsymbol{\phi}$), and of the initial posterior ($\boldsymbol{\lambda}^{(1)}$) by gradient descent through the unrolled iterations. In principle, it is enough to minimize the final negative ELBO $\mathcal{L}^T$, but we found it beneficial to use a weighted sum which also includes earlier terms:

$$\mathcal{L}_{\text{total}} = \sum_{t=1}^{T} \frac{t}{T} \mathcal{L}^{(t)}. \tag{7.4}$$

Each refinement step of IODINE uses gradient information to optimize the posterior $\boldsymbol{\lambda}$. Unfortunately, backpropagating through this process leads to numerical instabilities connected to double derivatives like $\nabla_\Theta \nabla_z \mathcal{L}$. We found that this problem can be mitigated by dropping the double derivative terms, i.e. stopping the gradients from backpropagating through the gradient-inputs $\nabla_{\mu_k}\mathcal{L}$, $\nabla_{\mathbf{m}_k}\mathcal{L}$, and $\nabla_{\lambda_k}\mathcal{L}$.

Initialization of Posterior IODINE iteratively refines an initial posterior $\boldsymbol{\lambda}^{(1)}$ which is independent of the input data. Initially we set this initial value to match the prior (i.e. $q_{\boldsymbol{\lambda}}(\mathbf{z}_k^{(1)}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$). But we found that this poses problems for the model, because of the competing requirements it poses for structuring the latent space wrt. the prior: On the one hand, samples from the prior need to be good starting values for iterative refinement. On the other hand, the prior should correspond to the accumulated posterior (KL term). For this reason we decided to simply make the parameters $\boldsymbol{\lambda}^{(1)}$ of the initialization distribution trainable parameters which are optimized alongside the weights of the decoder ($\boldsymbol{\theta}$) and of the refinement network ($\boldsymbol{\phi}$). This lead to faster training, and improved the visual quality of reconstructions from prior samples.

7.2 Related Work

Representation learning [29] has received much attention and has seen several recent breakthroughs. This includes disentangled representations through the use of β-VAEs [141], adversarial autoencoders [241], Factor VAEs [197], and improved generalization through non-euclidean embeddings [278]. However, most advances have focused on the feature-level structure of representations, and do not address the issue of representing multiple, potentially repeating objects, which we tackle here.

Another line of work is concerned with obtaining segmentations of images, usually without considering representation learning. This has led to impressive results on real-world images, however, many approaches (such as "semantic segmentation" or object detection) rely on supervised signals [104, 136, 306], while others require hand-engineered features [347, 87]. In contrast, as we learn to both segment and represent, our method can perform inpainting (Figure 7.1) and deal with ambiguity (Figure 7.17), going beyond what most methods relying on feature engineering are currently able to do.

Works tackling the full problem of scene representation are rarer. Probabilistic programming based approaches, like stroke-based character generation [218] or 3D indoor scene rendering [67], have produced appealing results, but require carefully engineered generative models, which are typically not fully learned from data. Work on end-to-end models has shown promise in using autoregressive inference or generative approaches [84, 125] Few methods can achieve similar comparable with the complexity of the scenes we consider here.

Two other methods related to ours are *Neural Expectation Maximization* (N-EM; 123) (along with its sequential and relational extensions [385]) and Tagger [120]. N-EM uses recurrent neural networks to amortize expectation maximization for a spatial mixture model. However, N-EM variants fail to cope with colored scenes, as we note in our comparison in Section 7.3.3. Tagger also uses iterative inference to segment and represent images based on a denoising training objective. We disregard Tagger for our comparison, because (1) its use of a Ladder network means that there is no bottleneck and thus no explicit object representations, and (2) without adapting it to a convolutional architecture, it does not scale to larger images (Tagger would require $\approx$ 600M weights for CLEVR).

MONet The Multi-Object NETwork (MONet; Burgess et al. 43) is a complementary method for unsupervised object representation learning also developed recently. It learns to sequentially attend to individual objects using a masking network and a VAE. In each step the masking network segments out a yet unexplained part of the image (the next object) which is then fed to the VAE which has to reconstruct that object and the mask. Thus, in contrast to IODINE, MONet uses one iteration per object and doesn't adjust an object once it has been covered.

Both methods focus on the representation learning aspect and both ensure that all objects are encoded in the same format by sharing weights across objects. In our preliminary experiments MONet produced results very similar to IODINE on CLEVR both in terms of segmentation and regarding the quality of object representations, and also learns to inpaint occluded parts of objects. See Section 7.3.3 shows a preliminary comparison between MONet and IODINE.

Since MONet only visits each object once, it is a more lightweight method that requires less computation and memory to train and run. Recurrently iterating over objects also has the benefit that the model can dynamically vary the number of objects, whereas in IODINE the maximum number of objects is a hyperparameter that has to be fixed manually (though it can be changed at test time). The usage of a separate masking network which isn't directly subject to a representational bottleneck likely leads to less regularization for the segmentation mask. This could potentially allow MONet to better deal with complex segmentation shapes. But it also has to use that ability to directly produce masks that respect occlusion, whereas IODINE tends to produce masks for full unoccluded objects and leverages the softmax to resolve overlap. For more complex scenes, we also expect iterative refinement to be advantageous for resolving difficult cases. There, IODINE could start with a rough segmentation and then use the progressively better understanding of the constituent objects for refining the boundaries.

The segmentation process of MONet is deterministic which induces an order on the objects, which might be useful because it naturally prioritizes salient objects. We observed that it typically starts with the background, then processes large frontal objects, and finally smaller or farther away objects. But this approach does break symmetry between objects, and we prefer keeping such a bias out of the object segmentation learning as much as possible.

Another disadvantage of a deterministic segmentation is that it cannot directly deal with ambiguous cases like the one shown in Section 7.3.6 and Figure 7.17. The iterative message-passing-like approach of IODINE might also lend itself well for incorporating top-down feedback to bias the segmentation towards one that is useful for a given task. It is less clear how to do that in MONet, though adding a way for conditioning the masking network could potentially serve a similar purpose. Finally the iterative refinement of IODINE naturally extends to sequential data (see Section 7.3.7) which would be less straightforward for MONet.

In summary, it is not at all clear yet which approach will work better and under which circumstances. If the data is sequential or contains ambiguity, IODINE presents a better choice. For other data that is not visually more complex than CLEVR, both methods will likely produce similar results making MONet the simpler and less computationally intensive choice. For more complex data it is unclear yet which approach would be the better choice, and in fact a hybrid approach might be the most promising. Sequentially attending to objects and iterative refinement are not mutually exclusive and might support each other. We consider this a very attractive research direction and are excited to explore its possibilities.

7.3 Results

Unless otherwise specified all the models are trained with the *Adaptive Moment Estimation* (Adam; 198) optimizer, with default parameters, a learning rate of 0.0003, and a batch size of 32 (4×8GPUs). We used gradient clipping as recommended by [290]: if the norm of global gradient exceeds 5.0 then the gradient is scaled down to that norm. Note that this is virtually always the case as the gradient norm is typically on the order of 10^5, but we nonetheless found it useful to apply this strategy. We always use $\sigma = 0.1$ for the global scale of the output distribution $p(\mathbf{x} \mid \mathbf{z}^{(t)}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k^{(t)}, \sigma^2)$. We varied several hyperparameters, including: number of slots, dimensionality of $\mathbf{z}_k$, number of inference iterations, number of convolutional layers and their filter sizes.

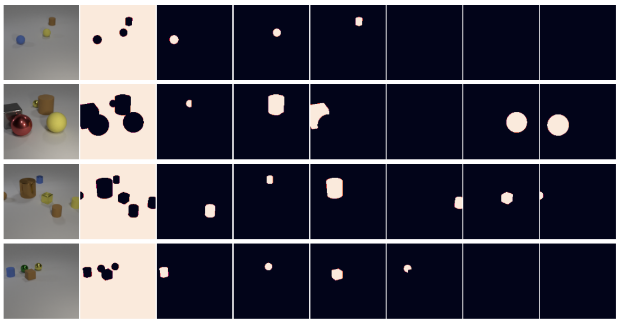


Figure 7.4. Samples from CLEVR6. The first column is the scene, the second column is the background mask and the following columns are the ground-truth object masks.

7.3.1 Datasets

We evaluate our model on three main datasets: 1) CLEVR [181] and a variant CLEVR6 which uses only scenes with up to 6 objects, 2) a multi-object version of the dSprites dataset [249], and 3) a dataset of multiple "Tetris"-like pieces that we created.

CLEVR We regenerated the CLEVR dataset [181] using the authors' open-source code, because we needed ground-truth segmentation masks for evaluation purposes. The dataset contains 70 000 images with a resolution of 240×320 pixels, from which we extract a square center crop of 192×192 and scale it to 128×128 pixels. Each scene contains between three and ten objects, characterized in terms of shape (cube, cylinder, or sphere), size (small or large), material (rubber or metal), color (8 different colors), position (continuous), and rotation (continuous). The subset of images which contain 3-6 objects (inclusive) served as the training set for our experiments; we refer to it as CLEVR6. Unless noted otherwise, we evaluate models on the full CLEVR distribution, containing 3-10 objects. All references to CLEVR refer to the full distribution. We do not make use of the question answering task. Figure 7.4 shows a few samples from the dataset.

Multi-dSprites This dataset, based on the dSprites dataset [249], consists of 60 000 images with a resolution of 64×64. Each image contains two to five random sprites, which vary in terms of shape (square, ellipse, or heart), color (uniform saturated colors), scale (continuous), position (continuous), and rotation (continuous). Furthermore the background color is varied in brightness but always remains grayscale. Figure 7.5 shows a few samples from the dataset. We also used a binarized version of Multi-dSprites, where the sprites are always white, the background is always black, and each image contains two to three random sprites.

Tetris We generated this dataset of 60 000 images by placing three random Tetrominoes without overlap in an image of 35×35 pixels. Each Tetromino is composed of four blocks that are each 5×5 pixels. There are a total of 17 different Tetrominoes (counting rotations). We randomly color each Tetromino with one of 6 colors (red, green, blue, cyan, magenta, or yellow). Figure 7.6 shows a few samples from the dataset.

Shapes We use the same shapes dataset as in [309]. It contains 60 000 binary images of size 28×28 each with three random shapes from the set $\{\triangle, \nabla, \square\}$.

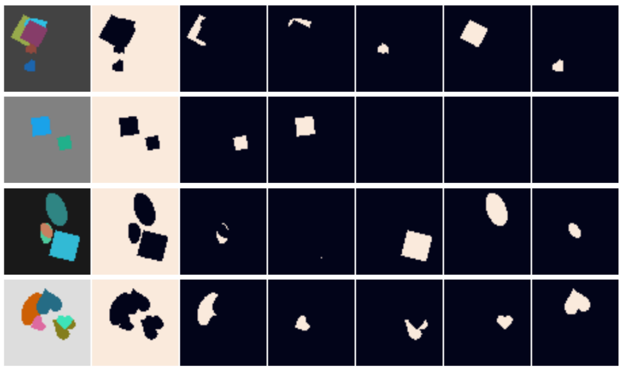


Figure 7.5. Samples from the Multi-dSprites dataset. The first column is the full image, the second column is the background mask and the following columns are the ground-truth object masks.

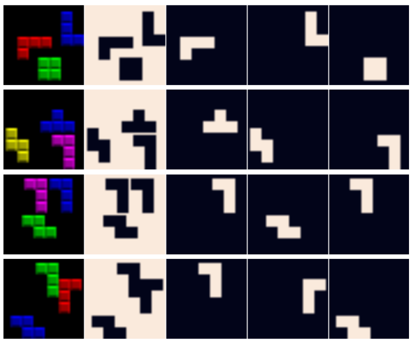


Figure 7.6. Samples from the Tetris dataset. The first column is the full image, the second column is the background mask and the following columns are the ground-truth object masks.

Objects Room For the preliminary sequential experiments we used a sequential version of the *Objects Room* dataset [43]. This dataset consists of 64x64 RGB images of a cubic room, with randomly colored walls, floors and objects randomly scattered around the room. The camera is always positioned on a ring inside the room, always facing towards the centre and oriented vertically in the range $(-25°, 22°)$. There are 3 randomly shaped objects in the room with 1-3 objects visible in any given frame. This version contains sequences of camera-flights for 16 time steps, with the camera position and angle (within the above constraints) changing according to a fixed velocity for the entire sequence (with a random velocity sampled for each sequence).

7.3.2 Architecture and Hyperparameters

All layers use the ELU [57] activation function and the Convolutional layers use a stride equal to 1, unless mentioned otherwise.

CLEVR All models were trained on scenes with 3-6 objects (CLEVR6) with $K = 7$ slots and $T = 5$ iterations, and a latent object dimension of size $\dim(\mathbf{z}_k) = 64$. When evaluating on the full CLEVR dataset, we increased the number of slots to $K = 11$. For some of the analysis, we varied T and K as mentioned in the text. The architecture and hyperparameters are described in the following tables:

Multi-dSprites Models were trained with $K = 6$ slots, and used $T = 5$ iterations.

Decoder

Type	Size/*Ch.*	Act. Func.	Comment
Input: $\mathbf{z}$	64		
Broadcast	*66*		+ coordinates
Conv 3 × 3	*64*	ELU	
Conv 3 × 3	*64*	ELU	
Conv 3 × 3	*64*	ELU	
Conv 3 × 3	*64*	ELU	
Conv 3 × 3	*4*	Linear	RGB + Mask

(a)

Refinement Network

Type	Size/*Ch.*	Act. Func.	Comment
MLP	128	Linear	
LSTM	256	Tanh	
Concat $[\boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}}\mathcal{L}]$	512		
MLP	256	ELU	
Avg. Pool	64		
Conv 3 × 3	*64*	ELU	stride 2
Conv 3 × 3	*64*	ELU	stride 2
Conv 3 × 3	*64*	ELU	stride 2
Conv 3 × 3	*64*	ELU	stride 2
Inputs	*17*		

(b)

Table 7.1. Network architectures used for CLEVR dataset.

Deconv Decoder

Type	Size/*Ch.*	Act. Func.	Comment
Input: $\mathbf{z}$	64		
MLP	512	ELU	
MLP	512	ELU	
Reshape	*8*		8 × 8 × 8
Conv 5 × 5	*64*	ELU	stride 2
Conv 5 × 5	*64*	ELU	stride 2
Conv 5 × 5	*64*	ELU	stride 2
Conv 5 × 5	*64*	ELU	stride 2
Conv 5 × 5	*64*	ELU	
Conv 5 × 5	*4*	Linear	RGB + Mask

Table 7.2. Deconv-based decoder architecture used in Section 7.3.5.

Decoder

Type	Size/*Ch.*	Act. Func.	Comment
Input: $\boldsymbol{\lambda}$	32		
Broadcast	*34*		+ coordinates
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*4*	Linear	RGB + Mask

(a)

Refinement Network

Type	Size/*Ch.*	Act. Func.	Comment
MLP	32	Linear	
LSTM	128	Tanh	
Concat $[\boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}}\mathcal{L}]$	192		
MLP	128	ELU	
Avg. Pool	32		
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Inputs	*17*		

(b)

Table 7.3. Network architectures used for Multi-dSprites dataset.

Decoder

Type	Size/*Ch.*	Act. Func.	Comment
Input: $\boldsymbol{\lambda}$	64		
Broadcast	*66*		+ coordinates
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*4*	Linear	RGB + Mask

(a)

Refinement Network

Type	Size/*Ch.*	Act. Func.	Comment
MLP	64	Linear	
Concat $[\boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}}\mathcal{L}]$	256		
MLP	128	ELU	
Avg. Pool	32		
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Conv 5 × 5	*32*	ELU	
Inputs	*17*		

(b)

Table 7.4. Network architectures used for Tetris dataset.

Tetris Models were trained with $K = 4$ slots, and used $T = 5$ iterations. For Tetris, in contrast to the other models, we did not use an LSTM in the refinement network.

7.3.3 Decomposition

IODINE can provide a readily interpretable segmentation of the data, as seen in Figure 7.7. These examples clearly demonstrate the models ability to segmenting out the same objects which were used to generate the dataset, despite never having received supervision to do so. To quantify segmentation quality, we measure the similarity between ground-truth (instance) segmentations and our predicted object masks using the *Adjusted Rand Index* (ARI; 301, 157). ARI is a measure of clustering similarity that ranges from 0 (chance) to 1 (perfect clustering) and can handle arbitrary permutations of the clusters. We apply it as a measure of instance segmentation quality by treating each foreground pixel (ignoring the background) as one point and its segmentation as cluster assignment. As shown in Table 7.5, IODINE achieves almost perfect ARI scores of around 0.99 for CLEVR6, and Tetris as well as a relatively good score of 0.77 for Multi-dSprites. The lower scores on Multi-dSprites are largely because IODINE struggles to produce sharp boundaries for the sprites, and we are uncertain as to the reasons for this behaviour.

We compare with *Multi-Object Network* (MONet; 43), following the CLEVR model implementation described in the paper except using fewer (7) slots and different standard deviations for the decoder distribution (0.06 and 0.1 for σ_{bg} and σ_{fg}, respectively), which gave better scores. With this, MONet obtained a similar ARI score (0.96) as IODINE on CLEVR6, and on Multi-dSprites it performed significantly better with a score of 0.90 (using the unmodified model). We also attempted to compare ARI scores to N-EM, but neither *Relational Neural Expectation Maximization* (R-NEM; 385) nor the simpler *Recurrent Neural Network Expectation Maximization* (RNN-EM; 123) variant could cope well with colored images. As a result, we could only compare with those methods on a binarized version of Multi-dSprites and the Shapes dataset. These scores are summarized in Table 7.5.

7.3.4 Representation Quality

Information Content The object-reconstructions in Figure 7.7 show that their representations contain all the information about the object. But in what format, and how usable is it? To answer

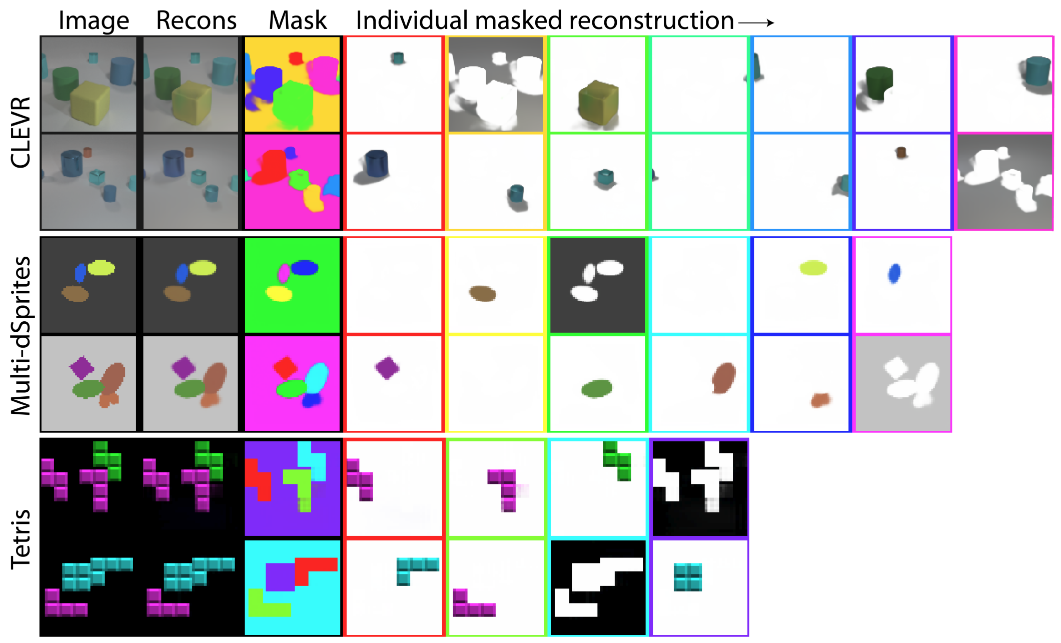


Figure 7.7. IODINE segmentations and object reconstructions on CLEVR6 (top), Multi-dSprites (middle), and Tetris (bottom). The individual masked reconstruction slots represent objects separately (along with their shadow on CLEVR). Border colours are matched to the segmentation mask on the left.

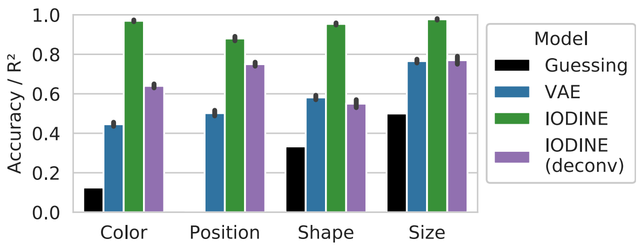


Figure 7.8. Prediction accuracy / R^2 score for the factor regression on CLEVR6. Position is continuous; the rest are categorical with 8 colors, 3 shapes, and 2 sizes. IODINE (deconv) does not use spatial broadcasting in the decoder.

	IODINE	R-NEM	MONet
CLEVR6	0.988	$*$	0.962 ± 0.006
M-dSprites	0.767 ± 0.056	$*$	0.904 ± 0.008
M-dSprites bin.	0.648 ± 0.172	0.685 ± 0.017	
Shapes	0.910 ± 0.119	0.776 ± 0.019	
Tetris	0.992 ± 0.004	$*$	

Table 7.5. Summary of IODINEs segmentation performance in terms of ARI (mean ± stddev across five seeds) versus baseline models. For each independent run, we computed the ARI score over 320 images, using only foreground pixels. We then picked the best hyperparameter combination for each model according to the mean ARI score over five random seeds.

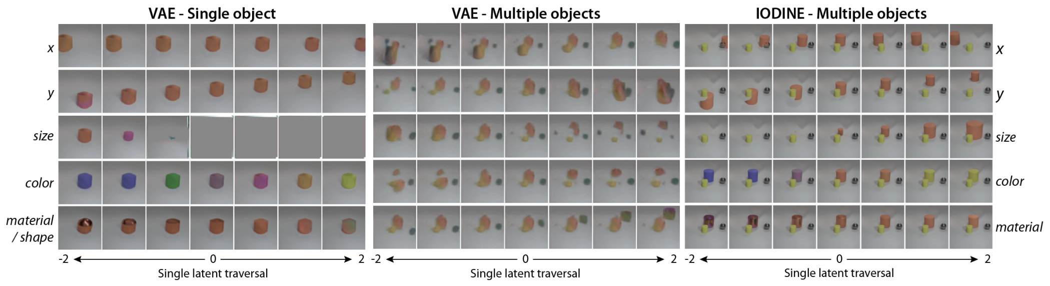


Figure 7.9. Disentanglement in regular VAEs vs IODINE. Rows indicate traversals of single latents, annotated by our interpretation of their effects. (Left) When a VAE is trained on single-object scenes it can disentangle meaningful factors of variation. (Center) When the same VAE is trained on multi-object scenes, the latents entangle across both factors and objects. (Right) In contrast, traversals of individual latents in IODINE vary individual factors of single objects, here the orange cylinder. Thus, the architectural bias for discovering multiple entities in a common format enables not only the discovery of objects, but also facilitates disentangling of their features.

this question we associate each ground-truth object with its corresponding $\mathbf{z}_k$ based on the segmentation masks. We then train a single-layer network to predict ground-truth factors for each object. Note that this predictor is trained *after* IODINE has finished training (i.e. no supervised fine-tuning). It tells us if a linear mapping is sufficient to extract information like color, position, shape or size of an object from its latent representation, and gives an important indication about the usefulness of the representation. Results in Figure 7.8 clearly show that a linear mapping is sufficient to extract relevant information about these object attributes from the latent representation to high accuracy. This result is in contrast with the scene representations learned by a standard VAE. Here even training the factor-predictor is difficult, as there is no obvious way to align objects with features. To make this comparison, we chose a canonical ordering of the objects based on their size, material, shape, and position (with decreasing precedence). The precedence of features was intended as a heuristic to maximize the predictability of the ordering. We then trained a linear network to predict the concatenated features of the canonically ordered objects from the latent scene representation. As the results in Figure 7.8 indicate, the information is present, but in a much less explicit/usable state.

Disentanglement Disentanglement is another important desirable property of representations [29] that captures how well learned features separate and correspond to individual, interpretable factors of variation in the data. While its precise definition is still highly debated [140, 79, 314, 235], the concept of disentanglement has generated a lot of interest recently. Good disentanglement is believed to lead to both better generalization and more interpretable features [219, 142]. Interestingly, for these desirable advantages to bear out, disentangled features seem to be most useful for properties of single objects, such as color, position, shape, etc. It is much less clear how to operationalize this in order to create disentangled representations of entire scenes with variable numbers of objects. And indeed, if we train a VAE that can successfully disentangle features of a single-object dataset, we find that that its representation becomes highly entangled on a multi-object dataset, (see Figure 7.9 left vs middle, and Figure 7.10). IODINE, on the other hand, successfully learns disentangled representations, because it is able to

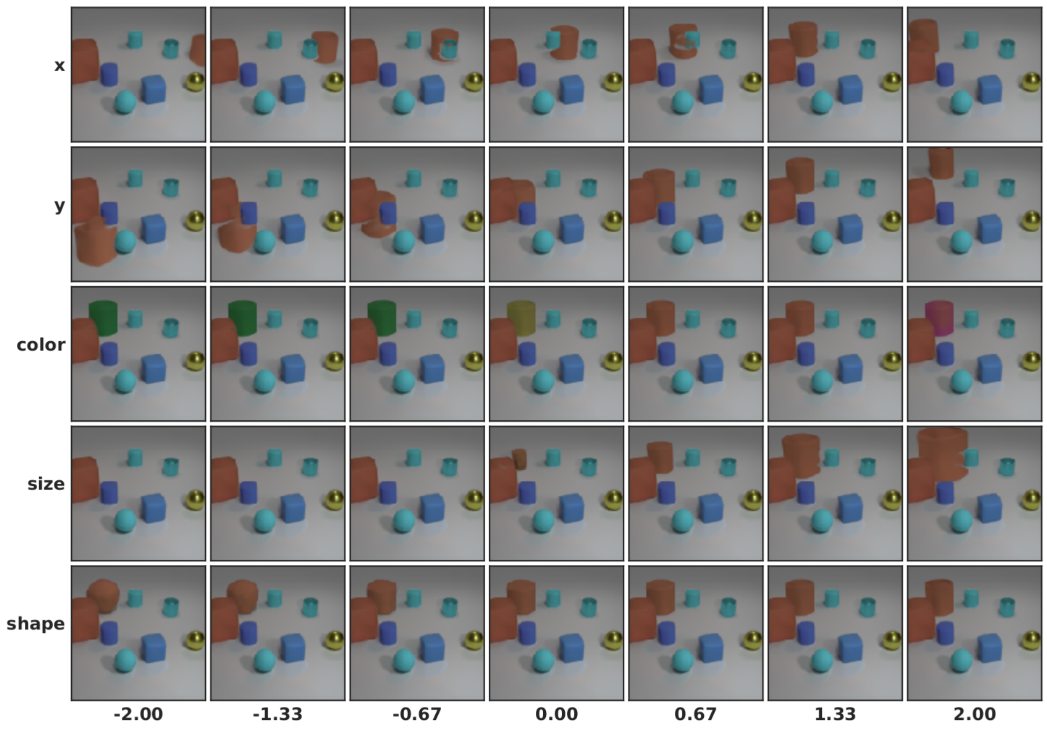


Figure 7.10. Latent traversal of IODINE on CLEVR (like right side of Figure 7.9), for a randomly chosen example and randomly chosen slot. Here the brown cylinder in the back is changing. Occlusion handling shows several flaws, that could be fixed by adjusting another latent (not shown) that encodes the depth ordering.

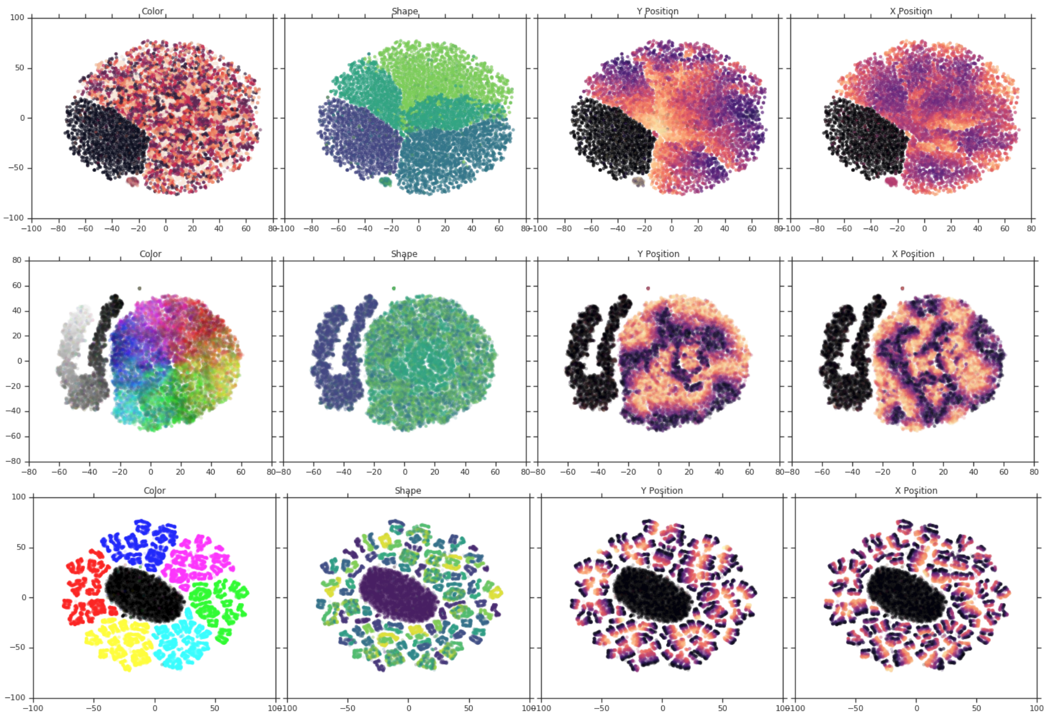


Figure 7.11. Each row shows the t-SNE of the latent distribution for the **CLEVR6**, **Multi-dSprites**, and **Tetris** datasets respectively. Each dot represents one object latent and in each column is colored according to a single ground truth factor. Note that representations of the background do not include a position and thus appear as large black areas.

first decompose the scene and then represent individual objects (Figure 7.9 right). In Figure 7.9 we show traversals of the most important features (selected by KL) of a standard VAE vs IODINE. While the standard VAE clearly entangles many properties even across multiple objects, IODINE is able to neatly separate them.

Projections of Object Latents Another way of of qualitatively evaluating the structure of the object representations is by visualizing the embedding space. Figure 7.11 shows how object latents are clustered when projected using t-SNE [384].

Generalization Finally, we can ask directly: Does the system generalize to novel scenes in a systematic way? Specifically, does it generalize to scenes with more or fewer objects than ever encountered during training? Slots are exchangeable by design, so we can freely vary the number of slots during test-time (more on this in Section 7.3.5). So in Figure 7.12 we qualitatively show the performance of a system that was trained with $K = 7$ on up to 6 objects, but evaluated with $K = 10$ on 9 objects. In Figure 7.14a the orange boxes show, that, even quantitatively, the segmentation performance decreases little when generalizing to more objects.

A more extreme form of generalization involves handling unseen feature combinations. To test this we trained our system on a subset of CLEVR that does not contain green spheres (though it does contain spheres and other green objects). And then we tested what the system does

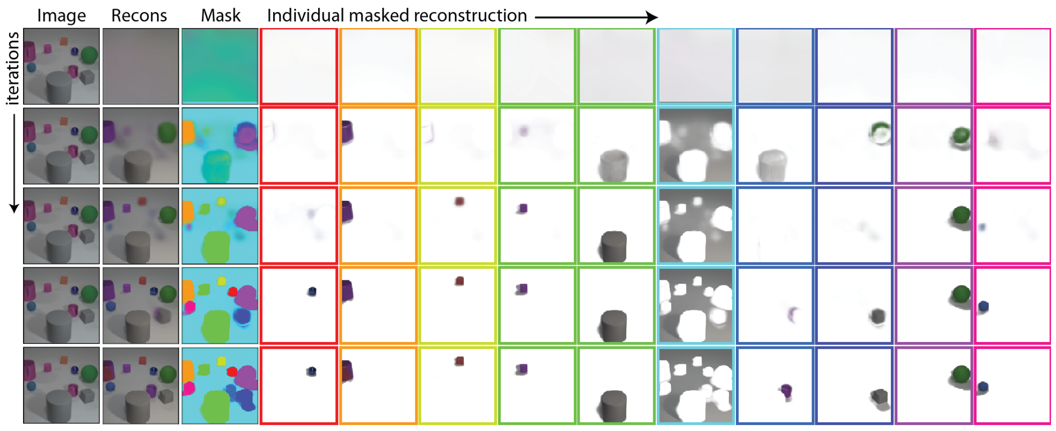


Figure 7.12. IODINEs iterative inference process and generalization capabilities. Rows indicate steps of iterative inference, refining reconstructions and segmentations when moving down the figure. Of particular interest is the *explaining away* effect visible between slots 6 and 7, where they settle on different objects despite both starting with the large cylinder. The model was only trained with $K = 7$ slots on 3-6 objects (excluding green spheres), and yet is able to generalize to $K = 10$ slots on a scene with 9 objects, including the never seen before green sphere (9th slot).

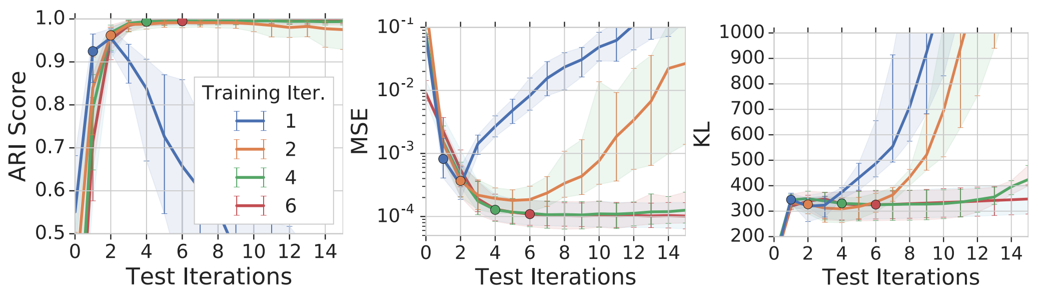
when confronted with a green sphere. In Figure 7.12 it can be seen that IODINE is still able to represent green spheres, despite never having seen this combination during training.

7.3.5 Robustness & Ablation

Now that we've established the usefulness of the object-representations produced by IODINE, we turn our attention to investigating its behavior in more detail.

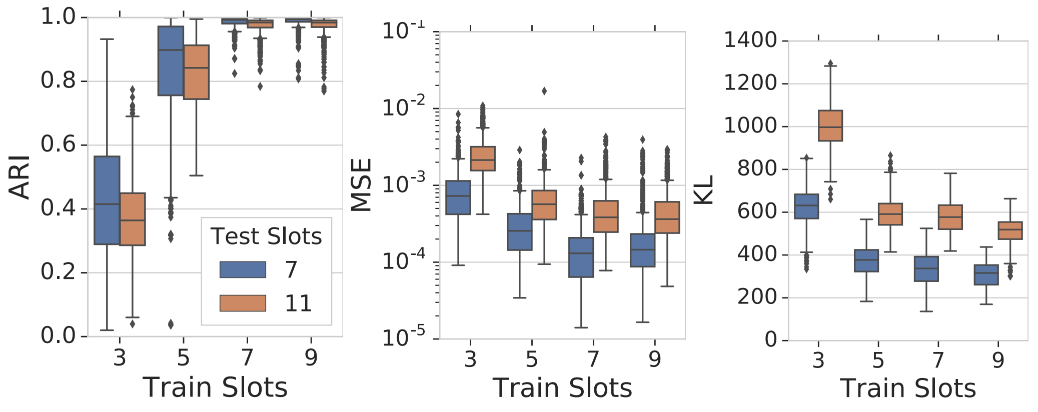
Iterations The number of iterations is one of the central hyperparameters to our approach. To investigate its impact, we trained four models with 1, 2, 4 and 6 iterations on CLEVR6, and evaluated them all using 15 iterations (see Figure 7.13). The first thing to note is that the inference converges very quickly within the first 3-5 iterations after which neither the segmentation nor reconstruction change much. The second important finding is that the system is very stable for much longer than the number of iterations it was trained with. The model even further improves the segmentation and reconstruction when it is run for more iterations, though it eventually starts to diverge after about two to three times the number of training iterations as can be seen with the blue and orange curves in Figure 7.13.

Slots The other central parameter of IODINE is the number of slots K, as it controls the maximum number of objects the system can separate. It is important to distinguish varying K for training vs varying it at test-time. As can be seen in Figure 7.14, if the model was trained with sufficiently many slots to fit all objects ($K = 7$, and $K = 9$), then test-time behavior generalizes very well. Typical behavior (not shown) is to leave excess slots empty, and when confronted with too many objects it will often completely ignore some of them, leaving the other object-representations mostly intact. Given enough slots at test time, such a model can even



(a) ARI (b) MSE (c) KL

Figure 7.13. The effect of varying the number of iterations, for both training and at test time. (a) Median ARI score, (b) MSE and (c) KL over test-iterations, for models trained with different numbers of iterations on CLEVR6. The region beyond the filled dots thus shows test-time generalization behavior. Shaded region from 25th to 75th percentile.



(a) ARI (b) MSE (c) KL

Figure 7.14. IODINE trained on CLEVR6 with varying numbers of slots (columns). Evaluation of (a) ARI, (b) MSE, and (c) KL with 7 slots on 3-6 Objects (blue) and 11 slots on 3-9 objects (orange).

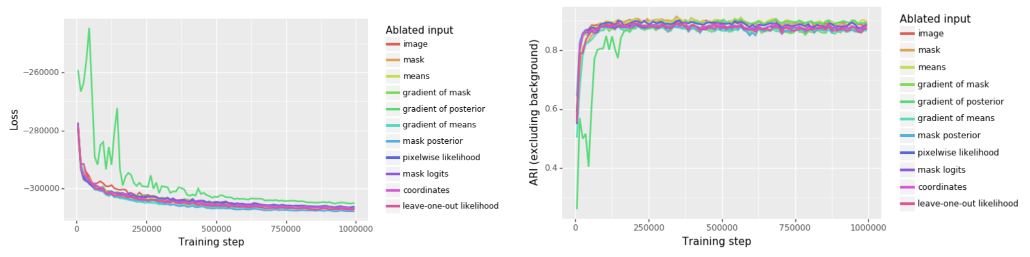


Figure 7.15. Ablation study for the model's total loss (left) and ARI (right) on the CLEVR6 dataset. Each curve denotes the result of training the model without a particular input.

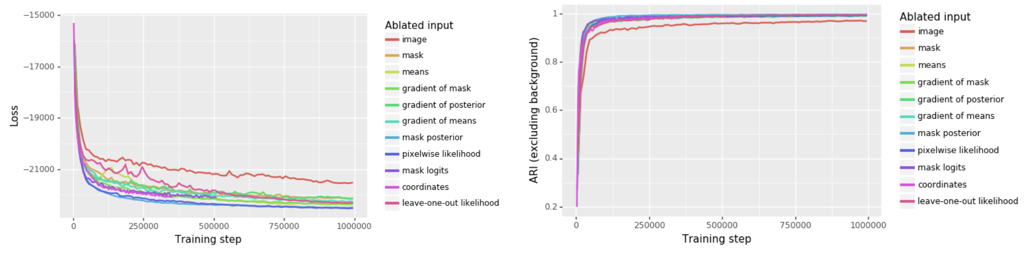


Figure 7.16. Ablation study for the model's total loss (left) and ARI (right) on the Tetris dataset. Each curve denotes the result of training the model without a particular input.

segment and represent scenes of higher complexity (more objects) than any scene encountered during training (see Figure 7.12 and the orange boxes in Figure 7.14). If on the other hand, the model was trained with too few slots ($K = 3$ and $K = 5$), its performance suffers substantially. This happens because, here the only way to reconstruct the entire scene during training is to consistently represent multiple objects per slot. And that leads to the model learning inefficient and entangled representations akin to the VAE in Figure 7.9 (also apparent from their much higher KL in Figure 7.14c). Once learned, this sub-optimal strategy cannot be mitigated by increasing the number of slots at test-time as can be seen by their decreased performance in Figure 7.14a.

Input Ablations We ablated each of the different inputs to the refinement network described in Section 7.1.2. Broadly, we found that individually removing an input did not noticeably affect the results (with two exceptions noted below). See Figures 7.15 and 7.16, which demonstrate this lack of effect in terms of the model's loss and the ARI segmentation score on both CLEVR6 and Tetris. A more comprehensive analysis could ablate combinations of inputs and identify synergistic or redundant groups, and thus potentially simplify the model. We didn't pursue this direction since none of the inputs incurs any noticeable computational overhead and at some point during our experimentation each of them contributed towards stable training behavior.

The main exceptions to the above are $\nabla_{\lambda}\mathcal{L}$ and $\mathbf{x}$. Computing the former requires an entire backward pass through the decoder, and contributes about 20% of the computational cost of the entire model. But we found that it often substantially improves performance and training convergence, which justifies its inclusion. A somewhat surprising finding was that for the Tetris dataset, removing $\mathbf{x}$ from the list of inputs had a pronounced detrimental effect, while for CLEVR

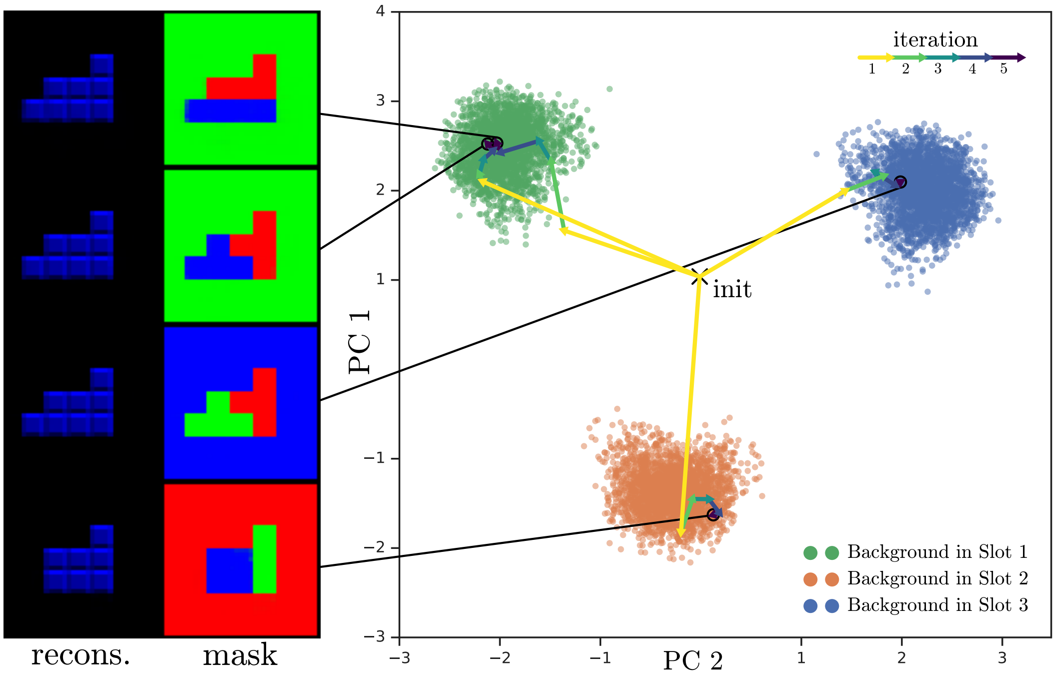


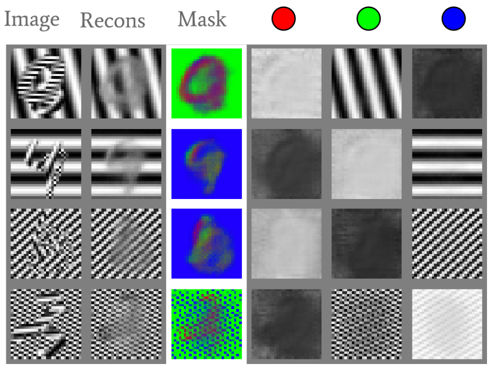
Figure 7.17. Multi-stability of segmentation when presented with an ambiguous stimulus. **Left:** Depending on the random sampling during iterative refinement, IODINE can produce different permutations of groups (row 2 vs 3), a different decomposition (row 1) or sometimes an invalid segmentation and reconstruction (row 4). **Right:** PCA of the latent space, coloured by which slot corresponds to the background. Paths show the trajectory of the iterative refinement for the four examples on the left.

it was negligible. At this point we do not have a good explanation for this effect.

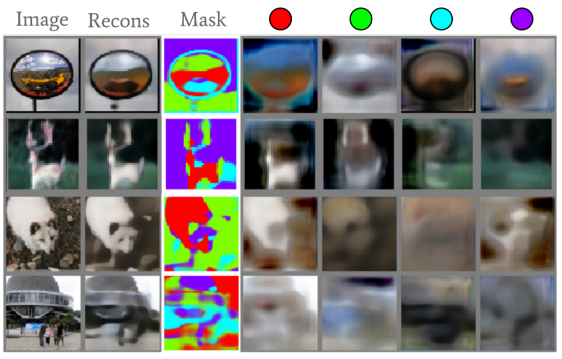
Broadcast Decoder Ablation We use the spatial broadcast decoder [405] primarily for its significant impact on the disentanglement of the representations, but its continuous spatial representation bias also seems to help decomposition. When replacing it with a deconvolution-based decoder the factor regression scores on CLEVR6 are significantly worse as can be seen in Figure 7.8. Especially for shape and size it now performs no better than the VAE which uses spatial broadcasting. The foreground-ARI scores also drop significantly (0.67 ± 0.06 down from 0.99) and the model seems less able to specialize slots to single objects. Note though, that these discrepancies might easily be reduced, since we haven't invested much effort in tuning the architecture of the deconv-based decoder.

7.3.6 Multi-Modality and Multi-Stability

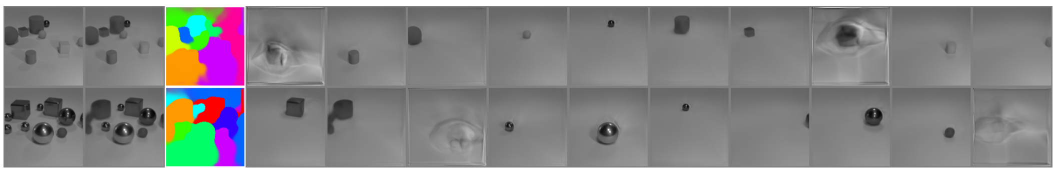
Standard VAEs are unable to represent multi-modal posteriors, because $q_\lambda(\mathbf{z} \mid \mathbf{x})$ is parameterized using a unimodal Gaussian distribution. However, as demonstrated in Figure 7.17, IODINE can actually handle this problem quite well. How is that possible? It turns out that this is an important side-effect of iterative variational inference, that to the best of our knowledge



(a) Textured MNIST



(b) ImageNet



(c) Grayscale CLEVR

Figure 7.18. Segmentation challenges a) IODINE did not succeed in capturing the foreground digits in the Textured MNIST dataset. b) IODINE groups ImageNet not into meaningful objects but mostly into regions of similar color. c) On a grayscale version of CLEVR, IODINE still produces the desired groupings.

has not been noticed before: The stochasticity at each iteration, which results from sampling **z** to approximate the likelihood, implicitly acts as an auxilliary (inference) random variable. This effect compounds over iterations, and is amplified by the slot-structure and the effective message-passing between slots over the course of iterations. In effect the model can implicitly represent multiple modes (if integrated over all ways of sampling **z**) and thus converge to different modes (see Figure 7.17 left) depending on these samples. This does not happen in a regular VAE, where no stochasticity enters the inference process. If we had an exact and deterministic way to compute the likelihood and its gradient, this effect would vanish.

A neat side-effect of this is the ability to elegantly capture ambiguous (aka multi-stable) segmentations such as the ones shown in Figure 7.17. We presented IODINE with an ambiguous arrangement of Tetris blocks, which has three different yet equally valid "explanations" (given the data distribution). When we evaluate a trained model on this image, we get different segmentations on different evaluations. Some of these correspond to different slot-orderings (1st vs 3rd row). But we also find qualitatively different segmentations (i.e. 3rd vs 4th row) that correspond to different interpretations of the scene. This is an impressive result given that multi-stability is a well-studied, pervasive feature of human perception that is important for handling ambiguity, and that is not modelled by any standard image recognition networks.

7.3.7 Sequences

The iterative nature of IODINE lends itself readily to sequential data, by, eg., feeding a new frame at every iteration, instead of the same input image **x**. This setup corresponds to one iteration per timestep, and using next-step-prediction instead of reconstruction as part of the training objective. An example of this can be seen in Figure 7.19 where we show a 16 timestep sequence along with reconstructions and masks. When using the model in this way, it automatically maintains the association of object to slot over time (i.e, displaying robust slot stability). Thus, object tracking comes almost for free as a by-product in IODINE. Notice though, that IODINE has to rely on the LSTM that is part of the inference network to model any dynamics. That means none of the dynamics of tracked objects (eg. velocity) will be part of the object representation.

7.4 Discussion and Future Work

We have introduced IODINE, a novel approach for unsupervised representation learning of multi-object scenes, based on amortized iterative refinement of the inferred latent representation. We analyzed IODINE's performance on various datasets, including realistic images containing variable numbers of partially occluded 3D objects, and demonstrated that our method can successfully decompose the scenes into objects and represent each of them in terms of their individual properties such as color, size, and material. IODINE can robustly deal with occlusions by inpainting covered sections, and generalises beyond the training distribution in terms of numerosity and object-property combinations. Furthermore, when applied to scenes with ambiguity in terms of their object decomposition, IODINE can represent – and converge to – multiple valid solutions given the same input image.

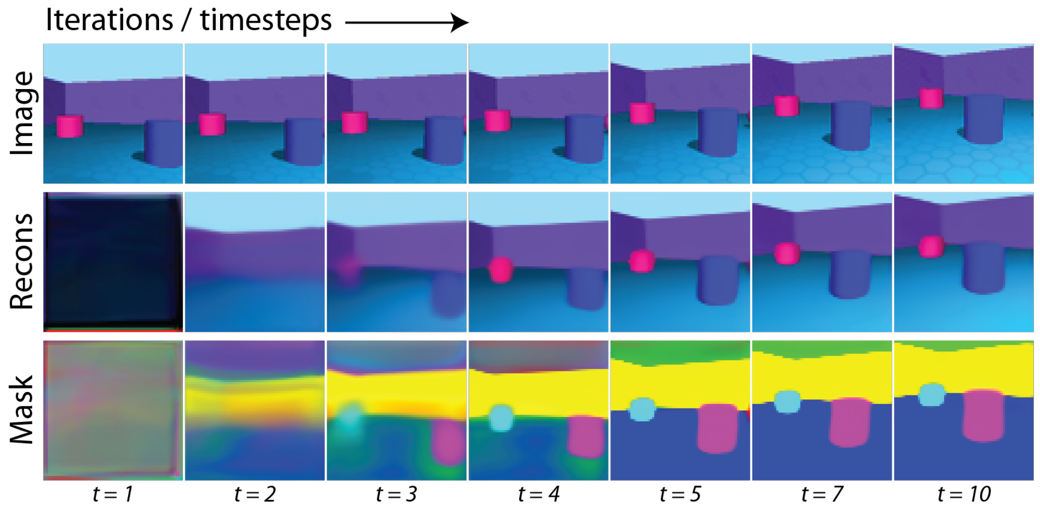


Figure 7.19. IODINE applied to Objects Room sequences by setting N, the number of refinement iterations, equal to the number of timesteps in the data.

7.4.1 Limitations

More Complex Data We also probed the limits of our current setup by applying IODINE to the Textured MNIST dataset [120] and to ImageNet, testing how it would deal with texture-segmentation and more complex real-world data (Figure 7.18). Trained on ImageNet data, IODINE segmented mostly by color rather than by objects. This behavior is not unexpected: ImageNet was never designed as a dataset for unsupervised learning, and likely lacks the richness in poses, lighting, sizes, positions and distance variations required to learn object segmentations from scratch. Trained on Textured MNIST, IODINE was able to model the background, but mostly failed to capture the foreground digits. Together these results point to the importance of color as a strong cue for segmentation, especially early in the iterative refinement process. As demonstrated by our results on grayscale CLEVR (Figure 7.18c) though, color is not a requirement.

Beyond more diverse training data, we want to highlight three other promising directions to scale IODINE to richer real-world data. First, an extension to sequential data is attractive, because temporal data naturally contains rich statistics about objectness both in the movement itself, and in the smooth variations of object factors. IODINE can readily be applied to sequences feeding a new frame at every iteration, and we have done some preliminary experiments described in Section 7.3.7. As a nice side-effect, the model automatically maintains the object to slot association, turning it into an unsupervised object tracker.

Physical interaction between objects is another common occurrence in sequential data. IODINE in its current form has limited abilities for modelling dynamics. Even statically placed objects commonly adhere to certain relations between each other, such as cars on streets. IODINE currently assumes objects to be placed independently of each other; relaxing this assumption will be important for modelling physical interactions. Yet there is also a need to balance this with the independence assumption required to split objects, since the system should still be able to segment out a car floating in space. Thus we believe integration with some form of graph network to support relations while preserving slot symmetry is another promising direction.

Finally, object representations have to be useful, such as for supervised tasks, or for agents in reinforcement learning setups. Whatever the task, it should provide important feedback about which objects matter and which are irrelevant. Complex visual scenes can contain an extremely large number of potential objects (think of sand grains on a beach), which can make it unfeasible to represent them all simultaneously. Allowing task-related signals to bias selection, for what and how to decompose, may enable scaling up unsupervised scene representation learning approaches like IODINE to arbitrarily complex scenes.

Memory It is worth pointing out that memory consumption presents an important limiting factor to scaling IODINE. To allow training by backpropagation, each slot and each refinement step require the storage of activations for an entire decoder and refinement network. Memory consumption during training thus scales linearly with both K and T. This is particularly restrictive for sequential data, where the number of steps can grow very large. In our experiments from Section 7.3.7, we found that 16 timesteps with a batch-size of 4 was the upper limit on GPUs with 12GB of RAM. Of course this also depends on the size of the input and the size of the network. Note also that at inference time there is no need to keep the activations of previous timesteps, so the dependence on T can be eliminated there.

Conceptual So far we've assumed the groups to represent independent objects or events. However, this assumption is unrealistic in many cases. Assuming only *conditional* independence would be considerably more reasonable, and could be implemented using a separate set of global latents that is used by all decoder slots. Alternatively, a relational mechanism could be added like in van Steenkiste et al. [385] to directly model interdependencies between objects.

The TAG framework assumes just one level of (global) groups, which does not reflect the hierarchical structure of the world. Therefore, another important future extension is to rather use a hierarchy of local groupings, by using our model as a component of a bigger system. This could be achieved by collapsing the groups of a Tagger network by summing them together at some hidden layer. That way this abstract representation could serve as input for another tagger with new groupings at this higher level. We hypothesize that a hierarchical Tagger could also represent relations between objects, because they are simply the couplings that remain from the assumption of independent objects."

Movement is a strong segmentation cue and a simple temporal extensions of the TAG framework could be to allow information to flow forward in time between higher layers, not just via the inputs. Iteration would then occur in time alongside the changing inputs. We believe that these extensions will make it possible to scale the approach to video.

Chapter 8

Conclusion

Humans naturally and effortlessly structure their perception and mental models in terms of discrete entities (objects). By decomposing the world in this way, humans are able to recombine their existing knowledge in a virtually unbounded number ways to understand unfamiliar situations, make novel inferences, or generate new behavior. This compositionality is at the heart of the impressive productivity and generalization ability of human reasoning, which reaches far beyond their direct experience. The ability to efficiently form, represent and relate modular object representations is thus critical to achieving human level generalization. In this work we have argued, that standard neural networks fail in this regard due to an inherent inability to dynamically and flexibly combine distributed information about one object, while simultaneously keeping it separate from the other information. This *binding problem*, we claim, is the underlying cause for many of the weaknesses of current neural networks, including their need for large amounts of labeled data, their lack of systematic generalization, and their fragility under distributional shift. We believe that addressing this problem is among the most important open challenges on the path to general artificial intelligence.

We have analyzed the binding problem of connectionism in terms of three aspects: Firstly, the ability to simultaneously represent multiple object representations in a common format, without interference between them (the representation problem). This aspect was first recognized in neuroscience, and has sparked decades of fascinating research and debate about potential mechanisms by which the brain might accomplish this. Fortunately, in the context of artificial neural networks, the space of possible solutions is less restricted, and we reviewed several promising approaches for addressing this problem. Secondly, the capacity to dynamically relate and compose modular object representations to build structured models for inference, prediction and behavior that generalize in predictable and systematic ways (composition problem). This aspect is strongly related to variable binding and the discussion of productivity and systematicity in linguistics. The expressiveness of compositional systems was recognized and leveraged already early on by symbol-based AI systems. Early symbol-based AI systems, already recognized and leveraged the expressiveness of compositional systems. However, their symbols had to be designed by humans, and the way in which they refer to and arise from reality still remains largely mysterious. The final aspect of the binding problem is thus the process of forming grounded object representations from raw unstructured inputs (segregation problem). Humans possess a remarkable ability to organize the unstructured stream of sensory information into meaningful entities, and even infants as young as four months old, have been shown to already posess at least a rudimentary understanding of objects. Working towards a solution to this problem in the context of neural networks has been the main focus of this thesis, and in our estimation, it is the most neglected and most important of the three aspects.

Perceptual grouping requires a proper treatment of the dynamic and hierarchical nature of objects. This implies a vast number of potentially useful objects, only a subset of which is relevant in any given situation. Furthermore, the most useful decomposition in terms of objects (and the associated level of abstraction) depends not only on the task, but also on the abstractions, relations, and general problem solving capabilities available to the entire system. Therefore, we believe that the ability to segregate must largely be learned in an unsupervised fashion, which is still often overlooked in the current literature.

In this thesis we have taken several steps towards this goal: We have proposed to understand "objects" from a functional (pragmatic) perspective, and argued that their defining quality of an object is that it is modular, i.e. it is self-contained and reusable independent of context. This information-based notion of objects is independent from any human supervision, and defined purely in terms of the (statistical) structure of the data and the predictive capacities of the model. This approach has allowed us to formalize the process of perceptual grouping as a form of clustering, using approximated mutual information as a distance metric (Chapter 4). Starting from an initial proof-of-concept, we were able to develop several frameworks for unsupervised perceptual grouping with neural networks based on generalized expectation maximization Chapter 5, denoising autoencoders Chapter 6, and iterative variational inference Chapter 7. Using simple synthetic datasets, we have demonstrated that it is possible to learn object representations and segmentation without supervision, that these representations help with prediction and semi-supervised tasks, and that they help with interpretability.

Perceptual grouping in neural networks is far from solved, and many important problems remain open. The main challenge is in scaling and improving methods to tackle real-world data, rather than the simplified synthetic datasets which we have used here. We expect that the extension from static images to video will be instrumental in this regard, because motion plays such a pivotal role in the perception of objects. Further challenging extensions of perceptual grouping include, the support of hierarchy, temporal grouping into events, other modalities such as audio or tactile, the support of abstract groupings towards categorization and concepts, and the integration with language models, to name but a few. Another open problem is to integrate segregation, representation, and composition into a single system in a way that resolves these dependencies (eg. via top-down feedback). Addressing these open problems may pave the way for an integrated system that can learn to dynamically construct structured models for prediction, inference, and behavior in a way that generalizes similarly to humans.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. [2016]. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, *arXiv:1603.04467 [cs]* .

[2] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. and Susstrunk, S. [2012]. SLIC superpixels compared to state-of-the-art superpixel methods, *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(11): 2274–2282.

[3] Ackley, D. H., Hinton, G. E. and Sejnowski, T. J. [1985]. A learning algorithm for Boltzmann machines, *Cogn. Sci.* **9**: 147–169.

[4] Alexe, B., Deselaers, T. and Ferrari, V. [2010]. What is an object?, *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On*, ieeexplore.ieee.org, pp. 73–80.

[5] Almeida, L. B. [1987]. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment, *Proceedings, 1st First International Conference on Neural Networks*, Vol. 2, ci.nii.ac.jp, pp. 609–618.

[6] Ancona, M., Ceolini, E., Öztireli, C. and Gross, M. [2017]. A unified view of gradient-based attribution methods for deep neural networks, *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning-Now What?(NIPS 2017)*, ETH Zurich.

[7] Andreas, J. [2019]. Measuring compositionality in representation learning, *arXiv preprint arXiv:1902.07181* .

[8] Arandjelović, R., Zisserman, A., Luo, Y., Hu, C., Lu, X. and Yu, X. [2019]. Object discovery with a copy-pasting GAN, *arXiv preprint arXiv:1905.11369* .

[9] Arbeláez, P., Maire, M., Fowlkes, C. C. and Malik, J. [2011]. Contour detection and hierarchical image segmentation, *IEEE transactions on pattern analysis and machine intelligence* **33**(5): 898–916.

[10] Attneave, F. [1971]. Multistability in perception, *Scientific American* **225**(6): 62–71.

[11] Atwood, J. and Towsley, D. [2016]. Diffusion-convolutional neural networks, *Advances in Neural Information Processing Systems*, pp. 1993–2001.

[12] Atzmon, Y., Berant, J., Kezami, V., Globerson, A. and Chechik, G. [2016]. Learning to generalize to new compositions in image understanding.

[13] Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z. and Ionescu, C. [2016]. Using fast weights to attend to the recent past, *in* D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett (eds), *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pp. 4331–4339.

[14] Ba, J., Kiros, J. R. and Hinton, G. E. [2016]. Layer normalization, *arXiv:1607. 06450 [cs, stat]* .

[15] Ba, J., Mnih, V. and Kavukcuoglu, K. [2014]. Multiple object recognition with visual attention.

[16] Bahdanau, D., Cho, K. and Bengio, Y. [2014]. Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473* .

[17] Bai, M. and Urtasun, R. [2017]. Deep watershed transform for instance segmentation, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5221–5229.

[18] BamBini, V., Chesi, C. and Moro, A. [2012]. A conversation with Noam Chomsky: New insights on old foundations, *Phenomenology and Mind* (3): 166–178.

[19] Barlow, H. B., Kaushal, T. P. and Mitchison, G. J. [1989]. Finding minimum entropy codes, *Neural Comput.* **1**(3): 412–423.

[20] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, H. F., Ballard, A., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y. and Pascanu, R. [2018]. Relational inductive biases, deep learning, and graph networks, *arXiv preprint arXiv:1806.01261* .

[21] Battaglia, P. W., Pascanu, R., Lai, M. and Rezende, D. J. [2016]. Interaction networks for learning about objects, relations and physics, *Advances in Neural Information Processing Systems*, pp. 4502–4510.

[22] Batty, C. [2014]. Olfactory objects, *Perception and its modalities* pp. 222–224.

[23] Bear, D., Fan, C., Mrowca, D., Li, Y., Alter, S., Nayebi, A., Schwartz, J., Fei-Fei, L. F., Wu, J., Tenenbaum, J. and Yamins, D. L. [2020]. Learning physical graph representations from visual scenes, *Advances in Neural Information Processing Systems* **33**.

[24] Beck, A. and Teboulle, M. [2009]. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring, *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference On*, IEEE, pp. 693–696.

[25] Becker, S. and Hinton, G. E. [1992]. Self-organizing neural network that discovers surfaces in random-dot stereograms, *Nature* **355**(6356): 161–163.

[26] Behnke, S. [1999]. Hebbian learning and competition in the neural abstraction pyramid, *Neural Networks, 1999. IJCNN'99. International Joint Conference On*, Vol. 2, IEEE, pp. 1356–1361.

[27] Behnke, S. [2001]. Learning iterative image reconstruction in the neural abstraction pyramid, *Int. J. Comput. Intell. Appl.* **1**(04): 427–438.

[28] Behnke, S. [2003]. Learning iterative binarization using hierarchical recurrent networks, *Cell* **2**: 2i.

[29] Bengio, Y., Courville, A. and Vincent, P. [2013]. Representation learning: A review and new perspectives, *IEEE transactions on pattern analysis and machine intelligence* **35**(8): 1798–1828.

[30] Bengio, Y., Laufer, E., Alain, G. and Yosinski, J. [2014]. Deep generative stochastic networks trainable by backprop, pp. 226–234.

[31] Bengio, Y., Yao, L., Alain, G. and Vincent, P. [2013]. Generalized denoising auto-encoders as generative models, *Advances in Neural Information Processing Systems*, pp. 899–907.

[32] Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S. and Hesse, C. [2019]. Dota 2 with large scale deep reinforcement learning, *arXiv preprint arXiv:1912.06680* .

[33] Bianchini, M. and Scarselli, F. [2014]. On the complexity of neural network classifiers: A comparison between shallow and deep architectures, *IEEE Trans Neural Netw Learn Syst* **25**(8): 1553–1565.

[34] Bishop, C. [2006]. *Pattern Recognition and Machine Learning*, Information Science and Statistics, Springer-Verlag, New York.

[35] Blekas, K., Likas, A., Galatsanos, N. P. and Lagaris, I. E. [2005]. A spatially constrained mixture model for image segmentation, *IEEE Trans. Neural Netw.* **16**(2): 494–498.

[36] Bobrow, D. G. [1964]. Natural language input for a computer problem solving system.

[37] Bowers, J. S., Vankov, I. I., Damian, M. F. and Davis, C. J. [2014]. Neural networks learn highly selective representations in order to overcome the superposition catastrophe, *Psychological Review* **121**(2): 248–261.

[38] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S. and Zhang, Q. [2018]. JAX: Composable transformations of Python+NumPy programs.

[39] Brendel, W. and Bethge, M. [2019]. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet, *arXiv preprint arXiv:1904.00760* .

[40] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. and Vandergheynst, P. [2016]. Geometric deep learning: Going beyond Euclidean data.

[41] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G. and Askell, A. [2020]. Language models are few-shot learners, *arXiv preprint arXiv:2005.14165* .

[42] Bruce, N. D. B. and Tsotsos, J. K. [2006]. Saliency based on information maximization, *in* Y. Weiss, B. Schölkopf and J. C. Platt (eds), *Advances in Neural Information Processing Systems 18*, MIT Press, pp. 155–162.

[43] Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M. and Lerchner, A. [2019]. MONet: Unsupervised scene decomposition and representation, *arXiv preprint arXiv:1901.11390* .

[44] Butko, N. J. and Movellan, J. R. [2009]. Optimal scanning for faster object detection, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2751–2758.

[45] Campbell, M., Hoane Jr, A. J. and Hsu, F.-h. [2002]. Deep blue, *Artificial intelligence* **134**(1-2): 57–83.

[46] Cantwell-Smith, B. [1998]. *On the Origin of Objects*, A Bradford Book, 1st paperback ed edn, MIT Press, Cambridge, Mass.

[47] Cao, C., Liu, X., Yang, Y., Yu, Y., Wang, J., Wang, Z., Huang, Y., Wang, L., Huang, C., Xu, W., Ramanan, D. and Huang, T. S. [2015]. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks, pp. 2956–2964.

[48] Caporale, N. and Dan, Y. [2008]. Spike timing–dependent plasticity: A Hebbian learning rule, *Annual Review of Neuroscience* **31**: 25–46.

[49] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. and Zagoruyko, S. [2020]. End-to-end object detection with transformers, *arXiv preprint arXiv:2005.12872* .

[50] Chang, M. B., Ullman, T., Torralba, A. and Tenenbaum, J. B. [2016]. A compositional object-based approach to learning physical dynamics, *arXiv preprint arXiv:1612.00341* .

[51] Chater, N. [1996]. Reconciling simplicity and likelihood principles in perceptual organization., *Psychological review* **103**(3): 566–581.

[52] Chen, C., Deng, F. and Ahn, S. [2020]. Object-centric representation and rendering of 3D scenes, *arXiv preprint arXiv:2006.06130* .

[53] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. and Abbeel, P. [2016]. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets, *arXiv:1606.03657 [cs, stat]* .

[54] Cherry, E. C. [1953]. 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.

[55] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M. and Schmidhuber, J. [2011]. Flexible, high performance convolutional neural networks for image classification, *International Joint Conference on Artificial Intelligence*, pp. 1237–1242.

[56] Cireşan, D. C., Meier, U., Masci, J. and Schmidhuber, J. [2012]. Multi-column deep neural network for traffic sign classification, *Neural Networks* **32**: 333–338.

[57] Clevert, D.-A., Unterthiner, T. and Hochreiter, S. [2015]. Fast and accurate deep network learning by exponential linear units (ELUs).

[58] Crawford, E. and Pineau, J. [2019]. Spatially invariant unsupervised object detection with convolutional neural networks, *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 3412–3420.

[59] Csordás, R. and Schmidhuber, J. [2019]. Improved addressing in the differentiable neural computer.

[60] Csordás, R., van Steenkiste, S. and Schmidhuber, J. [2020]. Are neural nets modular? Inspecting functional modularity through differentiable weight masks, *arxiv preprint arXiv:2010.02066* .

[61] Cucchiara, R., Grana, C., Piccardi, M. and Prati, A. [2003]. Detecting moving objects, ghosts, and shadows in video streams, *IEEE transactions on pattern analysis and machine intelligence* **25**(10): 1337–1342.

[62] Dai, J., He, K. and Sun, J. [2016]. Instance-aware semantic segmentation via multi-task network cascades, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3150–3158.

[63] Das, S., Giles, C. L. and Sun, G.-Z. [1992]. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory, *Proceedings of The Fourteenth Annual Conference of Cognitive Science Society. Indiana University*, p. 14.

[64] Daubechies, I., Defrise, M. and De Mol, C. [2004]. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Commun. Pure Appl. Math.* **57**(11): 1413–1457.

[65] Deco, G. [2001]. Biased competition mechanisms for visual attention in a multimodular neurodynamical system, *Emergent Neural Computational Architectures Based on Neuroscience*, Springer, pp. 114–126.

[66] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J. and Kaiser, Ł. [2019]. Universal transformers, *International Conference on Learning Representations*.

[67] Del Pero, L., Bowdish, J., Fried, D., Kermgard, B., Hartley, E. and Barnard, K. [2012]. Bayesian geometric modeling of indoor scenes, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, ieeexplore.ieee.org, pp. 2719–2726.

[68] Del Pero, L., Bowdish, J., Kermgard, B., Hartley, E. and Barnard, K. [2013]. Understanding Bayesian rooms using composite 3d object models, *IEEE Conference on Computer Vision and Pattern Recognition*, cv-foundation.org, pp. 153–160.

[69] Dempster, A. P., Laird, N. M. and Rubin, D. B. [1977]. Maximum likelihood from incomplete data via the EM algorithm, *J. R. Stat. Soc. Series B Stat. Methodol.* pp. 1–38.

[70] Deng, Z., Nawhal, M., Meng, L. and Mori, G. [2019]. Continuous graph flow, *arXiv preprint arXiv:1908.02436* .

[71] Devereux, B. J., Tyler, L. K., Geertzen, J. and Randall, B. [2014]. The centre for speech, language and the brain (CSLB) concept property norms, *Behavior Research Methods* **46**(4): 1119–1127.

[72] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. [2019]. BERT: Pre-training of deep bidirectional transformers for language understanding, *arXiv:1810.04805 [cs]* .

[73] Doetsch, P., Kozielski, M. and Ney, H. [2014]. Fast and robust training of recurrent neural networks for offline handwriting recognition, *14th International Conference on Frontiers in Handwriting Recognition*.

[74] Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T. [2014]. Long-term recurrent convolutional networks for visual recognition and description, *arXiv:1411. 4389 [cs]* .

[75] Doumas, L. A. A., Hummel, J. E. and Sandhofer, C. M. [2008]. A theory of the discovery and predication of relational concepts., *Psychological Review* **115**(1): 1–43.

[76] Doumas, L. A. A., Puebla, G., Martin, A. E. and Hummel, J. E. [2019]. Relation learning in a neurocomputational architecture supports cross-domain transfer, *arXiv preprint arXiv:1910.05065* .

[77] Dreyfus, S. [1973]. The computational solution of optimal control problems with time lag, *IEEE Transactions on Automatic Control* **18**(4): 383–385.

[78] Duchi, J., Hazan, E. and Singer, Y. [2011]. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, p. 39.

[79] Eastwood, C. and Williams, C. K. I. [2018]. A framework for the quantitative evaluation of disentangled representations.

[80] Ehrhardt, S., Groth, O., Monszpart, A., Engelcke, M., Posner, I., Mitra, N. and Vedaldi, A. [2020]. RELATE: Physically plausible multi-object scene synthesis using structured latent spaces, *Advances in Neural Information Processing Systems* **33**.

[81] Elman, J. L. [1988]. Finding structure in time, *Cogn. Sci.* **211**(CRL Technical Report 8801): 1–28.

[82] Endres, I. and Hoiem, D. [2010]. Category independent object proposals, *European Conference on Computer Vision*, pp. 575–588.

[83] Engelcke, M., Kosiorek, A. R., Jones, O. P. and Posner, I. [2019]. Genesis: Generative scene inference and sampling with object-centric latent representations, *arXiv preprint arXiv:1907.13052* .

[84] Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K. and Hinton, G. E. [2016]. Attend, infer, repeat: Fast scene understanding with generative models, *Advances In Neural Information Processing Systems*, pp. 3225–3233.

[85] Fan, Y., Qian, Y., Xie, F. and Soong, F. K. [2014]. TTS synthesis with bidirectional LSTM based recurrent neural networks, *Proc. Interspeech*.

[86] Feldman, J. [2013]. The neural binding problem(s), *Cognitive neurodynamics* **7**(1): 1–11.

[87] Felzenszwalb, P. F. and Huttenlocher, D. P. [2004]. Efficient graph-based image segmentation, *Int. J. Comput. Vis.* **59**(2): 167–181.

[88] Fodor, J. A. [1975]. *The Language of Thought*, Vol. 5, Harvard university press.

[89] Fodor, J. A. and Pylyshyn, Z. W. [1988]. Connectionism and cognitive architecture: A critical analysis, *Cognition* **28**(1-2): 3–71.

[90] Földiák, P. [1990]. Forming sparse representations by local anti-Hebbian learning, *Biol. Cybern.* **64**(2): 165–170.

[91] Friedman, N. and Russell, S. [1997]. Image segmentation in video sequences: A probabilistic approach, *UAI'97*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 175–181.

[92] Fuchs, F. B., Kosiorek, A. R., Sun, L., Jones, O. P. and Posner, I. [2019]. End-to-end recurrent multi-object tracking and trajectory prediction with relational reasoning, *arXiv preprint arXiv:1907.12887* .

[93] Fukuda, K., Awh, E. and Vogel, E. K. [2010]. Discrete capacity limits in visual working memory, *Current opinion in neurobiology* **20**(2): 177–182.

[94] Gallinari, P., LeCun, Y., Thiria, S. and Fogelman-Soulie, F. [1987]. Mémoires associatives distribuées: Une comparaison (Distributed associative memories: A comparison), *Cesta-Afcet*.

[95] Gamrian, S. and Goldberg, Y. [2019]. Transfer learning for related reinforcement learning tasks via image-to-image translation, *arXiv:1806.07377 [cs]* .

[96] Gao, D. and Vasconcelos, N. [2005]. Discriminant saliency for visual recognition from cluttered scenes, *in* L. K. Saul, Y. Weiss and L. Bottou (eds), *Advances in Neural Information Processing Systems 17*, MIT Press, pp. 481–488.

[97] Gauthier, I. and Tarr, M. J. [1997]. Becoming a "Greeble" expert: Exploring mechanisms for face recognition, *Vision Res.* **37**(12): 1673–1682.

[98] Gayler, R. W. [1998]. Multiplicative binding, representation operators & analogy, *Cogprints* .

[99] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A. and Brendel, W. [2018]. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness, *arXiv preprint arXiv:1811.12231* .

[100] Gers, F. A. and Schmidhuber, J. [2000]. Recurrent nets that time and count, *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference On*, Vol. 3, IEEE, pp. 189–194.

[101] Gers, F. A., Schmidhuber, J. and Cummins, F. [1999]. Learning to forget: Continual prediction with LSTM, *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, Vol. 2, pp. 850–855.

[102] Giles, C. L., Sun, G.-Z., Chen, H.-H., Lee, Y.-C. and Chen, D. [1990]. Higher order recurrent networks and grammatical inference, *Advances in Neural Information Processing Systems*, pp. 380–387.

[103] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. and Dahl, G. E. [2017]. Neural message passing for quantum chemistry, *International Conference on Machine Learning*, Vol. 70, International Convention Centre, Sydney, Australia, pp. 1263–1272.

[104] Girshick, R. B. [2015]. Fast R-CNN, *CoRR* **abs/1504.08083**.

[105] Girshick, R. B., Donahue, J., Darrell, T. and Malik, J. [2014]. Rich feature hierarchies for accurate object detection and semantic segmentation, *IEEE Conference on Computer Vision and Pattern Recognition*, cv-foundation.org, pp. 580–587.

[106] Glauber, R. J. [1963]. Time-Dependent Statistics of the Ising Model, *Journal of Mathematical Physics* **4**(2): 294–307.

[107] Goodfellow, I., Bengio, Y. and Courville, A. [2016]. *Deep Learning*, MIT press.

[108] Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S. and Shet, V. [2013]. Multi-digit number recognition from street view imagery using deep convolutional neural networks, *arXiv preprint arXiv:1312.6082* .

[109] Gopalakrishnan, A., van Steenkiste, S. and Schmidhuber, J. [2020]. Unsupervised object keypoint learning using local spatial predictability, *arXiv preprint arXiv:2011.12930* .

[110] Gori, M., Monfardini, G. and Scarselli, F. [2005]. A new model for learning in graph domains, *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2, ieeexplore.ieee.org, pp. 729–734 vol. 2.

[111] Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y. and Schölkopf, B. [2019]. Recurrent independent mechanisms, *arXiv preprint arXiv:1909.10893* .

[112] Graves, A. [2013]. Generating sequences with recurrent neural networks, *arXiv preprint arXiv:1308. 0850* .

[113] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. and Schmidhuber, J. [2009]. A novel connectionist system for unconstrained handwriting recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(5): 855–868.

[114] Graves, A., Mohamed, A. and Hinton, G. E. [2013]. Speech recognition with deep recurrent neural networks, *ICASSP*, pp. 6645–6649.

[115] Graves, A. and Schmidhuber, J. [2005]. Framewise phoneme classification with bidirectional LSTM networks, *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference On*, Vol. 4, IEEE, pp. 2047–2052.

[116] Graves, A., Wayne, G. and Danihelka, I. [2014]. Neural Turing machines, *arXiv preprint arXiv:1410.5401* .

[117] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K. and Hassabis, D. [2016]. Hybrid computing using a neural network with dynamic external memory, *Nature* **538**(7626): 471–476.

[118] Green, E. J. [2018]. A theory of perceptual objects, *Philos. Phenomenol. Res.* **106**: 7345.

[119] Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C. P., Zoran, D., Matthey, L., Botvinick, M. and Lerchner, A. [2019]. Multi-object representation learning with iterative variational inference, *ICML*.

[120] Greff, K., Rasmus, A., Berglund, M., Hao, T. H., Valpola, H. and Schmidhuber, J. [2016]. Tagger: Deep unsupervised perceptual grouping, *in* D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett (eds), *NeurIPS*, pp. 4484–4492.

[121] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. and Schmidhuber, J. [2015]. LSTM: A search space odyssey, *IEEE TNNLS* **28**(10): 2222–2232.

[122] Greff, K., Srivastava, R. K. and Schmidhuber, J. [2015]. Binding via reconstruction clustering, *arXiv:1511.06418 [cs]* .

[123] Greff, K., van Steenkiste, S. and Schmidhuber, J. [2017]. Neural expectation maximization, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (eds), *NeurIPS*, Curran Associates, Inc., pp. 6694–6704.

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

[125] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J. and Wierstra, D. [2015]. DRAW: A recurrent neural network for image generation, *International Conference on Machine Learning*, Lille, France, pp. 1462–1471.

[126] Gregor, K. and LeCun, Y. [2010]. Learning fast approximations of sparse coding, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 399–406.

[127] Grunwald, P. and Vitanyi, P. [2004]. Shannon Information and Kolmogorov Complexity, *arXiv:cs/0410002* .

[128] Guerrero-Colón, J. A., Simoncelli, E. P. and Portilla, J. [2008]. Image denoising using mixtures of Gaussian scale mixtures, *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference On*, IEEE, pp. 565–568.

[129] Håastad, J. [1987]. Computational limitations of small-depth circuits.

[130] Hamilton, W., Ying, Z. and Leskovec, J. [2017]. Inductive representation learning on large graphs, *Advances in Neural Information Processing Systems*, pp. 1024–1034.

[131] Hamrick, J. B., Allen, K. R., Bapst, V., Zhu, T., McKee, K. R., Tenenbaum, J. B. and Battaglia, P. W. [2018]. Relational inductive bias for physical construction in humans and machines, *arXiv preprint arXiv:1806.01203* .

[132] Harnad, S. [1990]. The symbol grounding problem, *Physica D* **42**(1): 335–346.

[133] Harzallah, H., Jurie, F. and Schmid, C. [2009]. Combining efficient object localization and image classification, *IEEE International Conference on Computer Vision*, pp. 237–244.

[134] Hatfield, G. and Epstein, W. [1985]. The status of the minimum principle in the theoretical analysis of visual perception, *Psychological Bulletin* **97**(2): 155–186.

[135] Hayes, S. C., Barnes-Holmes, D. and Roche, B. (eds) [2001]. *Relational Frame Theory: A Post-Skinnerian Account of Human Language and Cognition*, Springer US.

[136] He, K., Gkioxari, G., Dollár, P. and Girshick, R. [2017]. Mask R-CNN, *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969.

[137] Henaff, M., Bruna, J. and LeCun, Y. [2015]. Deep convolutional networks on graph-structured data, *arXiv preprint arXiv:1506.05163* .

[138] Henaff, M., Weston, J., Szlam, A., Bordes, A. and LeCun, Y. [2016]. Tracking the world state with recurrent entity networks.

[139] Hershey, J. R., Roux, J. L. and Weninger, F. [2014]. Deep unfolding: Model-based inspiration of novel deep architectures, *arXiv preprint arXiv:1409.2574* .

[140] Higgins, I., Amos, D., Pfau, D., Racanière, S., Matthey, L., Rezende, D. J. and Lerchner, A. [2018]. Towards a definition of disentangled representations, *arXiv preprint arXiv:1812.02230* .

[141] Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M., Mohamed, S. and Lerchner, A. [2017]. Beta-VAE: Learning basic visual concepts with a constrained variational framework, *In Proceedings of the International Conference on Learning Representations (ICLR)*.

[142] Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C. and Lerchner, A. [2017]. DARLA: Improving zero-shot transfer in reinforcement learning, *ICML* .

[143] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P. and Sainath, T. N. [2012]. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal processing magazine* **29**(6): 82–97.

[144] Hinton, G. E. [1984]. Distributed representations, *Technical report*.

[145] Hinton, G. E., Krizhevsky, A. and Wang, S. D. [2011]. Transforming auto-encoders, *International Conference on Artificial Neural Networks*, Springer Berlin Heidelberg, pp. 44–51.

[146] Hinton, G. E., Sabour, S. and Frosst, N. [2018]. Matrix capsules with EM routing.

[147] Hochberg, J. and McAlister, E. [1953]. A quantitative approach to figural "goodness", *Journal of Experimental Psychology* **46**(5): 361–364.

[148] Hochreiter, S. [1991]. *Untersuchungen zu dynamischen neuronalen Netzen*, Masters Thesis, Technische Universität München, München.

[149] Hochreiter, S. and Schmidhuber, J. [1995]. Long short term memory, *Technical report*, Technische Universität München, München.

[150] Hochreiter, S. and Schmidhuber, J. [1997]. Long short-term memory, *Neural Comput.* **9**(8): 1735–1780.

[151] Hochreiter, S., Younger, A. S. and Conwell, P. R. [2001]. Learning to learn using gradient descent, *Proc. International Conference on Artificial Neural Networks*, Springer, pp. 87–94.

[152] Hoiem, D., Efros, A. A. and Hebert, M. [2011]. Recovering occlusion boundaries from an image, *International Journal of Computer Vision* **91**(3): 328–346.

[153] Hopfield, J. J. [1982]. Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the national academy of sciences* **79**(8): 2554–2558.

[154] Hornik, K., Stinchcombe, M., White, H. et al. [1989]. Multilayer feedforward networks are universal approximators., *Neural networks* **2**(5): 359–366.

[155] Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J. and Schölkopf, B. [2009]. Nonlinear causal discovery with additive noise models, *Advances in Neural Information Processing Systems*, pp. 689–696.

[156] Huang, J. and Murphy, K. [2015]. Efficient inference in occlusion-aware generative models of images, *arXiv preprint arXiv:1511.06362* .

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

[158] Hudson, D. A. and Manning, C. D. [2018]. Compositional attention networks for machine reasoning, *arXiv [cs.AI]* .

[159] Hughes, S. and Barnes-Holmes, D. [2016]. Relational frame theory: The basic account, *The Wiley Handbook of Contextual Behavioral Science* p. 129.

[160] Hummel, J. E. and Holyoak, K. J. [1993]. Distributing structure over time, *Behavioral and Brain Sciences* **16**(3): 464–464.

[161] Hummel, J. E., Holyoak, K. J., Green, C., Doumas, L. A. A., Devnich, D., Kittur, A. and Kalar, D. J. [2004]. A solution to the binding problem for compositional connectionism, *Compositional Connectionism in Cognitive Science: Papers from the AAAI Fall Symposium, Ed. SD Levy & R. Gayler*, vvvvw.aaai.org, pp. 31–34.

[162] Hupkes, D., Dankers, V., Mul, M. and Bruni, E. [2019]. The compositionality of neural networks: Integrating symbolism and connectionism, *arXiv preprint arXiv:1908.08351* .

[163] Hyvärinen, A. and Oja, E. [2000]. Independent component analysis: Algorithms and applications, *Neural networks* **13**(4-5): 411–430.

[164] Hyvärinen, A. and Perkiö, J. [2006]. Learning to segment any random vector, *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, IEEE, pp. 4167–4172.

[165] Ioffe, S. and Szegedy, C. [2015]. Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint arXiv:1502.03167* .

[166] Isola, P., Zoran, D., Krishnan, D. and Adelson, E. H. [2014]. Crisp boundary detection using pointwise mutual information, *European Conference on Computer Vision*, pp. 799–814.

[167] Isola, P., Zoran, D., Krishnan, D. and Adelson, E. H. [2015]. Learning visual groups from co-occurrences in space and time, *arXiv:1511. 06811 [cs]* .

[168] Itti, L., Koch, C. and Niebur, E. [1998]. A model of saliency-based visual attention for rapid scene analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* **20**(11): 1254–1259.

[169] Iuzzolino, M., Singer, Y. and Mozer, M. C. [2019]. Convolutional bipartite attractor networks, *arXiv preprint arXiv:1906.03504* .

[170] Ivakhnenko, A. G. [1968]. The group method of data of handling; a rival of the method of stochastic approximation, *Soviet Automatic Control* **13**: 43–55.

[171] Ivakhnenko, A. and Valentin, G. [1966]. Cybernetic predicting devices, *Technical report*.

[172] Jaderberg, M., Simonyan, K., Zisserman, A. and Kavukcuoglu, K. [2015]. Spatial transformer networks, *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 2017–2025.

[173] Jain, A. K., Dubes, R. C. et al. [1988]. *Algorithms for Clustering Data*, Vol. 6, Prentice hall Englewood Cliffs, NJ.

[174] Jakab, T., Gupta, A., Bilen, H. and Vedaldi, A. [2018]. Unsupervised learning of object landmarks through conditional image generation, *Advances in Neural Information Processing Systems*, pp. 4016–4027.

[175] Janner, M., Levine, S., Freeman, W. T., Tenenbaum, J. B., Finn, C. and Wu, J. [2018]. Reasoning about physical interactions with object-oriented prediction and planning.

[176] Jepson, A. D. and Black, M. J. [1993]. Mixture models for optical flow computation, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 760–761.

[177] Ji, X., Henriques, J. F. and Vedaldi, A. [2018]. Invariant information distillation for unsupervised image segmentation and clustering.

[178] Jiang, J. and Ahn, S. [2020]. Generative Neurosymbolic Machines, *Advances in Neural Information Processing Systems*, Vol. 33.

[179] Jiang, J., Janghorbani, S., de Melo, G. and Ahn, S. [2020]. SCALOR: Generative world models with scalable object representations, p. 22.

[180] Jo, J. and Bengio, Y. [2017]. Measuring the tendency of CNNs to learn surface statistical regularities.

[181] Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L. and Girshick, R. B. [2017]. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning, *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference On*, openaccess.thecvf.com, pp. 1988–1997.

[182] Johnson-Laird, P. N. [2010]. Mental models and human reasoning, *Proceedings of the National Academy of Sciences* **107**(43): 18243–18250.

[183] Jojic, N. and Frey, B. J. [2001]. Learning flexible sprites in video layers, *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference On*, Vol. 1, IEEE, pp. I–I.

[184] Jordan, M. I. and Jacobs, R. A. [1990]. Learning to control an unstable system with forward modeling, *Advances in Neural Information Processing Systems*, pp. 324–331.

[185] Joulin, A. and Mikolov, T. [2015]. Inferring algorithmic patterns with stack-augmented recurrent nets, *Advances in Neural Information Processing Systems*, pp. 190–198.

[186] Kaiser, \. and Sutskever, I. [2015]. Neural gpus learn algorithms, *arXiv preprint arXiv:1511.08228* .

[187] Kanerva, P. [1996]. Binary spatter-coding of ordered K-tuples, *Artificial Neural Networks — ICANN 96*, Springer Berlin Heidelberg, pp. 869–873.

[188] Kannan, A., Winn, J. and Rother, C. [2006]. Clustering appearance and shape by learning jigsaws, *Advances in Neural Information Processing Systems*, pp. 657–664.

[189] Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, D. S. and George, D. [2017]. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics, *International Conference on Machine Learning*, Sydney, NSW, Australia, pp. 1809–1818.

[190] Kappers, A. M. L. and Tiest, W. M. B. [2015]. Tactile and haptic perceptual organization, *The Oxford handbook of perceptual organization* pp. 621–638.

[191] Karpathy, A., Johnson, J. and Fei-Fei, L. [2015]. Visualizing and understanding recurrent networks, *arXiv:1506.02078 [cs]* .

[192] Kelley, H. J. [1960]. Gradient theory of optimal flight paths, *Ars Journal* **30**(10): 947–954.

[193] Kelly, M. A., Blostein, D. and Mewhort, D. J. K. [2013]. Encoding structure in holographic reduced representations, *Can. J. Exp. Psychol.* **67**(2): 79–93.

[194] Kemp, C. and Tenenbaum, J. B. [2008]. The discovery of structural form, *Proceedings of the National Academy of Sciences* **105**(31): 10687–10692.

[195] Kempter, R., Gerstner, W. and Van Hemmen, J. L. [1999]. Hebbian learning and spiking neurons, *Physical Review E* **59**(4): 4498.

[196] Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M. and Bousquet, O. [2020]. Measuring compositional generalization: A comprehensive method on realistic data, p. 38.

[197] Kim, H. and Mnih, A. [2018]. Disentangling by factorising.

[198] Kingma, D. P. and Ba, J. [2015]. Adam: A method for stochastic optimization, CBLS.

[199] Kingma, D. P. and Welling, M. [2013]. Auto-encoding variational Bayes, *arXiv preprint arXiv:1312.6114* .

[200] Kipf, T., Fetaya, E., Wang, K.-C., Welling, M. and Zemel, R. [2018]. Neural relational inference for interacting systems, *arXiv preprint arXiv:1802.04687* .

[201] Kipf, T. N. and Welling, M. [2016]. Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* .

[202] Koffka, K. [1935]. *Principles of Gestalt Psychology*, Vol. 44, Routledge.

[203] Kohonen, T. [1989]. *Self-Organization and Associative Memory*, third edn, Springer.

[204] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S. and Houlsby, N. [2020]. Big Transfer (BiT): General Visual Representation Learning, *in* A. Vedaldi, H. Bischof, T. Brox and J.-M. Frahm (eds), *Computer Vision – ECCV 2020*, Vol. 12350, Springer International Publishing, Cham, pp. 491–507.

[205] Kolmogorov, A. N. [1965]. Three approaches to the quantitative definition ofinformation', *Problems of information transmission* **1**(1): 1–7.

[206] Kong, S. and Fowlkes, C. C. [2018]. Recurrent pixel embedding for instance grouping, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9018–9028.

[207] Korzybski, A. [1958]. *Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics*, Institute of GS.

[208] Kosiorek, A., Bewley, A. and Posner, I. [2017]. Hierarchical attentive recurrent tracking, *Advances in Neural Information Processing Systems*, pp. 3053–3061.

[209] Kosiorek, A., Kim, H., Teh, Y. W. and Posner, I. [2018]. Sequential attend, infer, repeat: Generative modelling of moving objects, *Advances in Neural Information Processing Systems*, pp. 8606–8616.

[210] Kosiorek, A., Sabour, S., Teh, Y. W. and Hinton, G. E. [2019]. Stacked capsule autoencoders, *Advances in Neural Information Processing Systems*, pp. 15486–15496.

[211] Krizhevsky, A., Sutskever, I. and Hinton, G. E. [2012]. ImageNet classification with deep convolutional neural networks, *in* F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 1097–1105.

[212] Kulkarni, T. D., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A. and Mnih, V. [2019]. Unsupervised learning of object keypoints for perception and control, *Advances in Neural Information Processing Systems*, pp. 10723–10733.

[213] Kulkarni, T. D., Kohli, P., Tenenbaum, J. B. and Mansinghka, V. K. [2015]. Picture: A probabilistic programming language for scene perception, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4390–4399.

[214] Kurach, K., Andrychowicz, M. and Sutskever, I. [2016]. Neural random-access machines, *International Conference on Learning Representations*.

[215] Kusner, M. J., Paige, B. and Hernández-Lobato, J. M. [2017]. Grammar variational autoencoder, *arXiv:1703.01925 [stat]*.

[216] Lake, B. and Baroni, M. [2018]. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks, *International Conference on Machine Learning*, pp. 2873–2882.

[217] Lake, B. M. and Baroni, M. [2017]. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks.

[218] Lake, B. M., Salakhutdinov, R. R. and Tenenbaum, J. B. [2015]. Human-level concept learning through probabilistic program induction, *Science* **350**(6266): 1332–1338.

[219] Lake, B. M., Ullman, T. D., Tenenbaum, J. B. and Gershman, S. J. [2017]. Building machines that learn and think like people, *Behavioral and brain sciences* **40**: e253.

[220] Lampert, C. H., Blaschko, M. B. and Hofmann, T. [2008]. Beyond sliding windows: Object localization by efficient subwindow search, *2008 IEEE Conference on Computer Vision and Pattern Recognition*, ieeexplore.ieee.org, pp. 1–8.

[221] Lample, G. and Charton, F. [2019]. Deep learning for symbolic mathematics, *arXiv preprint arXiv:1912.01412* .

[222] Lang, K. J., Waibel, A. H. and Hinton, G. E. [1990]. A time-delay neural network architecture for isolated word recognition, *Neural Networks* **3**(1): 23–43.

[223] Le Roux, N., Heess, N., Shotton, J. and Winn, J. [2011]. Learning a generative model of images by factoring appearance and shape, *Neural Comput.* **23**(3): 593–650.

[224] LeCun, Y. [1987]. *Modèles Connexionnistes de l'apprentissage*, Ph.D. Thesis, Paris 6.

[225] Lee, J., Lee, I. and Kang, J. [2019]. Self-attention graph pooling, *arXiv:1904.08082 [cs, stat]* .

[226] Lee, T.-W. and Lewicki, M. S. [2002]. Unsupervised image classification, segmentation, and enhancement using ICA mixture models, *IEEE Transactions on Image Processing* **11**(3): 270–279.

[227] Li, Y., Lin, T., Yi, K., Bear, D., Yamins, D., Wu, J., Tenenbaum, J. and Torralba, A. [2020]. Visual grounding of learned physical models, *International Conference on Machine Learning*, pp. 5927–5936.

[228] Li, Y., Tarlow, D., Brockschmidt, M. and Zemel, R. [2016]. Gated graph sequence neural networks, *In Proceedings of the International Conference on Learning Representations (ICLR)*.

[229] Liao, Q. and Poggio, T. [2016]. Bridging the gaps between residual learning, recurrent neural networks and visual cortex, *arXiv:1604.03640 [cs]* .

[230] Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R. and Zemel, R. [2019]. Efficient graph generation with graph recurrent attention networks, *Advances in Neural Information Processing Systems*, pp. 4257–4267.

[231] Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J. and Ahn, S. [2020]. SPACE: Unsupervised object-oriented scene representation via spatial attention and decomposition, *arXiv:2001.02407 [cs, eess, stat]* .

[232] Linnainmaa, S. [1970]. *The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors*, PhD thesis, Univ. Helsinki.

[233] Litany, O., Bronstein, A., Bronstein, M. and Makadia, A. [2018]. Deformable shape completion with graph convolutional autoencoders, *IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 1886–1895.

[234] Liu, J., Kumar, A., Ba, J., Kiros, J. and Swersky, K. [2019]. Graph normalizing flows, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox and R. Garnett (eds), *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 13578–13588.

[235] Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B. and Bachem, O. [2018]. Challenging common assumptions in the unsupervised learning of disentangled representations.

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

[237] Lopez-Paz, D., Muandet, K., Schölkopf, B. and Tolstikhin, I. [2015]. Towards a learning theory of cause-effect inference, *International Conference on Machine Learning*, pp. 1452–1461.

[238] Loula, J., Baroni, M. and Lake, B. M. [2018]. Rearranging the familiar: Testing compositional generalization in recurrent networks, *arXiv preprint arXiv:1807.07545* .

[239] Lun, Z., Zou, C., Huang, H., Kalogerakis, E., Tan, P., Cani, M.-P. and Zhang, H. [2017]. Learning to group discrete graphical patterns, *ACM Trans. Graph.* **36**(6): 225:1–225:11.

[240] Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O. and Zaremba, W. [2014]. Addressing the rare word problem in neural machine translation, *arXiv preprint arXiv:1410. 8206* .

[241] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B. J. [2015]. Adversarial autoencoders.

[242] Malik, J., Belongie, S., Leung, T. and Shi, J. [2001]. Contour and texture analysis for image segmentation, *International journal of computer vision* **43**(1): 7–27.

[243] Mansinghka, V. K., Kulkarni, T. D., Perov, Y. N. and Tenenbaum, J. B. [2013]. Approximate Bayesian image interpretation using generative probabilistic graphics programs, *in* C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pp. 1520–1528.

[244] Marchi, E., Ferroni, G., Eyben, F., Gabrielli, L., Squartini, S. and Schuller, B. [2014]. Multi-resolution linear prediction based features for audio onset detection with bidirectional LSTM neural networks, *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2164–2168.

[245] Marcus, G. F. [2003]. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*, MIT press.

[246] Marino, J., Cvitkovic, M. and Yue, Y. [2018]. A general method for amortizing variational filtering, *in* S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett (eds), *Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., pp. 7868–7879.

[247] Marino, J., Yue, Y. and Mandt, S. [2018]. Iterative amortized inference, *in* J. Dy and A. Krause (eds), *Proceedings of Machine Learning Research*, Vol. 80, PMLR, Stockholmsmässan, Stockholm Sweden, pp. 3403–3412.

[248] Martin, D. R., Fowlkes, C. C. and Malik, J. [2004]. Learning to detect natural image boundaries using local brightness, color, and texture cues, *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(5): 530–549.

[249] Matthey, L., Higgins, I., Hassabis, D. and Lerchner, A. [2017]. dSprites: Disentanglement testing sprites dataset, https://github.com/deepmind/dsprites-dataset/.

[250] McCulloch, W. S. and Pitts, W. [1943]. A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics* **5**(4): 115–133.

[251] Mcgurk, H. and Macdonald, J. [1976]. Hearing lips and seeing voices, *Nature* **264**(5588): 746.

[252] McMahan, H. B. and Streeter, M. [2010]. Adaptive Bound Optimization for Online Convex Optimization, *arXiv:1002.4908 [cs]* .

[253] Meier, M., Haschke, R. and Ritter, H. J. [2014]. Perceptual grouping through competition in coupled oscillator networks, *Neurocomputing* **141**: 76–83.

[254] Michotte, A., Thinès, G. and Crabbé, G. [1991]. Amodal completion of perceptual structures, *Michotte's experimental phenomenology of perception* pp. 140–167.

[255] Mikolov, T., Chen, K., Corrado, G. and Dean, J. [2013]. Efficient estimation of word representations in vector space, *arXiv:1301.3781 [cs]* .

[256] Miller, G. A. [1956]. The magical number seven, plus or minus two: Some limits on our capacity for processing information., *Psychological review* **63**(2): 81.

[257] Milner, P. M. [1974]. A model for visual shape recognition, *Psychol. Rev.* **81**(6): 521.

[258] Minderer, M., Sun, C., Villegas, R., Cole, F., Murphy, K. P. and Lee, H. [2019]. Unsupervised learning of object structure and dynamics from videos, *Advances in Neural Information Processing Systems*, pp. 92–102.

[259] Mitchell, T. M. [1997]. *Machine Learning*, McGraw-Hill Series in Computer Science, international ed., [reprint.] edn, McGraw-Hill, New York, NY.

[260] Mnih, V., Heess, N., Graves, A. and Kavukcuoglu, K. [2014]. Recurrent models of visual attention, *Advances in Neural Information Processing Systems*, Vol. 27, pp. 2204–2212.

[261] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K. and Ostrovski, G. [2015]. Human-level control through deep reinforcement learning, *Nature* **518**(7540): 529–533.

[262] Mo, S., Cho, M. and Shin, J. [2018]. Instagan: Instance-aware image-to-image translation, *arXiv preprint arXiv:1812.10889* .

[263] Mobahi, H., Rao, S. R., Yang, A. Y., Sastry, S. S. and Ma, Y. [2011]. Segmentation of natural images by texture and boundary compression, *International journal of computer vision* **95**(1): 86–98.

[264] Mordatch, I. [2019]. Concept learning with energy-based models, *in* A. K. Goel, C. M. Seifert and C. Freksa (eds), *Proceedings of the 41th Annual Meeting of the Cognitive Science Society, CogSci 2019: Creativity + Cognition + Computation, Montreal, Canada, July 24-27, 2019*, cognitivesciencesociety.org, pp. 58–59.

[265] Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D. and Jimenez Rezende, D. [2019]. Towards interpretable reinforcement learning using attention augmented agents, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox and R. Garnett (eds), *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 12350–12359.

[266] Mozer, M. C. [1989]. Types and tokens in visual letter perception, *Journal of experimental psychology: Human perception and performance* **15**(2): 287–303.

[267] Mozer, M. C. and Das, S. [1993]. A connectionist symbol manipulator that discovers the structure of context-free languages, *Advances in Neural Information Processing Systems*, pp. 863–870.

[268] Mozer, M. C., Kazakov, D. and Lindsey, R. V. [2018]. State-denoised recurrent neural networks, *cs.colorado.edu* .

[269] Mozer, M. C., Zemel, R. S. and Behrmann, M. [1992]. Learning to segment images using dynamic feature binding, *in* J. E. Moody, S. J. Hanson and R. P. Lippmann (eds), *Advances in Neural Information Processing Systems 4*, Morgan-Kaufmann, pp. 436–443.

[270] Mrowca, D., Zhuang, C., Wang, E., Haber, N., Fei-Fei, L., Tenenbaum, J. B. and Yamins, D. L. K. [2018]. Flexible neural representation for physics prediction.

[271] Munkhdalai, T. and Yu, H. [2017]. Neural semantic encoders, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Vol. 1, pp. 397–407.

[272] Nanbo, L., Eastwood, C. and Fisher, R. [2020]. Learning object-centric representations of multi-object scenes from multiple views, *Advances in Neural Information Processing Systems* **33**.

[273] Nash, C., Eslami, S. M. A., Burgess, C. P., Higgins, I., Zoran, D., Weber, T. and Battaglia, P. W. [2017]. The multi-entity variational autoencoder, *Neural Information Processing Systems (NeurIPS) Workshop on Learning Disentangled Representations: from Perception to Control* .

[274] Neto, C. R. and Fontanari, J. F. [1999]. Multivalley structure of attractor neural networks, *Journal of Physics A: Mathematical and General* **30**(22): 7945.

[275] Newell, A., Shaw, J. C. and Simon, H. A. [1959]. Report on a general problem solving program, *IFIP Congress*, Vol. 256, Pittsburgh, PA, p. 64.

[276] Newell, A. and Simon, H. A. [1981]. Computer science as empirical inquiry: Symbols and search, *Mind design* p. 4l.

[277] Nguyen-Phuoc, T., Richardt, C., Mai, L., Yang, Y.-L. and Mitra, N. [2020]. BlockGAN: Learning 3D object-aware scene representations from unlabelled images, *arXiv preprint arXiv:2002.08988* .

[278] Nickel, M. and Kiela, D. [2017]. Poincaré embeddings for learning hierarchical representations, *Advances in Neural Information Processing Systems*, Vol. 30, pp. 6338–6347.

[279] Niemeyer, M. and Geiger, A. [2020]. GIRAFFE: Representing scenes as compositional generative neural feature fields, *arXiv preprint arXiv:2011.12100* .

[280] Nissim, M., van Noord, R. and van der Goot, R. [2019]. Fair is better than sensational: Man is to doctor as woman is to doctor, *arXiv:1905.09866 [cs]* .

[281] Nowicki, D. and Siegelmann, H. T. [2010]. Flexible kernel memory, *PLoS One* **5**(6): e10955.

[282] Odena, A., Dumoulin, V. and Olah, C. [2016]. Deconvolution and checkerboard artifacts, *Distill* .

[283] Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M. and Carter, S. [2020]. Zoom in: An introduction to circuits, *Distill* **5**(3): e00024.001.

[284] Olah, C., Mordvintsev, A. and Schubert, L. [2017]. Feature visualization, *Distill* **2**(11): e7.

[285] Olshausen, B. A. and Field, D. J. [1996]. Emergence of simple-cell receptive field properties by learning a sparse code for natural images, *Nature* **381**(6583): 607–609.

[286] Orbán, G., Fiser, J., Aslin, R. N. and Lengyel, M. [2008]. Bayesian learning of visual chunks by human observers, *Proc. Natl. Acad. Sci. U. S. A.* **105**(7): 2745–2750.

[287] O'Reilly, R. C. and Busby, R. S. [2002]. Generalizable relational binding from coarse-coded distributed representations, *Adv. Neural Inf. Process. Syst.* **1**: 75–82.

[288] Paletta, L., Fritz, G. and Seifert, C. [2005]. Q-learning of sequential attention for visual object recognition from informative local descriptors, *International Conference on Machine Learning*, New York, NY, USA, pp. 649–656.

[289] Palm, R. B., Paquet, U. and Winther, O. [2018]. Recurrent relational networks, *arXiv:1711.08028 [cs]* .

[290] Pascanu, R., Mikolov, T. and Bengio, Y. [2012]. Understanding the exploding gradient problem, *CoRR, abs/1211. 5063* .

[291] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. [2019]. PyTorch: An imperative style, high-performance deep learning library, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox and R. Garnett (eds), *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., pp. 8024–8035.

[292] Pathak, D., Girshick, R. B., Dollár, P., Darrell, T. and Hariharan, B. [2016]. Learning features by watching objects move, *arXiv:1612.06370 [cs, stat]* .

[293] Pearl, J. [2019]. The seven tools of causal inference, with reflections on machine learning, *Communications of the ACM* **62**(3): 54–60.

[294] Peters, J., Bühlmann, P. and Meinshausen, N. [2016]. Causal inference by using invariant prediction: Identification and confidence intervals, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **78**(5): 947–1012.

[295] Peters, J., Janzing, D. and Schölkopf, B. [2017]. *Elements of Causal Inference: Foundations and Learning Algorithms*, MIT press.

[296] Pham, V., Bluche, T., Kermorvant, C. and Louradour, J. [2013]. Dropout improves recurrent neural networks for handwriting recognition, *arXiv:1312. 4569 [cs]* .

[297] Pineda, F. J. [1987]. Generalization of back-propagation to recurrent neural networks, *Physical review letters* **59**(19): 2229–2232.

[298] Plate, T. A. [1995]. Holographic reduced representations, *IEEE Trans. Neural Netw.* **6**(3): 623–641.

[299] Pollack, J. B. [1990]. Recursive distributed representations, *Artificial Intelligence* **46**(1-2): 77–105.

[300] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. [2019]. Language models are unsupervised multitask learners, *OpenAI Blog* **1**(8): 9.

[301] Rand, W. M. [1971]. Objective criteria for the evaluation of clustering methods, *J. Am. Stat. Assoc.* **66**(336): 846–850.

[302] Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J. and Black, M. J. [2019]. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12240–12249.

[303] Rao, A. R. and Cecchi, G. A. [2010]. An objective function utilizing complex sparsity for efficient segmentation in multi-layer oscillatory networks, *Int. J. Intell. Comput. Cybern.* **3**(2): 173–206.

[304] Rao, A. R., Cecchi, G. A., Peck, C. C. and Kozloski, J. R. [2008]. Unsupervised segmentation with dynamical units, *IEEE Trans. Neural Netw.* **19**(1): 168–182.

[305] Rasmus, A., Berglund, M., Honkala, M., Valpola, H. and Raiko, T. [2015]. Semi-supervised learning with ladder networks, *NIPS*, pp. 3532–3540.

[306] Redmon, J. and Farhadi, A. [2018]. YOLOv3: An incremental improvement, *CoRR* **abs/1804.02767**.

[307] Reed, S. and de Freitas, N. [2015]. Neural programmer-interpreters, *International Conference on Learning Representations*.

[308] Reichert, D. P., Seriès, P. and Storkey, A. J. [2011]. A hierarchical generative model of recurrent object-based attention in the visual cortex, *ICANN*, Springer, pp. 18–25.

[309] Reichert, D. P. and Serre, T. [2013]. Neuronal synchrony in complex-valued deep networks, *arXiv:1312. 6115 [cs, q-bio, stat]* .

[310] Ren, M. and Zemel, R. S. [2017]. End-to-end instance segmentation with recurrent attention, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Honolulu, HI, pp. 293–301.

[311] Ren, S., He, K., Girshick, R. B. and Sun, J. [2015]. Faster R-CNN: Towards real-time object detection with region proposal networks, *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 91–99.

[312] Ren, X. and Malik, J. [2003]. Learning a classification model for segmentation, *Proceedings Ninth IEEE International Conference on Computer Vision*, ieeexplore.ieee.org, pp. 10–17 vol.1.

[313] Rezende, D. J., Mohamed, S. and Wierstra, D. [2014]. Stochastic backpropagation and approximate inference in deep generative models, *in* E. P. Xing and T. Jebara (eds), *Proceedings of Machine Learning Research*, Vol. 32, PMLR, Bejing, China, pp. 1278–1286.

[314] Ridgeway, K. and Mozer, M. C. [2018]. Learning deep disentangled embeddings with the F-statistic loss.

[315] Riedmiller, M. and Braun, H. [1993]. A direct adaptive method for faster backpropagation learning: The RPROP algorithm, *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1.

[316] Robinson, A. J. and Fallside, F. [1987]. The utility driven dynamic error propagation network, *Technical report*, Cambridge University Engineering Department.

[317] Romaszko, L., Williams, C. K. I., Moreno, P. and Kohli, P. [2017]. Vision-as-inverse-graphics: Obtaining a rich 3D explanation of a scene from a single image, *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 851–859.

[318] Rosenblatt, F. [1961]. Principles of neurodynamics. perceptrons and the theory of brain mechanisms, *Technical report*, DTIC Document.

[319] Roskies, A. L. [1999]. The binding problem, *Neuron* **24**(1): 7–9, 111–25.

[320] Ross, D. A. and Zemel, R. S. [2006]. Learning parts-based representations of data, *J. Mach. Learn. Res.* **7**: 2369–2397.

[321] Ross, S. M. [2014]. *A First Course in Probability*, ninth edition edn, Pearson, Boston.

[322] Rowley, H. A., Baluja, S. and Kanade, T. [1998]. Neural network-based face detection, *IEEE Transactions on pattern analysis and machine intelligence* **20**(1): 23–38.

[323] Rozell, C. J., Johnson, D. H., Baraniuk, R. G. and Olshausen, B. A. [2008]. Sparse coding via thresholding and local competition in neural circuits, *Neural computation* **20**(10): 2526–2563.

[324] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. [1986]. Learning internal representations by error propagation, *in* D. E. Rumelhart and J. L. McClelland (eds), *Parallel Distributed Processing*, Vol. 1, MIT Press, pp. 318–362.

[325] Rumelhart, D. E., McClelland, J. L. and Research Group, P. [1987]. *Parallel Distributed Processing*, MIT press Cambridge, MA, USA.

[326] Sabour, S., Frosst, N. and Hinton, G. E. [2017]. Dynamic routing between capsules, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (eds), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pp. 3856–3866.

[327] Sak, H., Senior, A. and Beaufays, F. [2014]. Long short-term memory recurrent neural network architectures for large scale acoustic modeling, *Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH)*.

[328] Sak, H., Vinyals, O., Heigold, G., Senior, A., McDermott, E., Monga, R. and Mao, M. [2014]. Sequence discriminative distributed training of long short-term memory recurrent neural networks, *Interspeech*.

[329] Samadani, R. [1995]. A finite mixtures algorithm for finding proportions in SAR images, *IEEE Trans. Image Process.* **4**(8): 1182–1186.

[330] Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R. and Lillicrap, T. [2018]. Relational recurrent neural networks, *Advances in Neural Information Processing Systems*, pp. 7299–7310.

[331] Santoro, A., Hill, F., Barrett, D. G. T., Morcos, A. and Lillicrap, T. [2018]. Measuring abstract reasoning in neural networks, *in* J. Dy and A. Krause (eds), *Proceedings of Machine Learning Research*, Vol. 80, PMLR, Stockholmsmässan, Stockholm Sweden, pp. 4477–4486.

[332] Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P. W. and Lillicrap, T. [2017]. A simple neural network module for relational reasoning, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (eds), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pp. 4967–4976.

[333] Saund, E. [1995]. A multiple cause mixture model for unsupervised learning, *Neural Comput.* **7**(1): 51–71.

[334] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G. [2009]. The graph neural network model, *IEEE Transactions on Neural Networks* **20**(1): 61–80.

[335] Schlag, I. and Schmidhuber, J. [2018]. Learning to reason with third order tensor products, *Advances in Neural Information Processing Systems*, pp. 9981–9993.

[336] Schlag, I., Smolensky, P., Fernandez, R., Jojic, N., Schmidhuber, J. and Gao, J. [2019]. Enhancing the transformer with explicit relational encoding for math problem solving, *arXiv preprint arXiv:1910.06611* .

[337] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I. and Welling, M. [2018]. Modeling relational data with graph convolutional networks, *in* A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai and M. Alam (eds), *The Semantic Web*, Lecture Notes in Computer Science, Cham, pp. 593–607.

[338] Schmidhuber, J. [1992a]. Learning factorial codes by predictability minimization, *Neural Comput.* **4**(6): 863–879.

[339] Schmidhuber, J. [1992b]. Learning to control fast-weight memories: An alternative to dynamic recurrent networks, *Neural Comput.* **4**(1): 131–139.

[340] Schmidhuber, J. [1993a]. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets, *ICANN'93*, Springer, pp. 460–463.

[341] Schmidhuber, J. [1993b]. A "self-referential" weight matrix, *ICANN'93*, Springer, pp. 446–450.

[342] Schmidhuber, J. [2015]. Deep learning in neural networks: An overview, *Neural Netw.* **61**: 85–117.

[343] Schmidhuber, J. and Huber, R. [1991]. Learning to generate artificial fovea trajectories for target detection, *Int. J. Neural Syst.* **2**(01n02): 125–134.

[344] Schölkopf, B. [2019]. Causality for machine learning, *arXiv preprint arXiv:1911.10500* .

[345] Schwartz, J.-L., Grimault, N., Hupé, J.-M., Moore, B. C. J. and Pressnitzer, D. [2012]. Multistability in perception: Binding sensory modalities, an overview, *Philosophical Transactions of the Royal Society B* **367**(1591): 896–905.

[346] Sherrington, D. and Kirkpatrick, S. [1975]. Solvable Model of a Spin-Glass, *Physical Review Letters* **35**(26): 1792–1796.

[347] Shi, J. and Malik, J. [2000]. Normalized cuts and image segmentation, *IEEE Transactions on pattern analysis and machine intelligence* **22**(8): 888–905.

[348] Shortliffe, E. H., Davis, R., Axline, S. G., Buchanan, B. G., Green, C. C. and Cohen, S. N. [1975]. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system, *Computers and biomedical research* **8**(4): 303–320.

[349] Siegelmann, H. T. and Sontag, D. [1991]. Turing computability with neural nets, *Applied Mathematics Letters* **4**(6): 77–80.

[350] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D. and Graepel, T. [2017]. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, *arXiv preprint arXiv:1712.01815* .

[351] Singer, W. [1999]. Neuronal synchrony: A versatile code for the definition of relations?, *Neuron* **24**(1): 49–65, 111–25.

[352] Singer, W. [2001]. Consciousness and the binding problem, *Annals of the New York Academy of Sciences* **929**(1): 123–146.

[353] Singer, W. [2009]. Distributed processing and temporal codes in neuronal networks, *Cognitive neurodynamics* **3**(3): 189–196.

[354] Singer, W. [2018]. Neuronal oscillations: Unavoidable and useful?, *European Journal of Neuroscience* **48**(7): 2389–2398.

[355] Smolensky, P. [1987]. Analysis of distributed representation of constituent structure in connectionist systems, *Neural Information Processing Systems*, pp. 730–739.

[356] Smolensky, P. [1988]. On the proper treatment of connectionism, *Behav. Brain Sci.* **11**(1): 1–23.

[357] Smolensky, P. [1990]. Tensor product variable binding and the representation of symbolic structures in connectionist systems, *Artificial intelligence* **46**(1): 159–216.

[358] Sohn, K., Zhou, G., Lee, C. and Lee, H. [2013]. Learning and selecting features jointly with point-wise gated Boltzmann machines, *Proceedings of The 30th International Conference on Machine Learning*, pp. 217–225.

[359] Sollow, E. [1987]. Assessing the maintainability of XCQN-in-RIME: Coping with the problems of a VERY large rule-base.

[360] Solomonoff, R. [1964]. A formal theory of inductive inference i, *Information and Control* **7**: 1–22.

[361] Sønderby, S. K. and Winther, O. [2014]. Protein secondary structure prediction with long short term memory networks, *arXiv:1412. 7828 [cs, q-bio]* .

[362] Sperduti, A. and Starita, A. [1997]. Supervised neural networks for the classification of structures, *IEEE Transactions on Neural Networks* **8**(3): 714–735.

[363] Sprechmann, P., Bronstein, A. M. and Sapiro, G. [2015]. Learning efficient sparse and low rank models, *IEEE Trans. Pattern Anal. Mach. Intell.* **37**(9): 1821–1833.

[364] Springenberg, J. T., Dosovitskiy, A., Brox, T. and Riedmiller, M. [2014]. Striving for simplicity: The all convolutional net, *arXiv preprint arXiv:1412.6806* .

[365] Srikumar, V., Kundu, G. and Roth, D. [2012]. On amortizing inference cost for structured prediction, *EMNLP-CoNLL '12*, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1114–1124.

[366] Stanić, A., van Steenkiste, S. and Schmidhuber, J. [2020]. Hierarchical relational inference, *Proceedings of the AAAI Conference on Artificial Intelligence*.

[367] Stanley, K. O. and Miikkulainen, R. [2004]. Evolving a roving eye for go, *Citeseer* .

[368] Stephen, C. [1956]. Kleene. Representation of events in nerve nets and finite automata, *Automata studies* .

[369] Sukhbaatar, S., szlam, a., Weston, J. and Fergus, R. [2015]. End-to-end memory networks, *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 2440–2448.

[370] Sun, C., Karlsson, P., Wu, J., Tenenbaum, J. B. and Murphy, K. [2019]. Stochastic prediction of multi-agent interactions from partial observations, *arXiv preprint arXiv:1902.09641* .

[371] Sun, R. [1992]. On variable binding in connectionist networks, *Connection Science* **4**(2): 93–124.

[372] Sundararajan, M., Taly, A. and Yan, Q. [2017]. Axiomatic attribution for deep networks, *arXiv:1703.01365 [cs]* .

[373] Tai, K. S., Socher, R. and Manning, C. D. [2015]. Improved semantic representations from tree-structured long short-term memory networks, *arXiv preprint arXiv:1503.00075* .

[374] Tang, Y., Salakhutdinov, R. R. and Hinton, G. E. [2012]. Robust Boltzmann machines for recognition and denoising, *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference On*, IEEE, pp. 2264–2271.

[375] Tang, Y., Srivastava, N. and Salakhutdinov, R. R. [2014]. Learning generative models with visual attention, *in* Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pp. 1808–1816.

[376] The Theano Development Team [2016]. Theano: A python framework for fast computation of mathematical expressions, *arXiv:1605.02688 [cs]* .

[377] Treisman, A. [1996]. The binding problem, *Current opinion in neurobiology* **6**(2): 171–178.

[378] Treisman, A. [1999]. Solutions to the binding problem: Progress through controversy and convergence, *Neuron* **24**(1): 105–10, 111–25.

[379] Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B. and Gershman, S. J. [2017]. Human learning in Atari, *2017 AAAI Spring Symposium Series*.

[380] Tu, Z., Chen, X., Yuille, A. L. and Zhu, S.-C. [2005]. Image parsing: Unifying segmentation, detection, and recognition, *Int. J. Comput. Vis.* **63**(2): 113–140.

[381] Tu, Z. and Zhu, S.-C. [2002]. Image segmentation by data-driven Markov chain Monte Carlo, *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(5): 657–673.

[382] Uhlhaas, P., Pipa, G., Lima, B., Melloni, L., Neuenschwander, S., Nikolić, D. and Singer, W. [2009]. Neural synchrony in cortical networks: History, concept and current status, *Frontiers in integrative neuroscience* **3**: 17.

[383] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T. and Smeulders, A. W. M. [2013]. Selective search for object recognition, *International journal of computer vision* **104**(2): 154–171.

[384] van der Maaten, L. and Hinton, G. E. [2008]. Visualizing data using t-SNE, *Journal of machine learning research* **9**(Nov): 2579–2605.

[385] van Steenkiste, S., Chang, M., Greff, K. and Schmidhuber, J. [2018]. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions, *ICLR*.

[386] van Steenkiste, S., Kurach, K., Schmidhuber, J. and Gelly, S. [2019]. Investigating object compositionality in generative adversarial networks, *arXiv:1810.10340 [cs]* .

[387] van Steenkiste, S., Locatello, F., Schmidhuber, J. and Bachem, O. [2019]. Are disentangled representations helpful for abstract visual reasoning?, *arXiv:1905.12506 [cs, stat]* .

[388] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. [2017]. Attention is all you need, *Advances in Neural Information Processing Systems*, pp. 5998–6008.

[389] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. [2018]. Graph Attention Networks, *International Conference on Learning Representations*.

[390] Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R. and Fragkiadaki, K. [2017]. SfM-net: Learning of structure and motion from video, *arXiv:1704.07804 [cs]* .

[391] Vincent, P., Larochelle, H., Bengio, Y. and Manzagol, P.-A. [2008]. Extracting and composing robust features with denoising autoencoders, *Proceedings of the 25th International Conference on Machine Learning*, ACM, pp. 1096–1103.

[392] Vinh, N. X., Epps, J. and Bailey, J. [2010]. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, *J. Mach. Learn. Res.* **11**: 2837–2854.

[393] Vinyals, O., Fortunato, M. and Jaitly, N. [2015]. Pointer networks, *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 2692–2700.

[394] Viola, P. and Jones, M. J. [2001]. Rapid object detection using a boosted cascade of simple features, *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. I–511–I–518 vol.1.

[395] von der Malsburg, C. [1981]. The correlation theory of brain function, *MPI* .

[396] Von Der Malsburg, C. [1986]. Am I thinking assemblies?, *Brain Theory*, Springer, pp. 161–176.

[397] von der Malsburg, C. [1995]. Binding in models of perception and brain function, *Curr. Opin. Neurobiol.* **5**(4): 520–526.

[398] von Kügelgen, J., Ustyuzhaninov, I., Gehler, P., Bethge, M. and Schölkopf, B. [2020]. Towards causal generative scene models via competition of experts, *International Conference on Learning Representations (ICLR) Workshop on "Causal learning for decision making"* .

[399] Wagemans, J. [2015]. *The Oxford Handbook of Perceptual Organization*, Oxford University Press.

[400] Wagemans, J., Elder, J. H., Kubovy, M., Palmer, S. E., Peterson, M. A., Singh, M. and von der Heydt, R. [2012]. A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization, *psycnet.apa.org* .

[401] Wagemans, J., Feldman, J., Gepshtein, S., Kimchi, R., Pomerantz, J. R., van der Helm, P. A. and van Leeuwen, C. [2012]. A century of Gestalt psychology in visual perception: II. Conceptual and theoretical foundations, *Psychol. Bull.* **138**(6): 1218–1252.

[402] Wang, D. [2005]. The time dimension for scene analysis, *IEEE Trans. Neural Netw.* **16**(6): 1401–1426.

[403] Wang, D. and Terman, D. [1995]. Locally excitatory globally inhibitory oscillator networks, *IEEE Trans. Neural Netw.* **6**(1): 283–286.

[404] Wang, X., Girshick, R., Gupta, A. and He, K. [2018]. Non-local neural networks, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

[405] Watters, N., Matthey, L., Burgess, C. P. and Lerchner, A. [2019]. Spatial broadcast decoder: A simple architecture for learning disentangled representations in VAEs.

[406] Weiss, Y. and Adelson, E. H. [1996]. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models, *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ieeexplore.ieee.org, pp. 321–326.

[407] Weizenbaum, J. [1966]. ELIZA—a computer program for the study of natural language communication between man and machine, *Communications of the ACM* **9**(1): 36–45.

[408] Weng, S., Steil, J. J. and Ritter, H. J. [2006]. Learning lateral interactions for feature binding and sensory segmentation from prototypic basis interactions, *IEEE Trans. Neural Netw.* **17**(4): 843–862.

[409] Werbos, P. J. [1982]. Applications of advances in nonlinear sensitivity analysis, *in* R. F. Drenick and F. Kozin (eds), *System Modeling and Optimization*, Lecture Notes in Control and Information Sciences, Springer, Berlin, Heidelberg, pp. 762–770.

[410] Werbos, P. J. [1988]. Generalization of backpropagation with application to a recurrent gas market model, *Neural Netw.* **1**(4): 339–356.

[411] Wersing, H., Steil, J. J. and Ritter, H. J. [2001]. A competitive-layer model for feature binding and sensory segmentation, *Neural Comput.* **13**(2): 357–387.

[412] Wertheimer, M. [1912]. Experimentelle Studium uber das Sehen von Bewegung, *Z. Psychol.* **61**(3): 161–265.

[413] Wertheimer, M. [1923]. Untersuchungen zur Lehre von der Gestalt II, *Psychol. Forsch.* **4**(1): 301–350.

[414] Weston, J., Chopra, S. and Bordes, A. [2014]. Memory networks.

[415] Williams, R. J. and Zipser, D. [1989]. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, *Neural Computation* **1**(2): 270–280.

[416] Winograd, T. [1971]. Procedures as a representation for data in a computer program for understanding natural language, *Technical report*, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

[417] Wu, C. F. J. [1983]. On the convergence properties of the EM algorithm, *The Annals of statistics* pp. 95–103.

[418] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q. and Macherey, K. [2016]. Google's neural machine translation system: Bridging the gap between human and machine translation, *arXiv preprint arXiv:1609.08144* .

[419] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P. S. [2019]. A comprehensive survey on graph neural networks, *arXiv preprint arXiv:1901.00596* .

[420] Xu, K., Ba, J., Kiros, J. R., Courville, A., Salakhutdinov, R. R., Zemel, R. S. and Bengio, Y. [2015]. Show, attend and tell: Neural image caption generation with visual attention, *arXiv preprint arXiv:1502.03044* .

[421] Yang, C., Zhuang, P., Shi, W., Luu, A. and Li, P. [2019]. Conditional structure generation through graph variational generative adversarial nets, *Advances in Neural Information Processing Systems*, pp. 1338–1349.

[422] Yang, Y., Chen, Y. and Soatto, S. [2020]. Learning to manipulate individual objects in an image, *arXiv:2004.05495 [cs]* .

[423] Yli-Krekola, A., Särelä, J. and Valpola, H. [2009]. Selective attention improves learning, *Artificial Neural Networks–ICANN 2009*, Springer, pp. 285–294.

[424] Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. [2014]. How transferable are features in deep neural networks?, *Advances in Neural Information Processing Systems*, pp. 3320–3328.

[425] Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M., Vinyals, O. and Battaglia, P. [2018]. Deep reinforcement learning with relational inductive biases, *International Conference on Learning Representations*.

[426] Zaremba, W., Sutskever, I. and Vinyals, O. [2014]. Recurrent neural network regularization, *arXiv:1409. 2329 [cs]* .

[427] Zeiler, M. D. and Fergus, R. [2014]. Visualizing and understanding convolutional networks, *European Conference on Computer Vision*, Springer, pp. 818–833.

[428] Zemel, R. S. and Mozer, M. C. [2001]. Localist attractor networks, *Neural Comput.* **13**(5): 1045–1064.

[429] Zemel, R. S., Williams, C. K. I. and Mozer, M. C. [1995]. Lending direction to neural networks, *Neural Netw.* **8**(4): 503–512.

[430] Zhang, C., Vinyals, O., Munos, R. and Bengio, S. [2018]. A study on overfitting in deep reinforcement learning, *arXiv:1804.06893 [cs, stat]* .

[431] Zhang, M., Jiang, S., Cui, Z., Garnett, R. and Chen, Y. [2019]. D-VAE: A variational autoencoder for directed acyclic graphs, *arXiv preprint arXiv:1904.11088* .

[432] Zhao, Y. and Zhu, S.-C. [2011]. Image parsing with stochastic scene grammar, *in* J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 24*, Curran Associates, Inc., pp. 73–81.

[433] Zhmoginov, A., Fischer, I. and Sandler, M. [2019]. Information-bottleneck approach to salient region discovery, *arXiv preprint arXiv:1907.09578* .

[434] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. and Torralba, A. [2014]. Object detectors emerge in deep scene CNNs.

[435] Zoran, D., Chrzanowski, M., Huang, P.-S., Gowal, S., Mott, A. and Kohl, P. [2019]. Towards robust image classification using sequential attention models, *arXiv:1912.02184 [cs]* .

List of Figures

List of Tables

Acronyms

Adam Adaptive Moment Estimation. 76, 90, 106, 124

AIR Attend Infer Repeat. 12, 23, 124

AMI Adjusted Mutual Information. 60, 62, 66, 76, 77, 80, 81, 90, 91, 124, 155, 159

ANN Artificial Neural Network. 44, 48, 124

ARI Adjusted Rand Index. 110, 124

BN BatchNorm. 89, 91, 94, 124, 159

BPTT Backpropagation Through Time. 73, 124

CNN Convolutional Neural Network. 12, 124

DAE Denoising Autoencoder. 3, 55, 58, 59, 60, 62, 66, 67, 68, 69, 70, 71, 124, 155

ELU Exponential Linear Unit. 45, 77, 79, 124

EM Expectation Maximization. 44, 55, 57, 58, 71, 72, 73, 75, 76, 81, 124

FIT Feature Integration Theory. 124

GAN Generative Adversarial Network. 32, 124

GCN Graph Convolutional Network. 31, 124

GNN Graph Neural Network. 30, 31, 32, 33, 124

GRU Gated Recurrent Unit. 124

ICA Independent Component Analysis. 24, 124

IODINE Iterative Object Decomposition Inference NEtwork. 3, 4, 12, 99, 100, 101, 103, 104, 105, 106, 110, 111, 112, 114, 115, 116, 118, 119, 120, 121, 122, 124, 157, 158, 159

LN LayerNorm. 91, 124, 159

LSTM Long Short-Term Memory. 49, 50, 124

MLE	Maximum Likelihood Estimate. 46, 72, 73, 124
MLP	Multi-Layer Perceptron. 11, 44, 45, 48, 124
MONet	Multi-Object Network. 12, 23, 110, 124
MPNN	Message Passing Neural Network. 31, 32, 124
MRBM	Masked Restricted Boltzman Machine. 12, 69, 75, 124
N-EM	Neural Expectation Maximization. 3, 12, 71, 72, 73, 74, 75, 76, 77, 81, 105, 110, 124, 155
NN	Neural Network. 45, 46, 47, 59, 71, 124
NTM	Neural Turing Machine. 33, 124
RBM	Restricted Boltzman Machine. 69, 75, 124
ReLU	Rectified Linear Unit. 44, 77, 79, 94, 124
R-NEM	Relational Neural Expectation Maximization. 32, 110, 124
RC	Reconstruction Clustering. 3, 4, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 71, 81, 83, 124, 154, 155
RFT	Relational Frame Theory. 28, 124
RIM	Recurrent Independent Mechanism. 12, 124
RNN	Recurrent Neural Network. 11, 12, 14, 15, 31, 32, 33, 49, 73, 74, 75, 96, 124
RNN-EM	Recurrent Neural Network Expectation Maximization. 74, 75, 76, 77, 78, 80, 81, 110, 124, 155
RTRL	Real Time Recurrent Learning. 124
SGD	Stochastic Gradient Descent. 60, 124
SRN	Simple Recurrent Network. 48, 49, 50, 124, 154
STDP	Spike-Timing Dependent Plasticity. 6, 14, 124
TAG	iTerative Amortized Grouping. 3, 83, 88, 91, 94, 96, 99, 124
TPR	Tensor Product Representation. 14, 33, 34, 124
VAE	Variational Autoencoder. 3, 32, 44, 99, 100, 101, 102, 103, 105, 112, 114, 117, 118, 120, 124, 157