Generative neuro-symbolic models of concept learning

by

Reuben Feinman

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Center for Neural Science

New York University

September 2023

Brenden M. Lake

Acknowledgments

First and foremost I would like to thank my thesis advisor, Brenden Lake. Brenden's ideas and mentorship are ingrained in all aspects of this thesis. His perspective has had a major influence on the way that I think about the field, and I am hugely grateful for it. He has showed selfless engagement with all aspects of my PhD, helping me apply for fellowships, write papers, and even get a summer internship at Facebook (Meta). I've met very few faculty who are as attentive and supportive of their students as Brenden is. I am indebted to him for this support.

I am also grateful to members of my thesis committee for providing invaluable feedback during every stage of my PhD which helped shape the direction of this thesis. These include Eero Simoncelli, Cristina Savin, and Joan Bruna. In particular, my committee helped direct my research focus and draw connections to human behavior.

I am grateful to current and past members of the MIT CoCoSci lab for fruitful discussions regarding the work of Chapters 2 & 3. These include Max Nye, Josh Tenenbaum, Tuan-Anh Le, and Lucas Tian. Max was particularly helpful during early brainstorming about generative models, and Tuan-Anh while thinking about approaches to probabilistic inference.

In addition, I would like to thanks Stéphane Deny for valuable feedback on an earlier draft of Chapter 3, and Jay McClelland for helpful discussions regarding this work.

I'd like to thank my classmate Nikhil Parthsarathy for engaging me in fruitful conversations and interactions throughout the course of my PhD. In addition to regular discussions, Nikhil also provided valuable feedback on earlier drafts of Chapter 6.

I am thankful to Guy Davidson and Emin Orhan for helpful discussions regarding Chapter 4. Emin also provided valuable comments on a draft of Chapter 6.

I am grateful to all members of the Human and Machine Learning Lab (HMLL) for providing thoughtful feedback during lab meetings at various stages of this thesis.

My research was supported by a Google PhD Fellowship in Computational Neuroscience, and by NSF Award 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science. I am grateful to these funding sources.

I would like to thank my brother, Nick; my mother, Mary; and my late father Andy. Each of you helped inspire me to fulfill my potential and make a contribution to science as I often dreamed of doing growing up.

Finally, I am grateful to my wife, Charlotte, who supported me throughout all years of my PhD and encouraged me through many challenging situations and difficult times. Your love and support have lifted me through these hardships and I couldn't have done it without you.

Abstract

Human concepts exhibit a collection of unique qualities that together are not well-explained by current computational methods. On the one hand, human conceptual knowledge is distinguished for its productivity and generality: people learn new concepts very quickly from just one or a few examples, and the representations they acquire can be applied flexibly to a variety of tasks without retraining. In addition, people's conceptual knowledge interacts directly with raw signals: people learn new concepts directly from raw, high-dimensional data, capturing complex correlations and invariances that support the recognition and generation of new examples in equally high-dimensional media. Two modeling traditions have explained different components of these empirical phenomena, but we lack a unified computational framework to understand and account for the collective capabilities.

This thesis presents a new computational framework for modeling human concepts that builds on two rich traditions in cognitive science. We hypothesize that human concept representations include a combination of structural and statistical ingredients, and that models with an appropriate synthesis of these ingredients will help account for the collective capabilities of human concept learning. Our approach—dubbed Generative Neuro-Symbolic (GNS) modeling—uses the control flow of a probabilistic program, coupled with symbolic primitives and renderers, to model the causal and compositional processes by which concepts are formed. At the same time, it integrates neural network subroutines to interface directly with raw data and capture complex correlations in observations. We demonstrate two instances of this approach developed to model human concepts

of handwritten characters and synthetic "alien figures." Our experiments show that GNS provides a useful framework to understand the dual structural and statistical natures of human concepts and account for a diversity of capabilities. Additional experiments explore alternate ways to integrate structural and statistical representation and account for psychological phenomena, studying the dynamics of learning-to-learn and the acquisition of inductive biases in neural network models.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

A signature of human intelligence is the ability to learn new concepts on the fly and make meaningful generalizations from limited observations. Consider, for example, a child learning the meaning of the word "fork" (Fig. 1.1). Given just one or a few examples of the new concept, a child can effectively reason about the meaning of the word (Bloom, 2000), confirmed by the ways in which he uses it in new contexts. Generalizing as such from limited data presents a challenging computational problem.

Not only do people learn new concepts so quickly, but the concept representations they acquire are often *task-general*, meaning they support a variety of unique capabilities (Fig. 1.2). These include discriminative tasks like recognizing new examples, as well as creative tasks like generating examples or imaging new concepts. In contrast, state-of-the-art machine learning methods are typically optimized for a single task (Lake et al., 2017; Yuille & Liu, 2019), and they have difficulty with more creative tasks such as example generation and structured imagination (Lake et al., 2019).

How do people acquire such rich representations from so little experience? What is the structure of the representation learned, and how does this structure support flexible generalization to a variety of tasks? Understanding these questions in computational terms represents the central motivation of

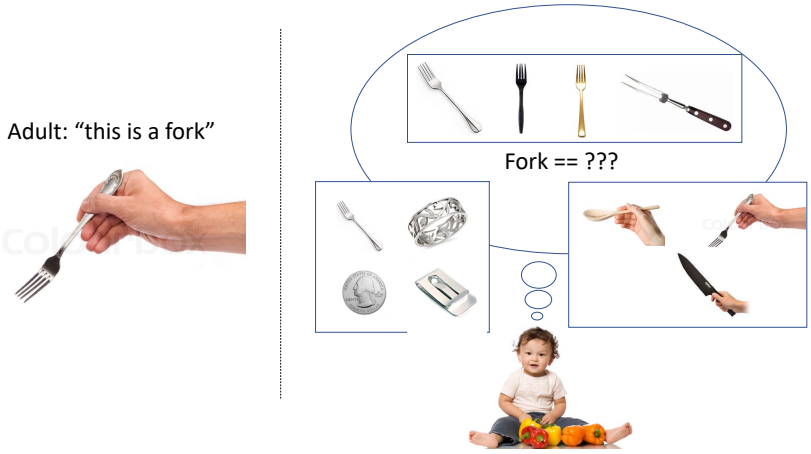


Figure 1.1: A child learning the meaning of “fork." Given just one example of the new concept, a large variety of models fit the observation. Correctly generalizing the word presents a challenging inductive problem.

this thesis.

There are two long-standing traditions in cognitive science for understanding conceptual knowledge, each of which has had its own successes and shortcomings in accounting for the capabilities discussed above. The first tradition emphasizes *structured knowledge* for representing causal and compositional processes. The second tradition emphasizes *statistical knowledge*, represented as patterns and correlations extracted from observations. Together, these two traditions provide a foundation for the contributions of this thesis.

1.1 Tradition 1: Structured knowledge

The first tradition centers around the idea that human concepts are embedded in intuitive theories (Murphy & Medin, 1985). According to this perspective, human knowledge of everyday concepts like cars, dogs and trees manifests like a scientific theory: it is predicated on a set of formalisms, and it provides explicit explanations for observations. Importantly, people’s concepts go beyond observable features to include a notion of the latent, unobserved *causal processes*.

As an example of structured knowledge, consider the graph in Fig. 1.3 which depicts a Bayesian

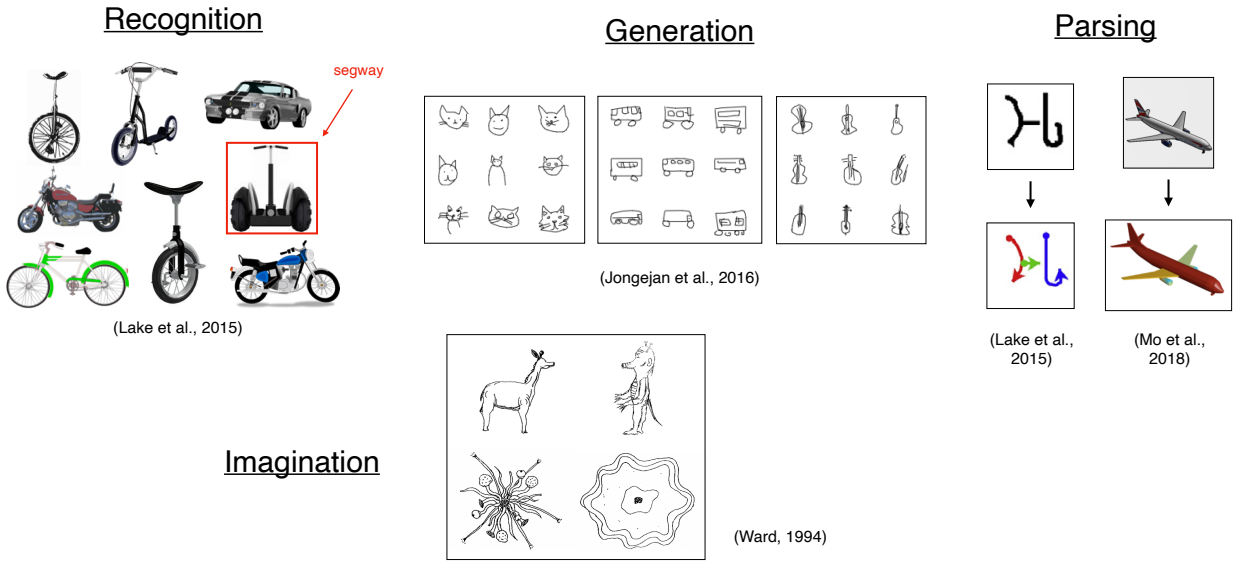


Figure 1.2: People's concept representations support a variety of capabilities including: 1) recognition, the ability to identify and distinguish new instances of the concept, 2) generation, the ability to synthesize new examples by a variety of media, 3) parsing an example into its parts and relations, and 4) imagination, the ability to creatively synthesize new concepts altogether which are novel yet structurally coherent.

network representation of the concept "bicycle." Each node in the network represents a knowledge variable. The variables below the dashed line are observed perceptual features, while those above the line are latent, unobserved factors. Arrows in the network represent causal influences between variables. In this example, the presence of latent factor "used for transportation" influences observed feature "has wheels," expressing the idea that wheels are used for mobility. Further, the feature "used for transportation" also influences the latent factor "material:" if a piece of machinery will be used for transportation, then it will likely be constructed from lightweight materials like aluminum or carbon fiber for efficiency. The material of an object influences its observed weight, and thus, through the intermediate latent factor "material," the factor "used for transportation" causally influences feature "weight."

The role of causality in human concept representations has been demonstrated in a variety of different ways. Studies have shown that people's representations of natural kinds include beliefs about a core, unobserved *essence* that is invariant to changes of observed features (S. A. Gelman,

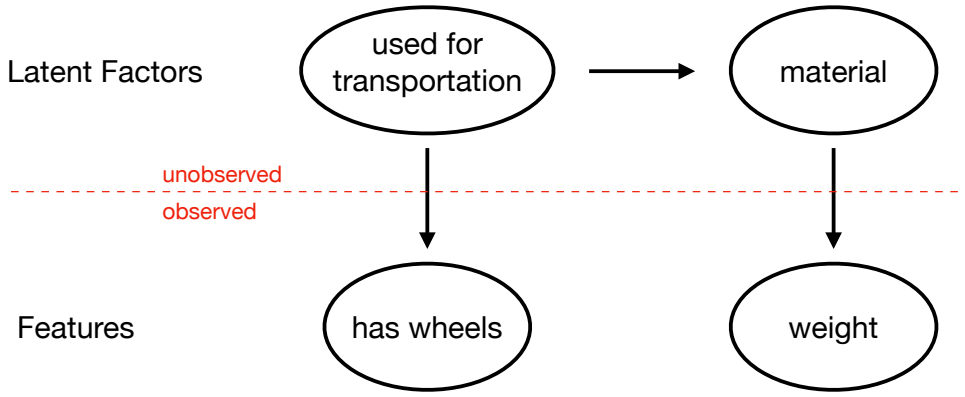


Figure 1.3: A Bayesian Network depiction of the causal knowledge underlying a concept like "bicycle."

2003; Keil, 1989). Moreover, manipulating people's causal knowledge about the features of objects has been shown to influence the ways that they learn and generalize (Rehder, 2003; Rehder & Hastie, 2001). Other works provide evidence for analysis-by-synthesis theories of perception (Eden, 1962; Halle & Stevens, 1962; Neisser, 1966; Bever & Poeppel, 2010): for instance, people's causal knowledge about how characters are drawn influences the way they recognize new characters (Freyd, 1983) and how they perceive motion in characters presented stroke-by-stroke (Tse & Cavanagh, 2000).

Structured knowledge is traditionally represented using symbolic primitives in a "language-of-thought" (Fodor, 1975). Symbolic models provide combinatorial syntax and semantics, offering a *compositional* representation that explains the systematicity of mental representation (Murphy, 2002). This approach was paramount in classical models of the mind, which took inspiration from computer architecture and emphasized logical representations and symbol manipulation. Concepts were viewed as rigid definitions or rules for categorization (Bruner et al., 1956). Although useful as a high-level framework, these logical representations fall short of explaining the majority of natural concepts (Murphy, 2002). Moreover, classical symbolic models like these do not account for the ways that people learn new concepts from experience, an emphasis of the second tradition.

One important shortcoming of the structured knowledge view is its lack of account for incomplete knowledge, a psychological phenomenon exemplified by Rozenblit & Keil (2002). If conceptual knowledge is embedded in theories, then we would expect that people are able to provide explanations of everyday concepts; however, Rozenblit & Keil (2002) showed that this is often not the case. When pressed to provide explanations about the workings of a speedometor, a zipper, a piano key, or a flush toilet, human participants from the study could only provide fragments of explanations for these entities. Furthermore, the level that participants rated their own knowledge of these concepts exhibited a sharp decline after being asked for an explanation. The authors called this phenomenon the “illusion of explanatory depth:" although we might think that we understand the workings of things like pianos and toilets, when pressed for explanations, we realize that our knowledge is actually very sparse.

1.2 Tradition 2: Statistical knowledge

The second tradition of research prioritizes *statistical knowledge*, a more amorphous form of conceptual knowledge that manifests patterns and correlations from observations. The meaning of a word, for example, is derived from its patterns of co-occurrence with other words (Deerwester et al., 1990). Similarly, latent representations of objects and other stimuli arise from accommodations of "suspicious coincidences" noted in the data (Barlow, 1989).

Central to the statistical tradition is the parallel distributed processing (PDP) approach of modeling cognitive processes (Rumelhart et al., 1987; Rogers & McClelland, 2004), also known as connectionism. In this approach, concepts are modeled with neural networks, a form of representation that consists of a hierarchy of neuron-like processing units. Using a technique known as backpropagation, neural networks can be trained to perform a variety of tasks including: learning exemplar-based representations of categories (Kruschke, 1992), assigning syntax to words from text sequences (Elman, 1990), and reasoning about semantic properties of natural kinds (Rogers &

McClelland, 2004). Importantly, these models can learn directly from raw data with little or no prior knowledge about a domain. Training neural networks to estimate unknown density functions and decision surfaces is considered an example of nonparametric (or model-free) statistical inference (Geman et al., 1992).

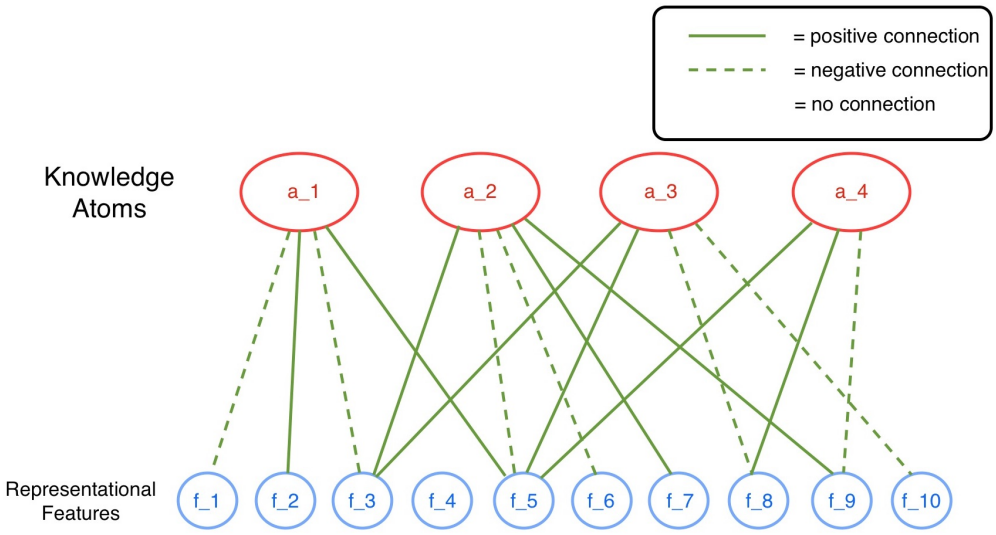


Figure 1.4: Visualization of a Restricted Boltzmann Machine (RBM), a neural network model inspired by harmony theory (Smolensky, 1987).

As a demonstrative example of how conceptual knowledge is represented in this tradition, consider the Restricted Boltzmann Machine (RBM)—a neural network model inspired by harmony theory (Smolensky, 1987)—depicted in Fig. 1.4. The RBM assigns probability to observations ("representational features") with an energy-based density model that is marginalized over a set of unobserved latent factors ("knowledge atoms"). During the training process, recurring feature patterns are detected and explained away by causal factors, following the "suspicious coincidences" learning principle (Barlow, 1989). To demonstrate this principle, consider an alien who visits Earth for the first time and is unfamiliar with human vehicles. The alien observes a number of objects rolling around with four wheels and a passenger compartment. In absence of a model for this compound entity, this repeated arrangement of parts is highly suspicious. Rather than accept this observation as coincidence, the alien creates a new knowledge atom "vehicle" to explain the

compound.

In a seminal work, Rogers & McClelland (2004) studied the development of semantic knowledge in a neural network trained to predict the attributes of biological kinds such as “canary," “rose," and “salmon." Attributes included whether or not the entity is an animal, whether it grows, and whether it has bark, among other things. Given a particular category as input, the network outputs probabilities for each of the possible attributes, and it is trained to maximize the likelihood of the correct attributes. Weights of the network are updated iteratively using the gradient-based backpropagation algorithm. Rogers & McClelland (2004) showed that the hidden units of their neural network learns to represent structured semantic knowledge about the organization of biological kinds. Importantly, these structured hypotheses emerge without the constraints of a formal hypothesis space.

More recently, neural network models have become widely adopted in machine learning research and have demonstrated powerful capabilities in domains including object recognition, speech recognition, and control (LeCun et al., 2015). Beginning with the success of AlexNet (Krizhevsky et al., 2012), a variety of neural networks for image recognition have achieved increasingly stronger performance on the challenging ImageNet benchmark. Sequence models such as Seq2Seq (Sutskever et al., 2014) and Transformer (Vaswani et al., 2017) have demonstrated promising results on natural language processing (NLP) tasks including translation and generation, a domain that has seen a very recent explosion of attention.

Although the computational approaches from this tradition help explain how people learn from raw data and capture complex patterns, neural network models have been criticized for lacking the type of compositional representation needed to support systematic generalization (Fodor & Pylyshyn, 1988; Marcus, 2003; Lake & Baroni, 2018). The internal models and representations that people develop can be applied flexibly to new tasks with little or no training experience, and neural networks provide an insufficient account for this type of flexible model building (Lake et al., 2017). Moreover, neural network models have difficulty replicating human behavior in generative tasks that require structured imagination and other creative abilities; even with relatively simple handwritten

characters, neurally-grounded models do not yet explain how people generate new concepts and new examples that are novel yet structurally consistent with familiar ones (Lake et al., 2019).

1.3 Integrating structure and statistics

The paradigms of tradition 1 and 2 each have unique strengths that help account for human concepts. Structured representations can explain the task-generality of concepts, attributing this flexibility to an explicit model of causal and compositional processes. On the other hand, statistical models help explain how concepts arise from experience and manifest patterns in observed data. A central motivation of this thesis is to explore new models of concepts that offer a synthesis of the structured and statistical traditions, combining the best qualities of each approach.

Previous efforts to integrate these two traditions have demonstrated ways of performing statistical inference over structured representations (Tenenbaum et al., 2011), offering an explanation for how concepts are learned from experience and providing an account of their graded nature. This includes models of concept learning as Bayesian inference over fully-symbolic expressions in formal logical (Goodman et al., 2008; Piantadosi et al., 2016), or models of inductive reasoning supported by structured intuitive theories (Kemp & Tenenbaum, 2009). In accounts of this nature, statistics is primary in selecting between structured symbolic hypotheses (Kemp & Tenenbaum, 2008; Perfors et al., 2011; Lake et al., 2015; Lake & Piantadosi, 2020), but plays little role in forming the individual hypotheses themselves. Specifically, each hypothesis may only have a few parametric distributions that need to be inferred, if any. Although more flexible than a priori structural constraints, models of this kind still make many assumptions, and they have not yet tackled the types of raw, high-dimensional stimuli that are distinctive of connectionist neural network models.

This thesis explores new ways to integrate the structured and statistical traditions and develop models that synthesize their key ingredients. The primary contribution is a new framework for

computational models of concepts that we denote Generative Neuro-Symbolic (GNS) modeling. Similar to the symbolic probabilistic models discussed above, a GNS model represents the *compositional* and *causal* structure of concepts, helping to support flexible generalization to a range of tasks. However, we advocate for a new layer of statistics that allows the model to represent complex correlations and interact directly with raw, high-dimensional data.

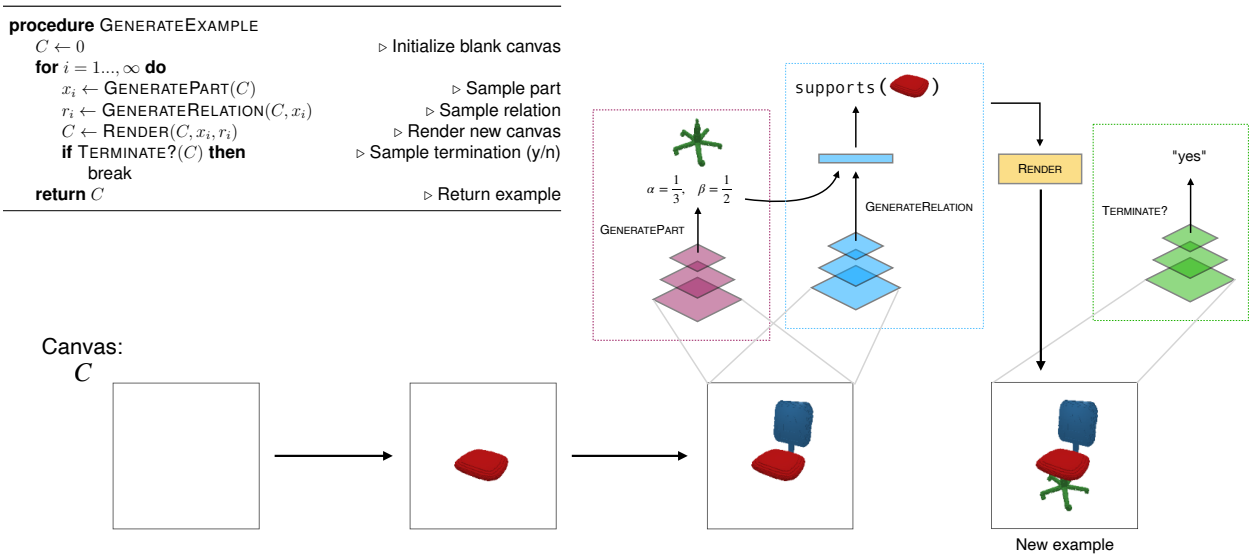


Figure 1.5: A hypothetical Generative Neuro-Symbolic (GNS) model of the concept "chair." GNS represents the concept as a probabilistic program for generating new examples, shown here as the procedure `GenerateExample`. New examples are generated part-by-part, using an image canvas C to maintain the sample state and propagate correlations between parts. At each iteration i, the current canvas C is fed to procedure `GeneratePart`, a neural network subroutine that produces a symbolic description of the next part x_i. In this case, x_i parameterizes a superquadratic that conveys the part's 3D shape. Next, the canvas C and part x_i are fed to procedure `GenerateRelation`, a separate neural network that produces a symbolic relation r_i for how part i relates to other entities in the sample. A symbolic renderer then processes primitives x_i, r_i and yields an updated canvas that contains the new part. Finally, procedure `Terminate?` reads the updated canvas with another neural network and samples a binary decision about whether to terminate the object or continue with another part.

A GNS model represents concepts as probabilistic programs with neural network subroutines (Fig. 1.5). As with traditional probabilistic programs, the control flow of a GNS program is an explicit representation of the *causal* generative process that produces new concepts and new examples.

Moreover, the modularity provided by repeated calls to procedures such as `GeneratePart` ensures a representation that is *compositional*, offering an appropriate inductive bias for combinatorial generalization. Unlike fully-symbolic probabilistic programs, however, the distribution of parts and correlations between parts in GNS are modeled with neural networks. This architectural choice allows the model to learn directly from raw data, capturing nonparametric statistics while requiring only minimal prior knowledge.

The control flow of a GNS program is depicted by the psuedocode in Fig. 1.5. An external image canvas C is used to maintain the state of the sample, providing a controlled memory for the program. Concepts are generated part-by-part by iteratively sampling the next part conditioned on the current partial-sample using neural network subroutines. Rather than generate raw data such as image pixels, GNS subroutines generate a symbolic representation of the next part, and a symbolic relation that describes how it should relate to the existing canvas. These symbolic primitives are translated by a rendering engine that produces an updated partial-object canvas with the new part.

This thesis describes a series of studies that test the GNS modeling framework in different contexts. We hypothesize that GNS models will provide a more comprehensive account of human concept learning in a variety of domains, offering improvements over models with both purely-symbolic and purely-neural architecture.

1.4 Contents of the thesis

Each of the remaining chapters of this thesis is based on a unique publication (excluding Chapter 7, a new conclusion). I begin each chapter with a preface that references the associated publication, provides additional context about the paper and discusses how it fits into the broader thesis narrative.

Chapters 2 & 3 describe an instance of the generative neuro-symbolic (GNS) modeling framework developed for the *Omniglot challenge* (Lake et al., 2015) of task-general representation learning with handwritten character concepts. The computational experiments are accompanied by

a rich human behavioral dataset collected in a previous study (Lake et al., 2015). Chapter 2 first develops a generative neuro-symbolic model to represent a prior over characters in general, used for generating and evaluating new character concepts unconditioned on any particular category. Chapter 3 then expands this prior into a full hierarchical model, adding a differentiable image renderer and developing procedures for approximate probabilistic inference from image data. Together these ingredients enable our model to perform four distinct concept learning tasks that people perform.

Chapter 4 investigates another instance of the GNS approach designed to model human few-shot learning of synthetic “alien figure" concepts. In contrast with handwritten characters, where the variability between different examples of a concept is limited to simple motor and affine noise, each alien figure concept represents a larger class of stimuli with more inter-token variability. This chapter is a collaborative work with another Ph.D. student, Yanli Zhou. Yanli’s contribution is to design the human behavioral experiment and developed a preliminary computational model based on Bayesian inference. My own thesis contribution is to develop a GNS model that accounts for the rich and unique ways that human participants generalize from examples in the behavioral experiment. The breakdown of contributions is discussed further in Section 4.1.

Chapters 5 and 6 explore other ways that inductive biases can be learned by and imposed upon neural network models. These efforts constitute another attempt to unify structured and statistical representation, although the approach diverges slightly from the preceding chapters. Unlike GNS models, which impose a strong inductive bias through causal generative modeling and symbolic program execution, the models presented in these chapters consider weaker inductive biases that manifest as parameter priors and learned similarity metrics. These models lie closer to "neuro" on the neuro-symbolic modeling spectrum, incorporating only minimal symbolic structures through notions of object shape and perceptual smoothness.

Finally, Chapter 7 ends with some concluding remarks and a discussion of future directions for the GNS modeling framework. Some of these directions include preliminary results from a GNS model of simplified object concepts. These results are unpublished and are not yet ready for

submission, but they help set a path towards achieving outstanding research objectives.

Chapter 2

Generating new concepts with neuro-symbolic models

2.1 Preface

This chapter represents the first of a two-part study on handwritten character concepts from the Omniglot dataset (Lake et al., 2015). The focus of this first study was to develop a character prior–i.e., a generative model of handwritten characters *in general*–which follows the GNS formula. By studying the character prior in isolation, we are able to rigorously test the GNS framework by focusing on generative models of handwriting and reserving image representation for later work. We compare our hybrid neuro-symbolic generative model against alternative models at different parts of the neuro/symbolic spectrum, using canonical evaluations including log-likelihood and sample comparison. Importantly, the alternative models are selected to parametrically vary in their level of inductive bias and architectural constraint. In Chapter 3, we later use this model as the type level of a type-token hierarchical Bayesian model that performs four distinct conceptual tasks.

The work of this chapter was published in Feinman & Lake (2020). Before its publication,

generative neural network models applied to Omniglot were not compositional and did not directly model the causal drawing process, and they were shown to produce new characters in un-human ways. At the same time, the symbolic Bayesian Program Learning (BPL) model (Lake et al., 2015)–which is fit directly with human drawing data–makes many simplifying assumptions and does not adequately capture the rich correlation structure in human handwriting. Our interest with this study was to develop a new generative model of characters that models causal handwriting data directly, maintains an explicit notion of parts, and effectively captures the complex correlations present in human drawings.

We developed a handwritten character prior that follows the GNS modeling formula: concepts are represented as probabilistic programs with neural network subroutines, and symbolic primitives and renderers are used to iteratively update the state of a sample. Using this formula, along with a powerful and recent paradigm for generating handwriting known as *mixture networks* (Graves, 2013), we achieve an improved likelihood account of human-drawn character concepts compared to alternative models. We also find that our model produces new character samples that are highly consistent with human drawings, yet sufficiently distinct from their nearest training example.

2.2 Introduction

People can synthesize new concepts in imaginative ways; architects design new houses, chefs invent new recipes, and entrepreneurs create new business models. The resulting productions exhibit novel variations but maintain important structural consistencies with known entities (Ward, 1994). In contrast, state-of-the-art generative models from machine learning struggle with creative imagination, producing samples that either closely mimic the training data or that exhibit anomalous characteristics (Lake et al., 2019). How do people create novel yet coherent new concepts? How can we understand these abilities in computational terms?

Human conceptual knowledge plays a central role in creative generalization. A chef knows not

only a repertoire of recipes, but also understands that recipes are built from *reusable* ingredients (e.g. carrots, flour, butter), and that these ingredients satisfy specific roles (thickening, seasoning, greasing). Furthermore, a chef understands which ingredients can substitute for others (e.g. butter for oil when greasing) and which should never be combined (e.g. ketchup and milk). In addition, they understand that recipes are composed of reusable causal procedures (cutting, whisking, browning), and they know how to compose these procedures in terms of order and substitutability. This *causal* and *compositional* knowledge is essential to understanding a culinary concept, as opposed to merely executing it, and is essential to a chef's ability to create new culinary concepts such as "carrots tartar" or "pea guacamole."

There have been two traditions of work on computational models of conceptual knowledge. The first tradition emphasizes "structured knowledge" for capturing relations between concepts and correlations between conceptual features, viewing concepts as embedded in intuitive theories (Murphy & Medin, 1985) or capturing structured knowledge through symbolic representations such as hierarchies, trees, grammars and programs (Kemp & Tenenbaum, 2008, 2009; Tenenbaum et al., 2011). This tradition has prioritized the compositional and causal nature of conceptual knowledge, as emphasized through accounts of concept learning as program induction (Goodman et al., 2008; Stuhlmuller et al., 2010; Lake et al., 2015; Goodman et al., 2015; Ellis et al., 2018; Lake & Piantadosi, 2020). The Bayesian Program Learning (BPL) framework (Lake et al., 2015), for example, demonstrates how to learn programs from images to express the causal and compositional nature of concepts and background knowledge. Although these models offer a convincing account for how strong inductive biases support flexible generalization, they often make simplifying and rigid parametric assumptions about the distributions of concepts in pursuit of a structured representation. As a result, they so far have been unsuccessful in characterizing the most complex correlations and invariances associated with human concepts in raw, high-dimensional stimulus spaces.

The second tradition in models of conceptual knowledge emphasizes "statistical knowledge," a more amorphous form of background knowledge that is often not amenable to symbolic descrip-

tion. In the statistics view, conceptual knowledge manifests as complex systems of patterns and correlations recorded from observations. The meaning of a word, for example, can be derived from its patterns of co-occurrance with other words (Deerwester et al., 1990). Similarly, latent representations of objects and other sensory stimuli can be derived from "suspicious coincidences" noted in the data (Barlow, 1989). The statistics view emphasizes emergence, where conceptual knowledge emerges from the interaction of simpler processes, as operationalized through training neural network architectures (McClelland, 2010). Although a powerful modeling tool, standard neural networks do not explicitly model the compositional and causal structure of concepts. As result, they have difficulty generalizing to examples that vary systematically from training (Marcus, 2003; Lake & Baroni, 2018), and to novel tasks, especially those that demand more generative and creative abilities (Lake et al., 2017, 2019).

Our goal in this paper is to explore generative models of concepts at the interface of these structured and statistical traditions, with the aim of combining strengths from both approaches. Previous efforts to integrate these traditions have demonstrated ways of performing statistical inference over structured representations (Tenenbaum et al., 2011). This includes models of concept learning as Bayesian inference over fully-symbolic expressions in formal logical (Goodman et al., 2008; Piantadosi et al., 2016), or models of inductive reasoning supported by structured intuitive theories (Kemp & Tenenbaum, 2009). In accounts of this nature, statistics is primary in selecting between structured symbolic hypotheses (Kemp & Tenenbaum, 2008; Perfors et al., 2011; Lake et al., 2015; Lake & Piantadosi, 2020), but plays little role in forming the individual hypotheses themselves. Specifically, each hypothesis may only have a few parametric distributions that need to be inferred (Gaussians, multinomials, etc.), if any.

Here we aim to more thoroughly integrate the structured and statistical traditions through hybrid neuro-symbolic generative models. Our goal is to devise a causal generative model with explicit compositional structure, and with complex correlations represented implicitly through neural networks rather than simple parametric distributions. We use simple visual concepts – handwritten

characters from the world's languages – as a case study for exploring neuro-symbolic models of concept generation. The Omniglot dataset (Lake et al., 2015) of handwritten characters provides an excellent preliminary modeling environment: it contains a large number of natural, simple concepts that people learn and use, and it has been explored extensively in prior work from both cognitive science and AI. Following the mixture density network framework for handwriting generation (Graves, 2013), we explore three distinct generative neural architectures, varying the strength and form of inductive bias imposed on the model, including their position on the neuro-symbolic spectrum and the fidelity in which compositionality and causality are presented. We evaluate the generalization capacity of these models by comparing their log-likelihoods on a holdout set of characters. Furthermore, we analyze the samples produced by each model, looking for characters that are qualitatively consistent but sufficiently dissimilar from the training set. We find that a hybrid neuro-symbolic architecture with the strongest form of compositional structure exhibits the best generalization performance, and that it generates characters that are highly consistent with human drawings. In contrast, the generic neural models exhibit weaker performance on the holdout set, and they produce characters that more closely mimic the training examples.

2.3 Related Work

In the machine learning community, there have been a number of works studying generative neural network models for handwritten characters, including DRAW (Gregor et al., 2015), AIR (Eslami et al., 2016) and SPIRAL (Ganin et al., 2018). Although these models learn a procedure to generate new characters, they do not use the human drawing data from Omniglot, and therefore the generative process may not reflect the true causal processes of human character production. Our goal is different in that we aim to model the causal process of human handwriting directly from drawing data.

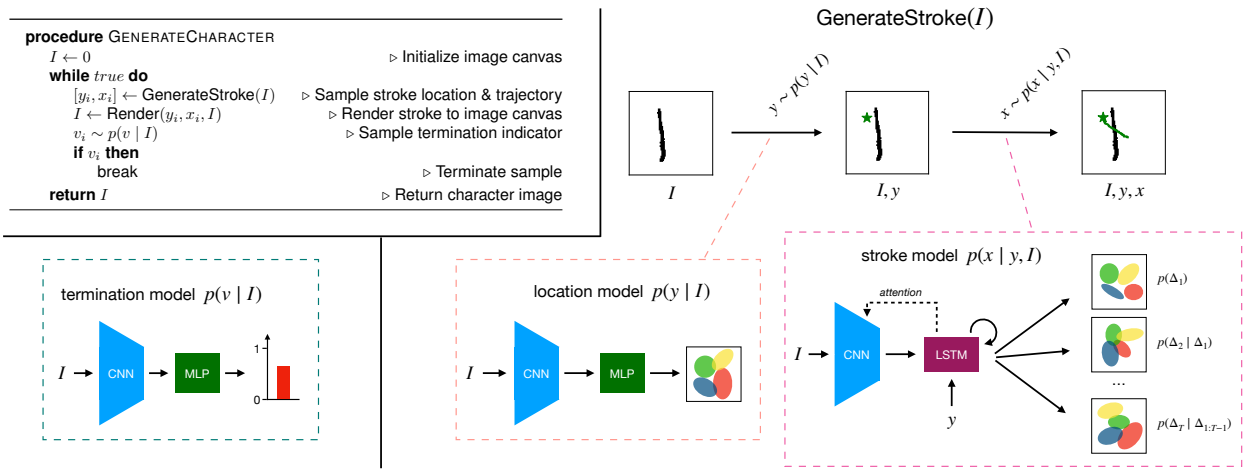


Figure 2.1: Full neuro-symbolic (Full NS) model. Our Full NS model produces character samples one stroke at a time. The procedure `GenerateCharacter` consists of sequentially reading from and rendering to an *image canvas*, which is initialized to zero. At each time step, the current canvas I is fed to procedure `GenerateStroke`, which produces a stroke sample. The canvas is first processed by the *location model*, a CNN-MLP architecture that processes the image and returns a Gaussian mixture model (GMM) distribution for the starting location of the next stroke y. The location y is then sampled and passed along with I to the *stroke model*. The stroke model processes I with a CNN and feeds the embedding to an LSTM with attention. The LSTM samples a stroke trajectory x sequentially one offset at a time using GMM outputs. The sampled stroke is passed to a symbolic renderer, and the updated image canvas is then processed by a *termination model* that decides whether to continue the character sample.

Ha & Eck (2018) introduced a neural network architecture called Sketch-RNN to model human drawing data for simple objects like cats, firetrucks, and windmills. Although their goal loosely resembles our own, the Sketch-RNN model is trained on just a single class of objects at one time (e.g. “cat"), and it receives 70,000 examples from the class. In contrast, our motivation is to model human conceptual knowledge of handwriting concepts in general. This background knowledge plays a central role in creative generalization, enabling people to synthesize new concepts that deviate from the observed entities. We train our models on many character classes at once, providing only 20 training examples of each class and asking them to generate new character concepts. The Sketch-RNN model has not been applied in this way.

Most related to our work is the Bayesian Program Learning (BPL) approach of Lake et al. (2015) that was also applied to the simple visual concepts in Omniglot. BPL is a parametric Bayesian model

that captures causal, compositional structure in human background knowledge of handwriting, and shows that these ingredients are important for few-shot learning of new character concepts. Beyond supporting few-shot learning, the BPL character prior can also generate new character concepts by unconditional sampling. Although a powerful demonstration of compositional representation, the BPL parametric model makes many simplifying assumptions about characters. For example, it assumes that strokes in a character are generated largely independently from each other in the prior (although they are strongly correlated in the posterior). As result, new characters generated by the model often lack the rich correlation structure of human drawings. We build on this work and develop a new neuro-symbolic model that represents the compositional structure of characters while using neural networks to capture richer correlations.

2.4 Omniglot Case Study

We use simple visual concepts as a case study for modeling conceptual structure. The Omniglot dataset contains human drawings of characters from 50 unique alphabets, providing a large set of cognitively natural concepts that are simple enough for evaluating models (Lake et al., 2015, 2019). In our experiments, we use drawings from the Omniglot background set to train our models, which contains 30 alphabets and a total of 19,280 unique drawings. We also use 10 alphabets from the Omniglot evaluation set as a holdout set for quantitative evaluations, reserving the remaining 10 alphabets for future work on few-shot classification.

In the drawing data, a stroke is represented as a variable-length sequence of pen locations $\{z_1, ..., z_T\}$, with $z_i \in \mathbb{R}^2$ (Fig. 2.2, left). During pre-processing, we convert each stroke into a minimal spline representation using least-squares optimization (Fig. 2.2, right), borrowing the B-spline tools from Lake et al. (2015). The number of spline control points depends on the stroke complexity and is determined by a residual threshold. Furthermore, we removed small strokes using a threshold on the trajectory length. These processing steps help suppress noise and emphasize

signal in the drawings. Our generative models are trained to produce character drawings, where each drawing is represented as an ordered set of splines (strokes). The number of strokes, and the number of spline coordinates per stroke, are allowed to vary.

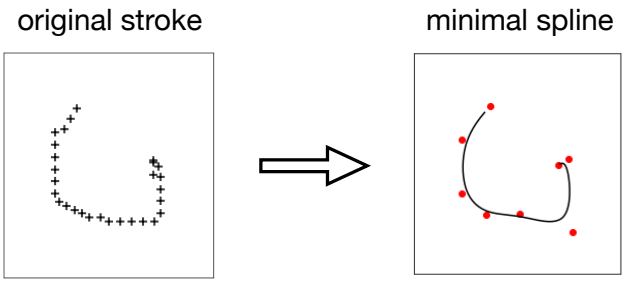


Figure 2.2: Spline representation. Raw strokes (left) are converted into minimal splines (right) using least-squares optimization. Crosses (left) indicate pen locations and red dots (right) indicate spline control points.

2.5 Neuro-Symbolic Model

Our primary interest is to test whether a hybrid neuro-symbolic model can capture the compositional, causal structure in a large corpus of simple natural concepts. The architecture and sampling procedure of our hybrid model, which we call the “Full Neuro-Symbolic" (Full NS) model, is given in Fig. 2.1. Compared to generic neural networks, the Full NS model lies closer to *structure* on the structure-statistics spectrum, possessing a much stronger inductive bias. As in BPL (Lake et al., 2015), the generative model is a probabilistic program that captures real compositional and causal structure by sampling characters as a sequence of parts and locations/relations. Unlike BPL, the model has a symbolic engine that renders each part to an *image canvas* before producing the next one, and parts are generated using a powerful recurrent neural network that encodes and attends to the current canvas. Although correlations between parts can be captured through a process of rendering and then encoding, the model does not allow arbitrary information to flow between parts and variables as in monolithic neural networks.

The Full NS model represents a character as a sequence of strokes, with each stroke decomposed

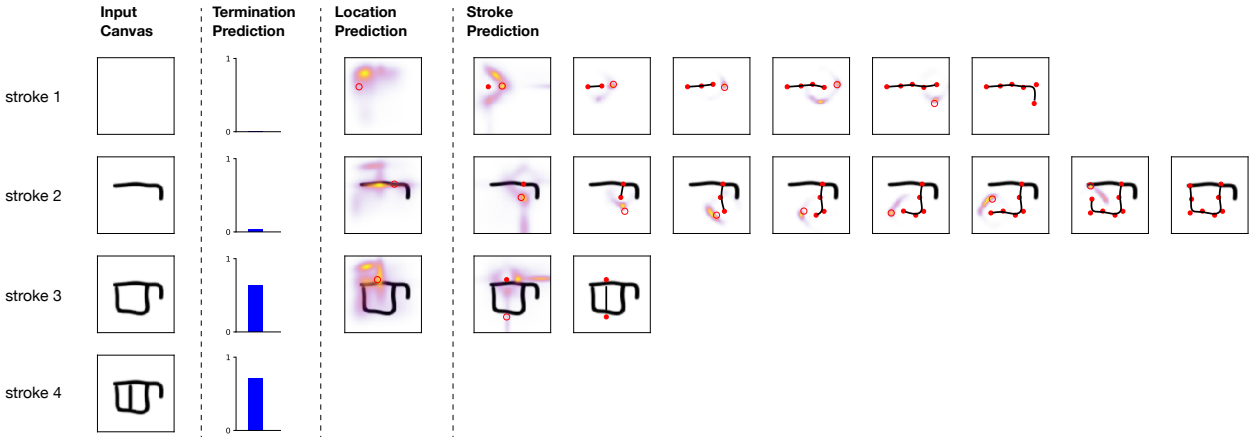


Figure 2.3: Predictions of the Full NS model for a test character. After each stroke, the model receives the current image canvas ("Input Canvas") and makes a series of predictions. **Termination Prediction.** First, the model predicts a termination probability p (blue bar), i.e. a probability of terminating the drawing. **Location Prediction.** Next, the model predicts a probability density for the next stroke's starting location. The heatmap indicates the predicted density, and the hollow red dot indicates the ground-truth location. **Stroke Prediction.** Finally, the model predicts an auto-regressive probability density for the next stroke's trajectory (the "stroke"). Red dots indicate the previous control points, heatmaps indicate the predicted density for the next control point, and hollow red dot indicates the ground-truth next control point.

into a starting location $y_t \in \mathbb{R}^2$, conveying the first spline control point, and a stroke trajectory $x_t = \{\Delta_1, ..., \Delta_N\}$, conveying deltas between spline control points. It generates characters one stroke at a time, using a symbolic rendering procedure called `Render`, as an intermediate processing step after forming each stroke. An image canvas I is used as a memory state to convey information about previous strokes. At each time step t, the next stroke's starting location and trajectory are sampled with procedure `GenerateStroke`. In this procedure, the current image canvas I is first read by the *location model* (Fig. 2.1; bottom middle), a convolutional neural network (CNN) that processes the image and returns a probability distribution for starting location y_t:

$$y_t \sim p(y_t \mid I).$$

A visualization of the density $p(y_t \mid I)$ is given in Fig. 2.3, "Location Prediction." The starting location y_t is then passed along with the image canvas I to the *stroke model* (Fig. 2.1; bottom right),

a Long Short-Term Memory (LSTM) architecture with a CNN-based image attention mechanism inspired by K. Xu et al. (2016). The stroke model samples the next stroke trajectory x_t sequentially one offset at a time, selectively attending to different parts of the image canvas at each sample step and combining this information with the context of y_t:

$$x_t \sim p(x_t \mid y_t, I).$$

A visualization of the auto-regressive density $p(x_t \mid y_t, I)$ is given in Fig. 2.3, "Stroke Prediction." Finally, a similar network decides when to terminate the character, $p(v_t \mid I)$.

Mixture Outputs

Both our location model and stroke model follow a technique from Graves (2013), who proposed to use neural networks with mixture outputs to model handwriting data. The parameters $\theta = \{\pi^{1:K}, \mu^{1:K}, \sigma^{1:K}, \rho^{1:K}\}$ output by our network specify a Gaussian mixture model (GMM) with K components (Fig. 2.1; colored ellipsoids), where $\pi^k \in (0, 1)$ is the mixture weight of the k^{th} component, $\mu^k \in \mathbb{R}^2$ its means, $\sigma^k \in \mathbb{R}^2_+$ its standard deviations, and $\rho^k \in (-1, 1)$ its correlation. In our location model, a single GMM describes the distribution $p(y_t \mid I)$. In our stroke model, the LSTM outputs one GMM at each timestep, describing $p(\Delta_t | \Delta_{1:t-1}, y_t, I)$.

Training

Our Full NS model provides a density function which can be used to score the log-likelihood for any character drawing. We train the model to maximize the log-likelihood (minimize log-loss) of the training set drawings, using mini-batch gradient descent with a batch size of 200 and the Adam update rule.

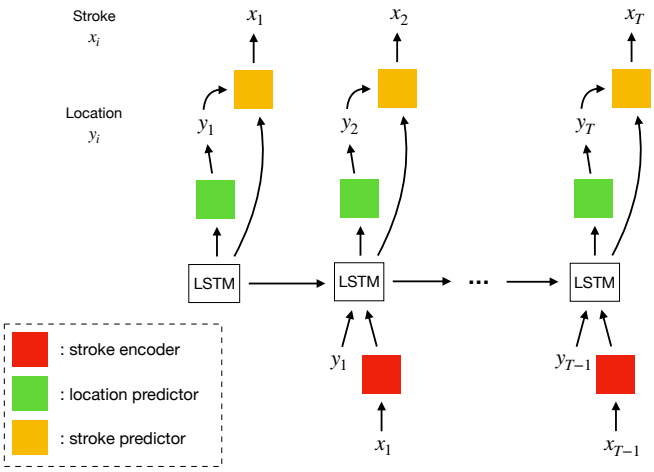


Figure 2.4: Hierarchical LSTM model. The model samples characters one stroke at a time, using a character-level LSTM as a memory state. At each time, the model samples a starting location for the next stroke from a location predictor (MLP), and a stroke trajectory from the stroke predictor (LSTM). These samples are then fed to the model as inputs for the next time, with the location fed directly and the trajectory processed by a stroke encoder (bi-directional LSTM).

2.6 Alternative Models

In addition to our Full NS model, we explored two alternative models with more generic neural network architectures. In each alternative, we lesioned key structural ingredients of the Full NS model, hoping to test the importance of these ingredients to model performance.

Hierarchical LSTM

As one alternative neural model, we explored a hierarchical recurrent architecture (Sordoni et al., 2015; Ling et al., 2016; Chung et al., 2017), which we denote “Hierarchical LSTM" (H-LSTM). Like our Full NS architecture, the H-LSTM model is trained on causal data demonstrating how people actually produce drawings of characters. In addition, it models the compositional structure of characters by separating them into explicit stroke parts, which defines the hierarchy in the hierarchical LSTM. Unlike our Full NS model, however, the H-LSTM has no renderer and thus lacks any explicit causal knowledge of how motor actions become raw images of inked characters.

Instead, information about the previous strokes is written to memory via recurrent connections and gating mechanisms. These transformations can propagate arbitrary correlations, and they must be learned entirely from the data.

Specifically, at each time step t, the previous stroke x_{t-1} is read by a *stroke encoder* f_{enc}, a bi-directional LSTM that processes the stroke and returns a fixed-length vector (red box in Fig. 2.4). This vector is then passed as an input to the character LSTM along with previous location y_{t-1} and previous hidden state h_{t-1}:

$$h_t = f_{\text{LSTM}}(y_{t-1}, f_{\text{enc}}(x_{t-1}), h_{t-1}).$$

The new hidden state h_t is then fed to the *location model* $p(y_t \mid h_t)$, a multi-layer perceptron that outputs a GMM distribution for the next stroke's starting location y_t (green box in Fig. 2.4). The location is sampled from this distribution and passed as an input along with h_t to the *stroke model* $p(x_t \mid h_t, y_t)$, an LSTM that samples a stroke trajectory one offset at a time with GMM outputs (yellow box in Fig. 2.4):

$$y_t \sim p(y_t \mid h_t)$$
$$x_t \sim p(x_t \mid h_t, y_t).$$

Baseline LSTM

A second alternative is even less structured and represents the most purely *statistical* architecture we examined. For this model, we explored a naive unrolled LSTM, denoted "Baseline." This model is a reproduction of the unconditional version of Sketch-RNN (Ha & Eck, 2018, Sec 3.3). Similar to Full NS and H-LSTM, the Baseline LSTM is trained on causal data demonstrating the process of producing characters; however, the architecture does not explicitly take compositional structure into account. Instead, it uses a single RNN to model a character as one long sequence of pen actions

with stroke breaks.

Following Sketch-RNN, we expand the binary pen state variable $v_t \in \{0, 1\}$ from Graves (2013) to a ternary variable $v_t \in \{0, 1, 2\}$ to handle multi-stroke drawings. Value 0 indicates that we are continuing the current stroke, 1 that we are ending the current stroke and starting a new one, and 2 that we are ending the drawing. The initial hidden and cell states of the LSTM are set to zero, and at each time step t, the previous offset Δ_{t-1}, previous pen state v_{t-1}, and previous hidden state h_{t-1} are fed as inputs to the LSTM, which outputs new hidden state h_t:

$$h_t = f_{\text{LSTM}}(\Delta_{t-1}, v_{t-1}, h_{t-1}).$$

An output layer receives h_t and returns a categorical distribution for next pen state v_t, and a GMM for next offset Δ_t:

$$\theta_v = f_v(h_t), \quad v_t \sim p(v_t \mid \theta_v)$$
$$\theta_\Delta = f_\Delta(h_t), \quad \Delta_t \sim p(\Delta_t \mid \theta_\Delta).$$

2.7 Model Hyperparameters

Here we review the hyperparameters (HPs) used for each of our models, indicating which HPs were fixed and which were tuned. All neural networks with GMM output layers use 20 mixture components.

Full NS. The Full NS model has 3 submodules: a *location* model, a *stroke* model, and a *termination* model. Each submodule uses a distinct CNN, and each receives an image canvas of size (28,28). The *location* and *termination* models–which return outputs for a single time step–each use a feed-forward CNN architecture inspired by Vinyals et al. (2016). The CNNs consist of a stack of 4 blocks, with each block i including a 3x3 convolution with K_i filters, batch normalization,

nonlinear activation f, 2x2 max-pooling, and dropout with rate p. These blocks are followed by a single fully-connected layer with D units, activation f and dropout p. Hyperparameters $\{K_i\}$, f, p and D were selected from tuning. The *stroke* model uses a modified CNN architecture without spatial pooling, designed to convey high-resolution spatial information for visual attention. The CNN returns a feature map of size (64, 14, 14), which is then passed to an LSTM. The LSTM predicts the spline trajectory of the next stroke one offset at a time, attending to different parts of the feature map at each step. The HPs of the CNN were fixed, but the HPs of the LSTM were tuned, including the number of LSTM layers and number of units per layer.

Hierarchical LSTM. The Hierarchical LSTM model has a character-level LSTM backbone and 3 submodules: a *stroke encoder* (BiLSTM), a *location model* (MLP), and a *stroke model* (LSTM). The number of LSTM layers, number of units per layer and dropout rate in the character-level LSTM were selected from tuning, but HPs of all submodules were fixed. The *stroke encoder* is a bidirectional LSTM with a single layer of 256 units. It outputs a fixed-length vector representation of the previous stroke, which is fed to the character LSTM as input. The *location model* is a 2-layer MLP that receives the current hidden state of the character LSTM and outputs a GMM for the next stroke's starting location. The *stroke model* is an LSTM with a single layer of 256 units and outputs a GMM at each time step for the next spline offset.

Baseline LSTM. The Baseline LSTM is a single module. It has L LSTM layers, each with K units and dropout rate p. The values of L, K and p were selected from tuning.

2.8 Experiments

We evaluated the creative generalizations of our 3 models using both quantitative and qualitative analyses. Each of our models estimates a probability density function for characters from training examples. This density function can be used to compute likelihoods for held-out characters and to generate new character samples. A generative model for characters that exhibits creative

generalization should produce high likelihood scores for novel character concepts from held-out classes. In addition, the model should generate new characters that are sufficiently dissimilar from the training examples, but that are structurally consistent with ground truth. In our quantitative analysis, we tested our models for their likelihood performance on novel character classes using a rigorous set of experiments with different train/test splits. In our qualitative analysis, we inspected the character samples, comparing with BPL, ground truth concepts, and nearest neighbors from the training set.

	Alphabet Splits			Character Splits			Holdout
Model	split1	split2	split3	split1	split2	split3	-
Full NS	**13.77**	**14.18**	**17.53**	12.35	**12.59**	12.57	**19.51**
H-LSTM	14.37	14.56	17.71	12.24	12.80	12.51	20.16
Baseline	14.32	14.42	17.71	**12.20**	12.77	**12.39**	19.66

Table 2.1: Test losses from our 3 models. Losses indicate the average negative log-likelihood per test character (lower is better). In our "alphabet splits" task, we divide the background set into train/test splits such that the model must generalize to new characters from novel alphabets. In our "character splits" task, we divide the background set such that the model must generalize to new characters from familiar alphabets. In our "holdout" task, we provide the entire background set for training and use the held-out evaluation set–which contains new characters from novel alphabets–for testing.

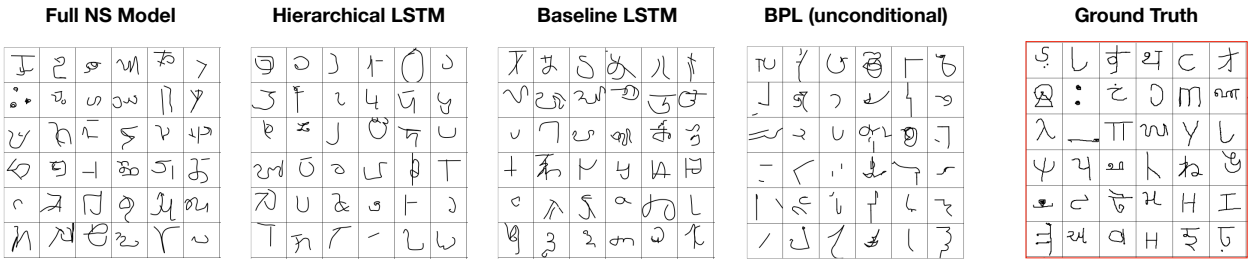


Figure 2.5: Character sample comparison. Characters generated by our Full NS, H-LSTM and Baseline LSTM models are shown side-by-side, along with samples from the BPL forward model[2] as well as ground truth characters from Omniglot.

2.8.1 Evaluation on held-out concepts

Methods

In our quantitative analysis, we evaluated our models for two different forms of likelihood generalization, corresponding to different train/test splits. In the first generalization task, denoted "character splits," we asked whether our models could generalize to new character classes from *familiar alphabets*. We created 3 train/test splits from the Omniglot background set, sampling 80% of characters per alphabet for train and 20% for test. In our second task, denoted "alphabet splits," we asked whether our models could generalize to new character classes from *novel alphabets*. We again sampled 3 train/test splits of size 80-20, this time splitting by alphabet. In both the "character splits" and "alphabet splits" tasks, we explored multiple hyperparameter configurations for our models, varying parameters such as the number of hidden layers, number of units per layer, and dropout probability (Section 2.7). Average validation loss across splits was used to select the best configuration for each model in each task. We then took our best configurations in each task and reported their validation losses on all 3 splits.

As a final quantitative analysis, we tested our models on one additional task that extends the "alphabet splits" task. Our motivation was to provide a more rigorous analysis using a completely withheld test set as per standard practice in machine learning evaluations. We re-trained our best configurations of each model on the entire background set, using the hyperparameters selected from our "alphabet splits" task. We then reported losses on the evaluation set, which contains character drawings from 10 completely novel alphabets.

Results

Results from the cross-validation splits are shown in Table 2.1, "Alphabet Splits" and "Character Splits." In our alphabet splits, the Full NS model consistently outperformed the alternatives, exhibiting the best generalization performance in each of the 3 splits. Thus, our neuro-symbolic

architecture appears best equipped to capture overarching principles in handwriting concepts that generalize far outside of the training examples.

In our character splits task, the Baseline LSTM exhibited best performance in 2 out of 3 splits, and the Full NS model in 1 of 3. The character splits present a much easier generalization task, where exemplar-based learning could offer a suitable alternative to learning general structural principles. Interestingly, the selected hyperparameter configuration for our Full NS model remained constant across the “alphabet" and “character" split tasks, whereas the configuration changed for both the Baseline and H-LSTM models.

Results for each model on the held-out set of characters are shown in Table 2.1, “Holdout." Similarly to the “alphabets" task, our Full NS model outperforms both alternative models on the holdout set, providing further support that this architecture learns the best general model of these simple visual concepts. A paired t-test reveals the Full NS model has reliably better loss per example than the next-best model (Baseline; t(5531) = 3.094; $p < 0.002$).

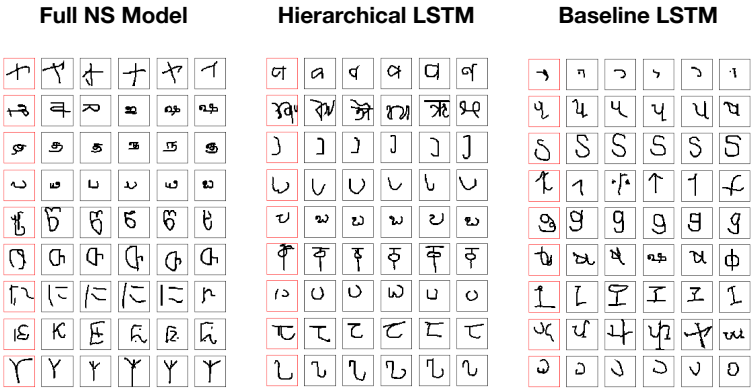


Figure 2.6: Novelty of character samples. Character drawings sampled from each model were compared to their 5 nearest neighbors from the training set. Each row corresponds to one character sample from the model. The red box indicates the model sample, and the 5 nearest neighbors are shown in the succeeding columns.

2.8.2 Generating new concepts

Methods

In our qualitative analysis, we analyzed the 3 neural network models on their ability to produce novel visual concepts. We took our trained models from the previous experiment and sampled 36 characters from each model, following the model's causal generative procedure. In addition, we sampled 36 characters from the BPL character prior, and we selected 36 "ground truth" characters from Omniglot at random. Samples were then compared visually side-by-side.

As an additional qualitative analysis, we compared character samples from each model for their similarity to the training examples. Although the complexity and structural coherence of generated characters are important criteria, these observations alone provide insufficient evidence for a human-like generative process; a model that memorizes the training examples might produce samples with structural coherence and rich variations, but such a model does not account for the flexible ways that humans generate new concepts. In our second analysis, we took the character samples from our models and found the 5 most-similar training characters for each, using cosine distance in the last hidden layer of a CNN classifier as a metric space for perceptual similarity. The CNN was trained to classify characters from the Omniglot background set, a 964-way classification task.

Results

Fig. 2.5 shows samples from each of our three models, as well as from the BPL forward model[1] and from the Omniglot data (ground truth). Compared to BPL, the neural-enhanced models capture more correlational structure and character complexity. For instance, the Full NS model propagates stylistic and structural consistency across three strokes to form a Braille-like character, as shown by

[1] BPL character samples have been centered for better visual appearance; the actual samples often protrude outside of the image window. A more complex non-parametric BPL model was used in the visual Turing tests in Lake et al. (2015) that has explicit re-use of character parts. Those samples were also centered.

the sample in column 1, row 2. Fig. 2.6 shows a handful of character samples produced by each neural model plotted alongside their five nearest neighbors from the Omniglot training set. Unlike the log-likelihood results, comparing models in this fashion is subjective; nevertheless the H-LSTM and Baseline LSTM produce more characters that closely mimic the nearest training examples (7/9 by our eyes). In contrast, our Full NS model produces only a few (3/9) characters that are close mirrors of training examples, suggesting that it can generalize further from the training observations.

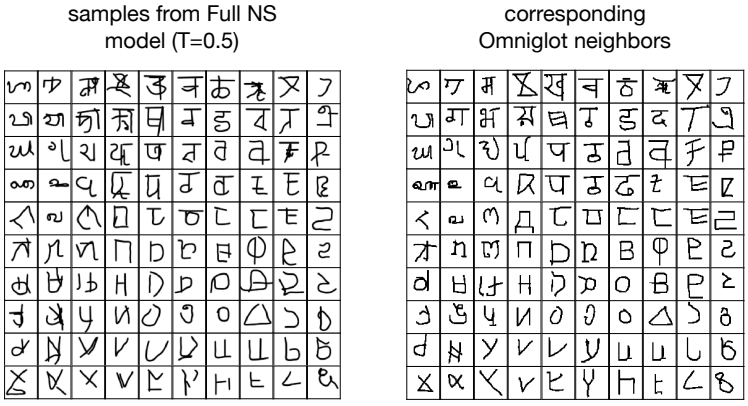


Figure 2.7: Topologically-organized character samples and their nearest Omniglot neighbors. We drew 100 character samples from our Full NS model and organized them into a 10x10 grid such that neighboring characters have similar drawing styles (left). We then found the “nearest neighbor" of each sample from the Omniglot character dataset and organized the neighbors into a corresponding 10x10 grid (right).

To get an idea of the different character styles produced by our Full NS model, we sampled 100 characters from the model and organized them into a 10x10 grid such that neighboring characters have high perceptual similarity (Fig. 2.7, left). Characters were sampled at a lower level of stochasticity, using the temperature parameter proposed by Ha & Eck (2018) to modify the entropy of the mixture density outputs (we used $T = 0.5$). The model produces characters in multiple distinct styles, with some having more angular, line-based structure and others relying on complex curves. In Fig. 2.7 (right), we plotted the most-similar Omniglot character for each sample in a corresponding grid. In many cases, samples from the model have a distinct style and are visually dissimilar from their nearest Omniglot neighbor.

Fig. 2.8 shows a larger collection of characters from our Full NS model, using color coding to convey the stroke composition of each sample. We produced character samples at two different levels of stochasticity, again using temperature to modify the entropy of the mixture density outputs. Samples are shown for temperature settings $T = 1.0$ and $T = 0.5$.

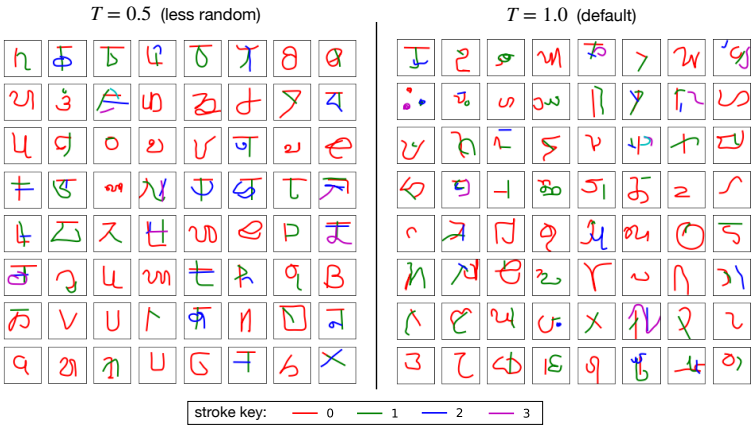


Figure 2.8: Samples with stroke decomposition. Character samples produced by our Full NS model are shown with stroke decompositions. Samples were produced at two temperature settings (Ha & Eck, 2018, Eq.8), using $T = 1.0$ and $T = 0.5$.

2.9 Discussion

We presented a new neuro-symbolic generative model of simple visual concepts. Our model successfully captures compositional and causal structure in handwritten character concepts, forming a representation that generalizes to new concepts. We tested our model by comparing its likelihood scores on a holdout set of novel characters, finding that it consistently outperforms two generic neural network alternatives when the test characters deviate significantly from the training examples. Furthermore, our generative model produces new character concepts with richer variations than simple parametric models, yet that remain structurally coherent and visually consistent with human productions.

Neuro-symbolic models offer a promising set of tools to express the rich background knowledge

that enables creative generation. These models can explain the nonparametric correlation structure embodied in conceptual knowledge while maintaining important inductive biases to account for the structured ways that people generate new concepts. We believe that models of this kind will be useful to explain a variety of human imaginative behaviors, such as when a chef creates the new recipe “pea guacamole." In future work, we’d like to explore applications of neuro-symbolic models to other types of concepts with varying complexity.

Chapter 3

Few-shot learning of handwritten character concepts

3.1 Preface

This chapter is based off of Feinman & Lake (2021). It explores a hierarchical generative model designed to tackle the Omniglot challenge (Lake et al., 2015) of task-general representation learning. The Omniglot challenge was released in 2015 as a challenge for machines to perform 5 concept learning tasks that people perform with ease. At the time we published this work, the challenge had become a popular benchmark and an active area of research in the machine learning community. Despite nearly 4 years of active research, neural network models had not yet explained how people successfully grasp new concepts and use them in a variety of ways. In their "3-year progress report," Lake et al. (2019) detailed the shortcomings of various machine learning models that had attempted the challenge. Although there had been considerable progress in one-shot classification, the majority of works had focused on this single task alone, and there had been little emphasis placed on developing task-general models. Our paper set out to address unanswered questions about

task-general representation learning, aiming to develop the first neurally-grounded model that could perform a variety of distinct Omniglot tasks.

This chapter builds on the work of Chapter 2, and aspects of the model were already introduced therein. The previous work presents only a prior distribution that alone performs just one task (generating new concepts); our new developments in this chapter include a full hierarchical model, a differentiable image renderer / likelihood, and a procedure for approximate probabilistic inference from image data. These ingredients together enable GNS to perform 4 unique conceptual tasks.

3.2 Introduction

Human conceptual knowledge supports many capabilities spanning perception, production and reasoning (Murphy, 2002). A signature of this knowledge is its productivity and generality: the internal models and representations that people develop can be applied flexibly to new tasks with little or no training experience (Lake et al., 2017). Another distinctive characteristic of human conceptual knowledge is the way that it interacts with raw signals: people learn new concepts directly from raw, high-dimensional sensory data, and they identify instances of known concepts embedded in similarly complex stimuli. A central challenge is developing machines with these human-like conceptual capabilities.

Engineering efforts have embraced two distinct paradigms: *symbolic* models for capturing structured knowledge, and *neural network* models for capturing nonparametric statistical relationships. Symbolic models are well-suited for representing the causal and compositional processes behind perceptual observations, providing explanations akin to people's intuitive theories (Murphy & Medin, 1985). Quintessential examples include accounts of concept learning as program induction (Goodman et al., 2008; Stuhlmuller et al., 2010; Lake et al., 2015; Goodman et al., 2015; Ellis et al., 2018; Lake & Piantadosi, 2020). Symbolic programs provide a language for expressing causal and compositional structure, while probabilistic modeling offers a means of learning programs and

expressing additional conceptual knowledge through priors. The Bayesian Program Learning (BPL) framework (Lake et al., 2015), for example, provides a dictionary of simple sub-part primitives for generating handwritten character concepts, and symbolic relations that specify how to combine sub-parts into parts (strokes) and parts into whole character concepts. These abstractions support inductive reasoning and flexible generalization to a range of different tasks, utilizing a single conceptual representation (Lake et al., 2015).

Symbolic models offer many useful features, but they come with important limitations. Foremost, symbolic probabilistic models make simplifying and rigid parametric assumptions, and when the assumptions are wrong—as is common in complex, high-dimensional data—they create bias (Geman et al., 1992). The BPL character model, for example, assumes that parts are largely independent a priori, an assumption that is not reflective of real human-drawn characters. As a consequence, characters generated from the raw BPL prior lack the complexity of real characters (Fig 3.1, left), even though the posterior samples can appear much more structured. Another limitation of symbolic probabilistic models is that the construction of structured hypothesis spaces requires significant domain knowledge (Botvinick et al., 2017). Humans, meanwhile, build rich internal models directly from raw data, forming hypotheses about the conceptual features and the generative syntax of a domain. As one potential resolution, previous work has demonstrated that the selection of structured hypotheses can itself be attributed to learning in a Bayesian framework (Tenenbaum et al., 2011; Goodman et al., 2008, 2011; Piantadosi et al., 2016; Kemp & Tenenbaum, 2009; Perfors et al., 2011). Although more flexible than a priori structural decisions, models of this kind still make many assumptions, and they have not yet tackled the types of raw, high-dimensional stimuli that are distinctive of the neural network approach.

The second paradigm, neural network modeling, prioritizes powerful nonparametric statistical learning over structured representations. This modeling tradition emphasizes *emergence*, the idea that conceptual knowledge arises from interactions of distributed sub-symbolic processes (McClelland et al., 2010; LeCun et al., 2015). Neural networks are adept at learning from raw

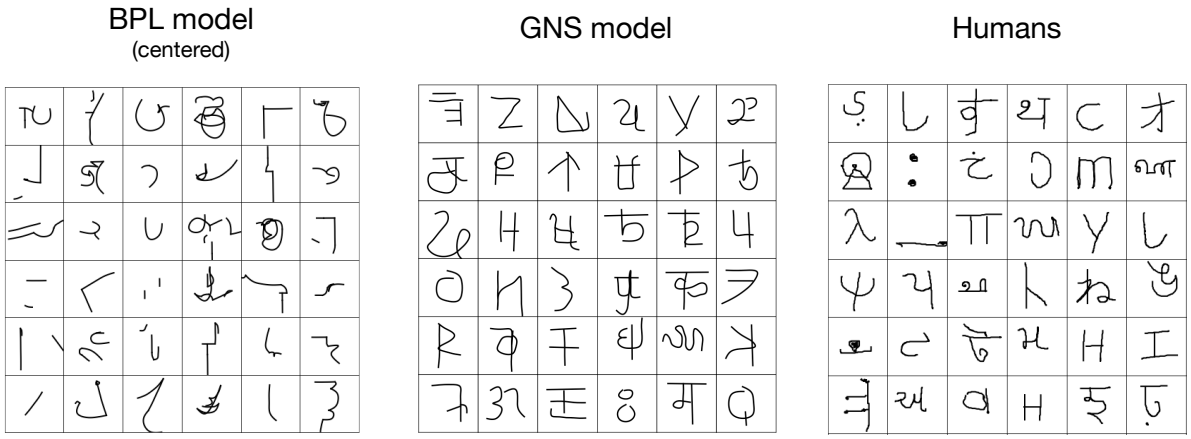


Figure 3.1: Character drawings produced by the BPL model (left), GNS model (middle), and humans (right).

data and capturing complex patterns. However, they can struggle to learn the compositional and causal structure in how concepts are formed (Lake et al., 2017); even when this structure is salient in the data, they may have no obvious means of incorporating it. These limitations have been linked to shortcomings in systematic generalization (Marcus, 2003; Lake & Baroni, 2018) and creative abilities (Lake et al., 2019). An illustrative example is the *Omniglot challenge*: in 4 years of active research, neural network models do not yet explain how people quickly grasp new concepts and use them in a variety of ways, even with relatively simple handwritten characters (Lake et al., 2019). Surveying over 10 neural models applied to Omniglot, Lake et al. (2019) found that only two attempted both classification and generation tasks, and they were each outperformed by the fully-symbolic, probabilistic BPL. Moreover, neural generative models tended to produce characters with anomalous characteristics, highlighting their shortcomings in modeling causal and compositional structure (see Fig. 3.2 and Lake et al. (2019, Fig. 2a)).

In this paper, we introduce a new approach that leverages the strengths of both the symbolic and neural network paradigms by representing concepts as probabilistic programs with neural network subroutines. We describe an instance of this approach developed for the *Omniglot challenge* (Lake et al., 2015) of task-general representation learning and discuss how we see our Omniglot model fitting into a broader class of Generative Neuro-Symbolic (GNS) models that seek to capture

the data-generation process. As with traditional probabilistic programs, the control flow of a GNS program is an explicit representation of the *causal* generative process that produces new concepts and new exemplars. Moreover, explicit re-use of parts through repeated calls to procedures such as `GeneratePart` (Fig. 3.3) ensures a representation that is *compositional*, providing an appropriate inductive bias for compositional generalization. Unlike fully-symbolic probabilistic programs, however, the distribution of parts and correlations between parts in GNS are modeled with neural networks. This architectural choice allows the model to learn directly from raw data, capturing nonparametric statistics while requiring only minimal prior knowledge.

We develop a GNS model for the Omniglot challenge of learning flexible, task-general representations of handwritten characters. We report results on 4 Omniglot challenge tasks with a single model: 1) one-shot classification, 2) parsing/segmentation, 3) generating new exemplars, and 4) generating new concepts (without constraints); the 5th and final task of generating new concepts (from type) is left for future work. We also provide log-likelihood evaluations of the generative model. Notably, our goal is not to chase state-of-the-art performance on one task across many datasets (e.g., classification). Instead we build a model that learns deep, task-general knowledge within a single domain and evaluate it on a range of different tasks. This "deep expertise" is arguably just as important as "broad expertise" in characterizing human-level concept learning (Murphy, 2002; Lake et al., 2019); machines that seek human-like abilities will need both. Our work here is one proposal for how neurally-grounded approaches can move beyond pattern recognition toward the more flexible model-building abilities needed for deep expertise (Lake et al., 2017). Subsequent chapters demonstrate how to extend GNS to another conceptual domain: Chapter 4 provides a complete case study with structured visual concepts, and Section 7.3 presents a proposal for a GNS model of 3D object concepts.

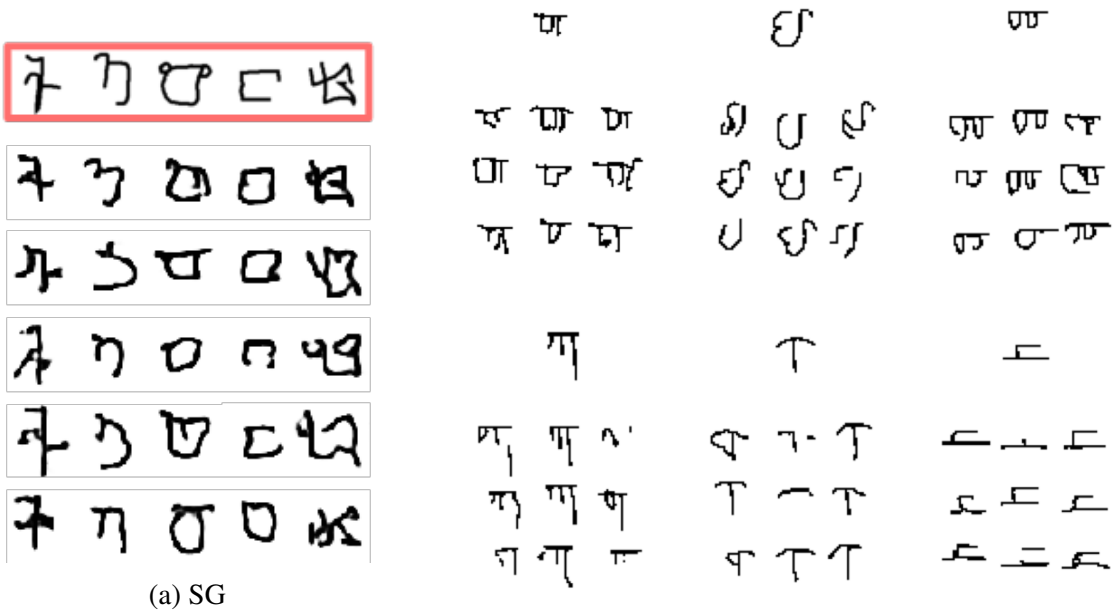
Task	BPL	RCN	VHE	SG	SPIRAL	Matching Net	MAML	Graph Net	Prototypical Net	ARC
One-shot classification	x	x	x			x	x	x	x	x
Parsing	x				x					
Generate exemplars	x	x	x	x						
Generate concepts (type)	x		x	x						
Generate concepts	x			x	x					

Table 3.1: Attempted Omniglot tasks by model. Attempt does not imply successful completion. Models shown: BPL (Lake et al., 2015), RCN (George et al., 2017), VHE (Hewitt et al., 2018), SG (Rezende et al., 2016), SPIRAL (Ganin et al., 2018), Matching Net (Vinyals et al., 2016), MAML (Finn et al., 2017), Graph Net (Garcia & Bruna, 2018), Prototypical Net (Snell et al., 2017), and ARC (Shyam et al., 2017).

3.3 Related Work

The Omniglot dataset and challenge has been widely adopted in machine learning, with models such as Matching Nets (Vinyals et al., 2016), MAML (Finn et al., 2017), and ARC (Shyam et al., 2017) applied to just one-shot classification, and others such as DRAW (Gregor et al., 2015), SPIRAL (Ganin et al., 2018), and VHE (Hewitt et al., 2018) applied to one or more generative tasks. In their "3-year progress report," Lake et al. (2019) reviewed the current progress on Omniglot, finding that although there was considerable progress in one-shot classification, there had been little emphasis placed on developing task-general models to match the flexibility of human learners (Table 3.1). Moreover, neurally-grounded models that attempt more creative generation tasks were shown to produce characters that either closely mimicked the training examples or that exhibited anomalous variations, making for easy identification from humans (see Fig. 3.2 and Lake et al. (2019, Fig. 2a)). Our goal is distinct in that we aim to learn a single neuro-symbolic generative model that can perform a variety of unique tasks, and that generates novel yet structured new characters.

Neuro-symbolic modeling has become an active area of research, with applications to learning input-output programs (Reed & de Freitas, 2016; Graves et al., 2014; Devlin et al., 2017; Nye et al., 2020; Valkov et al., 2018), question answering (Yi et al., 2018; Mao et al., 2019) and image description (Kulkarni & et al., 2015; Ellis et al., 2018). GNS modeling distinguishes itself from



(a) SG

(b) VHE

Figure 3.2: New exemplars produced by the Sequential Generative (SG) model (Rezende et al., 2016) and the Variational Homoencoder (VHE) (Hewitt et al., 2018). (a) The SG model shows far too much variability, drawing what is clearly the wrong character in many cases (e.g. right-most column). (b) The VHE character samples are often incomplete, missing important strokes of the target class.

prior work through its focus on hybrid generative modeling, combining both structured program execution and neural networks directly in the probabilistic generative process. Neuro-symbolic VQA models (Yi et al., 2018; Mao et al., 2019) are neither generative nor task-general; they are trained discriminatively to answer questions. Other neuro-symbolic systems use neural networks to help perform inference in a fully-symbolic generative model (Kulkarni & et al., 2015; Ellis et al., 2018), or to parameterize a prior over fully-symbolic hypotheses (Hewitt et al., 2020). In order to capture the dual structural and statistical characteristics of human conceptual representations, we find it important to include neural nets directly in the forward generative model. As applied to Omniglot, our model bears some resemblance to SPIRAL (Ganin et al., 2018); however, SPIRAL does not provide a density function, and it has no hierarchical structure, limiting its applications to image reconstruction and unconditional generation.

Another class of models on the neuro-symbolic spectrum aims to learn “object representations"

with neural networks (Eslami et al., 2016; Greff et al., 2019; Kosiorek et al., 2019), which add minimal object-like symbols to support systematic reasoning and generalization. Although these models have demonstrated promising results in applications such as scene segmentation and unconditional generation, they have not yet demonstrated the type of rich inductive capabilities that we are after: namely, the ability to learn "deep" conceptual representations from just one or a few examples that support a variety of discriminative and generative tasks.

Other works have used autoregressive models like ours with similar stroke primitives to model the causal generative processes of handwriting (Graves, 2013; Ha & Eck, 2018). We develop a novel architecture for generating handwriting, which represents explicit compositional structure by modeling parts and relations with separate modules and applying intermediate symbolic rendering. Most importantly, these prior models have not made a connection to the image; therefore while they can generate handwriting as symbolic coordinates, they cannot explain how people use their causal knowledge to learn new characters from visual presentations, how they infer the strokes of a character seen on paper, or how they generate a new example of an observed character. By combining a powerful autoregressive model of handwriting with a symbolic image renderer and algorithms for probabilistic inference, we seek to replicate a spectrum of unique human concept learning abilities.

3.4 Generative Model

Our GNS model leverages the type-token hierarchy of BPL (Lake et al., 2015), which offers a useful scaffolding for conceptual models (Fig. 3.3, left). The type-level model $P(\psi)$ defines a prior distribution over character concepts, capturing overarching principles and regularities that tie together characters from different alphabets and providing a procedure to generate new character concepts unconditionally in latent stroke format (Fig. 3.3, right). A token-level model $p(\theta|\psi)$ captures the within-class variability that arises from motor noise and drawing styles, and an image

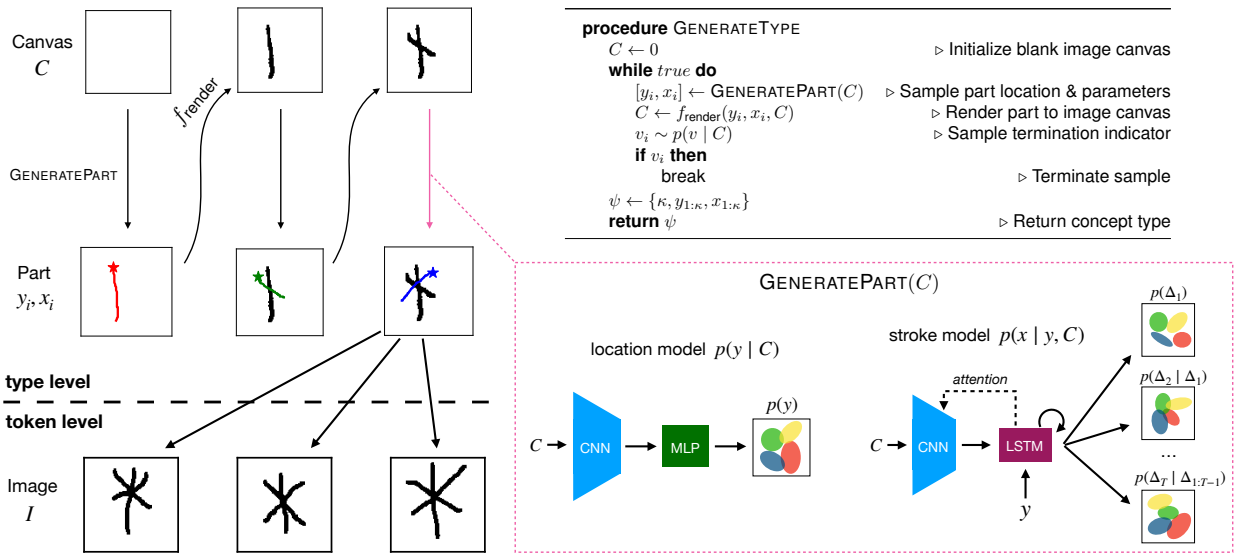


Figure 3.3: A generative neuro-symbolic (GNS) model of character concepts. The type model `GenerateType` ($P(\psi)$) produces character types one stroke at a time, using an image canvas C as memory. At each step, the current canvas C is fed to procedure `GeneratePart` and a stroke sample is produced. The canvas is first processed by the *location model*, a CNN-MLP architecture that samples starting location y, and next by the *stroke model*, a CNN-LSTM architecture that samples trajectory x while attending to the encoded canvas. Finally, a symbolic renderer updates the canvas according to x and y, and a *termination model* decides whether to terminate the type sample. Unique exemplars are produced from a character type by sampling from the token model conditioned on ψ, adding motor noise to the drawing parameters and performing a random affine transformation.

distribution $P(I|\theta)$ provides an explicit model of how causal stroke actions translate to image pixels. All parameters of our model are learned from the Omniglot background set of drawings (Appendix A). The full joint distribution over type ψ, token $\theta^{(m)}$ and image $I^{(m)}$ factors as

$$P(\psi, \theta^{(m)}, I^{(m)}) = P(\psi)P(\theta^{(m)}|\psi)P(I^{(m)}|\theta^{(m)}). \tag{3.1}$$

Although sharing a common hierarchy, the implementation details of each level in our GNS model differ from BPL in critical ways. The GNS type prior $P(\psi)$ is a highly expressive generative model that uses an external image canvas, coupled with a symbolic rendering engine and an attentive recurrent neural network, to condition future parts on the previous parts and model sophisticated

causal and correlational structure. This structure is essential to generating new character concepts in realistic, human-like ways (Sec. 3.6). Moreover, whereas the BPL model is provided symbolic relations for strokes such as “attach start" and “attach along," GNS learns implicit relational structure from the data, identifying salient patterns in the co-occurrences of parts and locations. Importantly, the GNS generative model is designed to be differentiable at all levels, yielding log-likelihood gradients that enable powerful new inference algorithms and estimates of marginal image likelihood (Sec. 3.5).

3.4.1 Type prior

The type prior $P(\psi)$ is captured by a neuro-symbolic generative model of character drawings. The model represents a character as a sequence of strokes (parts), with each stroke i decomposed into a starting location $y_i \in \mathbb{R}^2$ and a variable-length trajectory $x_i \in \mathbb{R}^{d_i \times 2}$. Rather than use raw pen trajectories as our stroke format, we use a *minimal spline* representation of strokes, obtained from raw trajectories by fitting cubic b-splines with a residual threshold. The starting location y_i therefore conveys the first spline control point, and trajectory $x_i = \{\Delta_{i1}, ..., \Delta_{id_i}\}$ conveys the offsets between subsequent points of a (d_i+1)-length spline. These offsets are transformed into a sequence of relative points $x_i = \{x_{i1}, ..., x_{id_i+1}\}$, with $x_{i1} = 0$, specifying locations relative to y_i.

The model samples a type one stroke at a time, using an image canvas C as memory to convey the sample state. At each step, a starting location for the next stroke is first sampled from the *location model*, followed by a trajectory from the *stroke model*. The stroke is then rendered to the canvas C, and a *termination model* decides whether to terminate or continue the sample. Each of the three model components is expressed by a neural network, using a LSTM as the stroke model to generate trajectories as in Graves (2013). The details of these neural modules are provided in Appendix A. The type model $P(\psi)$ specifies an auto-regressive density function that can evaluate exact likelihoods of character drawings, and its hyperparameters (the three neural networks) are learned from the Omniglot background set of 30 alphabets using a maximum likelihood objective.

A full character type ψ includes the random variables $\psi = \{\kappa, y_{1:\kappa}, x_{1:\kappa}\}$, where $\kappa \in \mathbb{Z}^+$ is the number of strokes. The density function $P(\psi)$ is also fully differentiable w.r.t. the continuous random variables in ψ.

3.4.2 Token model

A character token $\theta^{(m)} = \{y^{(m)}_{1:\kappa}, x^{(m)}_{1:\kappa}, A^{(m)}\}$ represents a unique instance of a character concept, where $y^{(m)}_{1:\kappa}$ are the token-level locations, $x^{(m)}_{1:\kappa}$ the token-level parts, and $A^{(m)} \in \mathbb{R}^4$ the parameters of an affine warp transformation. The token distribution factorizes as

$$P(\theta^{(m)}|\psi) = P(A^{(m)}) \prod_{i=1}^{\kappa} P(y_i^{(m)} \mid y_i) P(x_i^{(m)} \mid x_i). \tag{3.2}$$

Here, $P(y_i^{(m)} \mid y_i)$ represents a simple noise distribution for the location of each stroke, and $P(x_i^{(m)} \mid x_i)$ for the stroke trajectory. The first two dimensions of affine warp $A^{(m)}$ control a global re-scaling of the token drawing, and the second two a global translation of its center of mass. The distributions and pseudocode of our token model are given in Appendix A.4.

3.4.3 Image model

The image model $P(I^{(m)} \mid \theta^{(m)})$ is based on Lake et al. (2015) and is composed of two pieces. First, a differentiable symbolic engine f receives the token $\theta^{(m)}$ and produces an image pixel probability map $p_{\text{img}} = f(\theta^{(m)}, \sigma, \epsilon)$ by evaluating each spline and rendering the stroke trajectories. Here, $\sigma \in \mathbb{R}^+$ is a parameter controlling the rendering blur around stroke coordinates, and $\epsilon \in (0, 1)$ controlling pixel noise, each sampled uniformly at random. The result then parameterizes an image distribution $P(I^{(m)} \mid \theta^{(m)}) = \text{Bernoulli}(p_{\text{img}})$, which is differentiable w.r.t. $\theta^{(m)}$, σ, and ϵ.

3.5 Probabilistic Inference

Given an image I of a novel concept, our GNS model aims to infer the latent causal, compositional process for generating new exemplars. We follow the high-level strategy of BPL for constructing a discrete approximation $Q(\psi, \theta \mid I)$ to the desired posterior distribution (Lake et al., 2015),

$$P(\psi, \theta \mid I) \approx Q(\psi, \theta \mid I) = \sum_{k=1}^{K} \pi_k \delta(\theta - \theta_k)\delta(\psi - \psi_k). \tag{3.3}$$

A heuristic search algorithm is used to find K good parses, $\{\psi, \theta\}_{1:K}$, that explain the underlying image with high probability. These parses are weighted by their relative posterior probability, $\pi_k \propto \tilde{\pi}_k = P(\psi_k, \theta_k, I)$ such that $\sum_k \pi_k = 1$. To find the K good parses, search uses fast bottom-up methods to propose many variants of the discrete variables, filtering the most promising options, before optimizing the continuous variables with gradient descent (we use $K = 5$). The algorithm proceeds by the following steps:

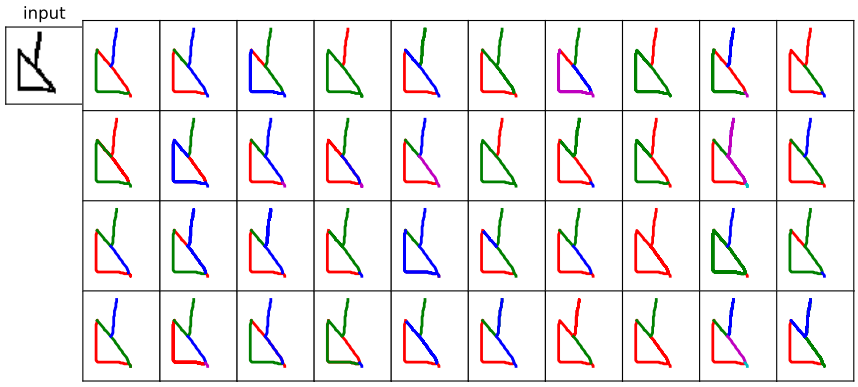


Figure 3.4: The initial “base" parses proposed for an image with skeleton extraction and random walks.

1. Propose a range of candidate parses with fast bottom-up methods. The bottom-up algorithm extracts an undirected skeleton graph from the character image and uses random walks on

the graph to propose a range of candidate parses. There are typically about 10-100 proposal parses, depending on character complexity (Fig. 3.4).

2. Select stroke order and stroke directions for each parse using exhaustive search with the type prior $P(\psi)$. Random search is used for complex parses with large configuration spaces.

3. Score each of the proposal parses using type prior $P(\psi)$ and select the top-K parses. We use $K = 5$ following previous work (Lake et al., 2015).

4. Separate each parse into type and token $\{\psi, \theta\}$ and optimize the continuous type- and token-level parameters with gradient descent to maximize the full joint density $P(\psi, \theta, I)$ of Eq. 3.1.

5. Compute weights $\pi_{1:K}$ for each parse by computing $\tilde{\pi}_k = P(\psi_k, \theta_k, I)$ and normalizing $\pi_k = \tilde{\pi}_k / \sum_{k=1}^{K} \tilde{\pi}_k$.

3.5.1 Inference for one-shot classification

In one-shot classification, models are given a single training image $I^{(c)}$ from each of $c = 1, ..., C$ classes, and asked to classify test images according to their corresponding training classes. For each test image $I^{(T)}$, we compute an approximation of the Bayesian score $\log P(I^{(T)} \mid I^{(c)})$ for every example $I^{(c)}$, using our posterior parses $\{\psi, \theta^{(c)}\}_{1:K}$ and corresponding weights $\pi_{1:K}$ from $I^{(c)}$ (Eq. 3.3). The approximation is formulated as

$$\begin{aligned} \log P(I^{(T)} \mid I^{(c)}) &\approx \log \int P(I^{(T)} | \theta^{(T)}) P(\theta^{(T)} \mid \psi) Q(\psi, \theta^{(c)}, \mid I^{(c)}) \partial\psi \partial\theta^{(c)} \partial\theta^{(T)} \\ &\approx \log \sum_{k=1}^{K} \pi_k \max_{\theta^{(T)}} P(I^{(T)} \mid \theta^{(T)}) P(\theta^{(T)} \mid \psi_k), \end{aligned} \tag{3.4}$$

where the maximum over $\theta^{(T)}$ is determined by refitting token-level parameters $\theta^{(c)}$ to image $I^{(T)}$ with gradient descent. Following Lake et al. (2015), we use a two-way version of the Bayesian

score that also considers parses of $I^{(T)}$ refit to $I^{(c)}$. The classification rule is therefore

$$c^* = \arg\max_c \log P(I^{(T)} \mid I^{(c)})^2 = \arg\max_c \log \big[\frac{P(I^{(c)} \mid I^{(T)})}{P(I^{(c)})} P(I^{(T)} \mid I^{(c)})\big], \tag{3.5}$$

where $P(I^{(c)}) \approx \sum_k \tilde{\pi}_k$ is approximated from the unnormalized weights of $I^{(c)}$ parses.

3.5.2 Inference for generating new exemplars

When generating new exemplars, we are given a single image $I^{(1)}$ of a novel class and asked to generate new instances $I^{(2)}$ (overloading the parenthesis notation from classification). To perform this task with GNS, we first sample from our approximate posterior $Q(\psi, \theta \mid I^{(1)})$ to obtain parse $\{\psi, \theta\}$ (see Eq. 3.3), and then re-sample token parameters θ from our token model $P(\theta^{(2)} \mid \psi)$. Due to high-dimensional images, mass in the approximate posterior often concentrates on the single best parse. To model the diversity seen in different human parses, we apply a temperature parameter to the log of unnormalized parse weights $\log(\tilde{\pi}'_k) = \log(\tilde{\pi}_k)/T$ before normalization, selecting $T = 8$ for our experiments. With updated weights $\pi'_{1:K}$ our sampling distribution is written as

$$P(I^{(2)}, \theta^{(2)} \mid I^{(1)}) \approx \sum_{k=1}^{K} \pi'_k P(I^{(2)} \mid \theta^{(2)}) P(\theta^{(2)} \mid \psi_k). \tag{3.6}$$

3.5.3 Inference for marginal image likelihoods

Let $z = \psi \cup \theta$ be a stand-in for the joint set of type- and token-level random variables in our GNS generative model. The latent z includes both continuous and discrete variables: the number of strokes κ and the number of control points per stroke $d_{1:\kappa}$ are discrete, and all remaining variables are continuous. Decomposing z into its discrete variables $z_D \in \Omega_D$ and continuous variables

$z_C \in \Omega_C$, the marginal density for an image I is written as

$$P(I) = \sum_{z_D \in \Omega_D} \int P(I, z_D, z_C) \partial z_C. \tag{3.7}$$

For any subset $\tilde{\Omega}_D \subset \Omega_D$ of the discrete domain, the following inequality holds:

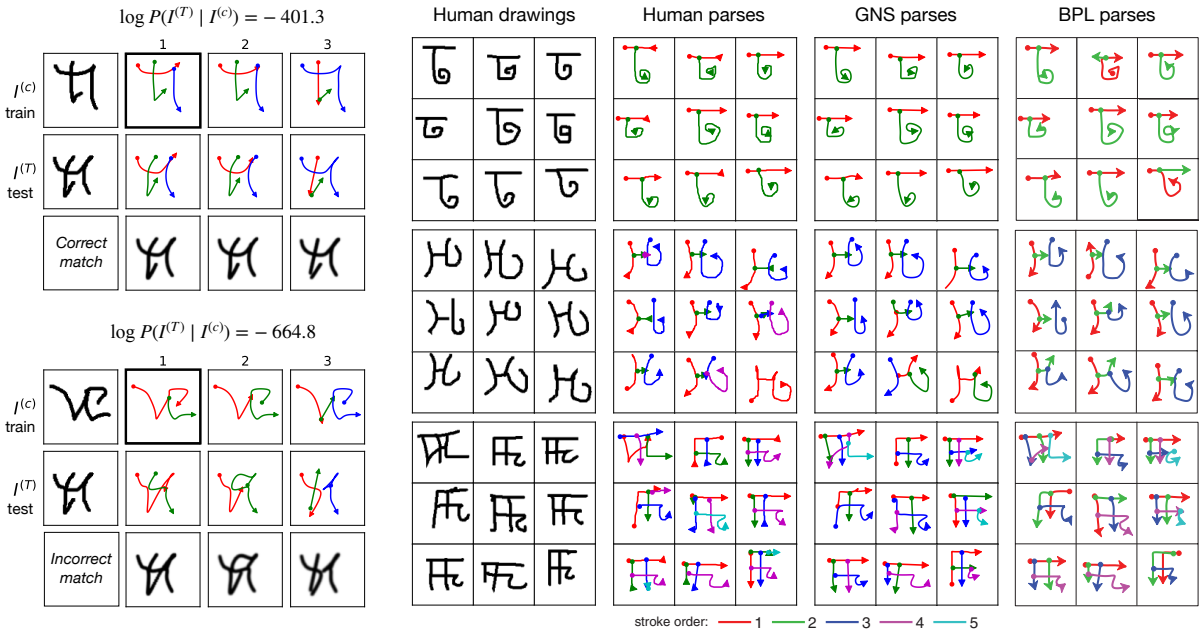
$$P(I) \geq \sum_{z_D \in \tilde{\Omega}_D} \int P(I, z_D, z_C) \partial z_C. \tag{3.8}$$

Our approximate posterior (Eq. 3.3) gives us K parses that represent promising modes $\{z_D, z_C\}_{1:K}$ of the joint density $P(I, z_D, z_C)$ for an image I, and by setting $\tilde{\Omega}_D = \{z_D\}_{1:K}$ to be the set of K unique discrete configurations from our parses, we can compute the lower bound of Eq. 3.8 by computing the integral $\int P(I, z_D, z_C) \partial z_C$ at each of these z_D.

At each $z_{Dk} \in \{z_D\}_{1:K}$, the log-density function $f(z_C) = \log P(I, z_{Dk}, z_C)$ has a gradient-free maximum at z_{Ck}, the continuous configuration of the corresponding posterior parse. These maxima were identified by our gradient-based continuous optimizer during parse selection (Section 3.5). If we assume that these maxima are sharply peaked, then we can use Laplace's method to estimate the integral $\int P(I, z_{Dk}, z_C) \partial z_C$ at each z_{Dk}. Laplace's method uses Taylor expansion to approximate the integral of $e^{f(x)}$ for a twice-differentiable function f around a maximum x_0 as

$$\int e^{f(x)} \partial x \approx e^{f(x_0)} \frac{(2\pi)^{\frac{d}{2}}}{|-H_f(x_0)|^{\frac{1}{2}}}, \tag{3.9}$$

where $x \in \mathbb{R}^d$ and $H_f(x_0)$ is the Hessian matrix of f evaluated at x_0. Our log-density function $f(z_C)$ is fully differentiable w.r.t. continuous parameters z_C, therefore we can compute $H(z_C) = \partial^2 f / \partial z_C^2$ with ease. Our approximate lower bound on $P(I)$ is therefore written as the sum of Laplace



(a) Classification fits

(b) Parsing

Figure 3.5: Classification fits and parsing. (a) Posterior parses from two training images were refit to the same test image. The first row of each grid shows the training image and its top-3 predicted parses (best emboldened). The second row shows the test image and its re-fitted training parses. Reconstructed test images are shown in the final row. The correct training image reports a high forward score, indicating that $I^{(T)}$ is well-explained by the motor programs for this $I^{(c)}$. (b) 27 character images from 3 classes are shown alongside their ground truth human parses, predicted parses from the GNS model, and predicted parses from the BPL model.

approximations at our K parses:

$$P(I) \geq \sum_{k=1}^{K} \int P(I, z_{Dk}, z_C) \partial z_C \approx \sum_{k=1}^{K} P(I, z_{Dk}, z_{Ck}) \frac{(2\pi)^{\frac{d}{2}}}{|-H(z_{Ck})|^{\frac{1}{2}}} \tag{3.10}$$

3.6 Experiments

GNS was evaluated on four concept learning tasks from the Omniglot challenge (Lake et al., 2019): one-shot classification, parsing, generating new exemplars, and generating new concepts. All evaluations use novel characters from completely held-out alphabets in the Omniglot evaluation set. As mentioned earlier, our goal is to provide a single model that captures deep knowledge of a

domain and performs strongly in a wide range of tasks, rather than besting all models on every task. Our experiments include a mixture of quantitative and qualitative evaluations, depending on the nature of the task. We have released an open-source code implementation of our experiments at the following url: https://github.com/rfeinman/GNS-Modeling.

3.6.1 One-shot classification

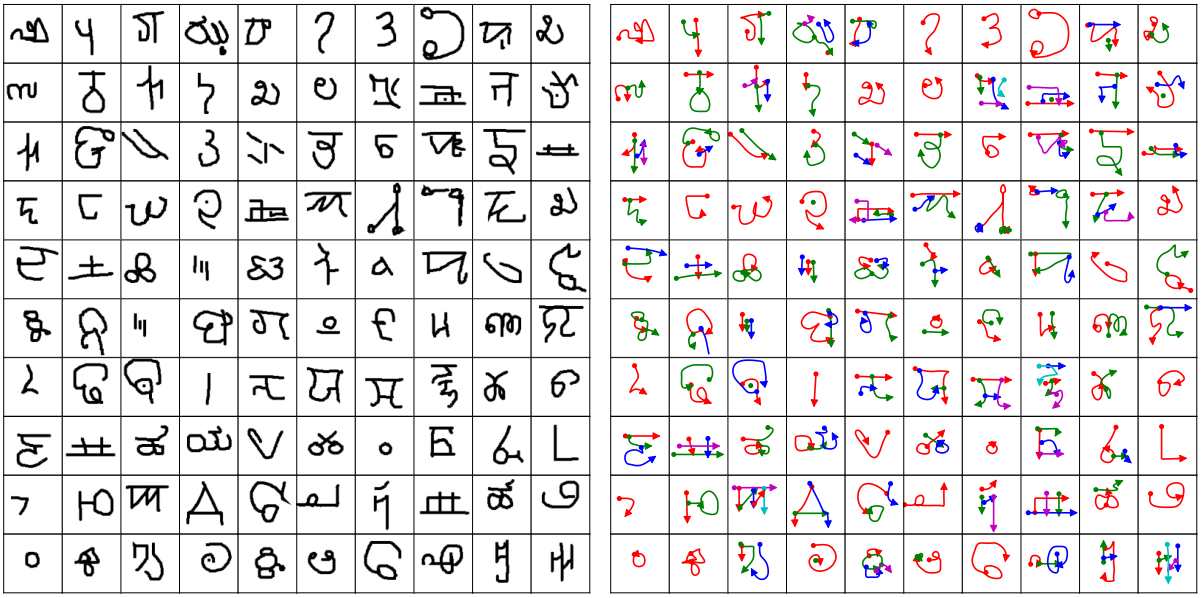
GNS was compared with alternative models on the one-shot classification task from Lake et al. (2015). The task involves a series of 20-way within-alphabet classification episodes, with each episode proceeding as follows. First, the machine is given one training example from each of 20 novel characters. Next, the machine must classify 20 novel test images, each corresponding to one of the training classes. With 20 episodes total, the task yields 400 trials. Importantly, all character classes in an episode come from the same alphabet as originally proposed in Lake et al. (2015), requiring finer discriminations than commonly used between-alphabet tests (Lake et al., 2019).

Model	Error
GNS	5.7%
BPL (Lake et al., 2015)	3.3%
RCN (George et al., 2017)	7.3%
VHE (Hewitt et al., 2018)	18.7%
Proto. Net (Snell et al., 2017)	13.7%
ARC (Shyam et al., 2017)	1.5%*

Table 3.2: Test error on within-alphabet one-shot classification. The ARC model used 4x training classes.

As illustrated in Fig. 3.5(a), GNS classifies a test image by choosing the training class with the highest Bayesian score (Eq. 3.5). A summary of the results is shown in Table 3.2. GNS was compared with other machine learning models that have been evaluated on the within-alphabets classification task (Lake et al., 2019). GNS achieved an overall test error rate of 5.7% across all 20 episodes (N=400). This result is very close to the original BPL model, which achieved 3.3% error

with significantly more hand-design. The symbolic relations in BPL's token model provide rigid constraints that are key to its strong classification performance (Lake et al., 2015). GNS achieves strong classification performance while emphasizing the nonparametric statistical knowledge needed for creative generation in subsequent tasks. Beyond BPL, our GNS model outperformed all other models that received the same background training. The ARC model (Shyam et al., 2017) achieved an impressive 1.5% error, although it was trained with four-fold class augmentation and many other augmentations, and it can only perform this one task. In Fig. A.5, we show a larger set of classification fits from GNS, including examples of misclassified trials.



(a) Target images

(b) GNS parses

Figure 3.6: Parsing. GNS predicted parses for 100 character images selected at random from the Omniglot evaluation set. (a) A 10x10 grid of target images. (b) A corresponding grid of GNS predicted parses per target image.

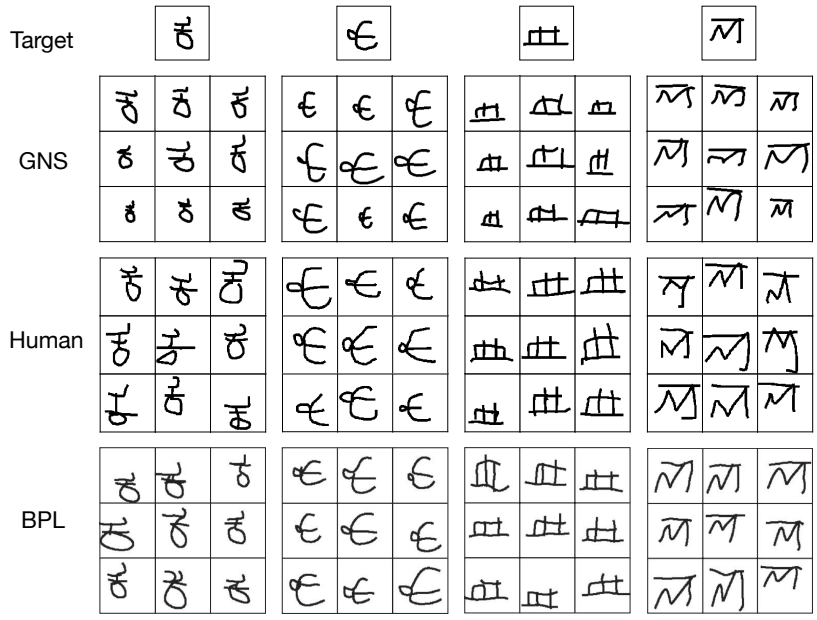
3.6.2 Parsing

In the Omniglot parsing task, machines must segment a novel character into an ordered set of strokes. These predicted parses can be compared with human ground-truth parses for the same images. The

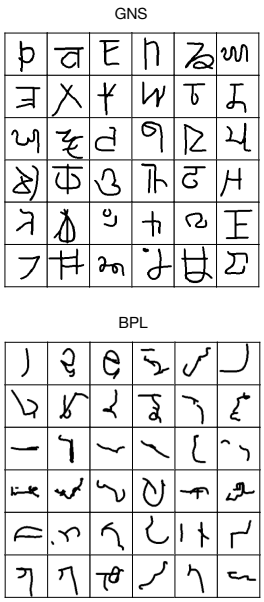
approximate posterior of GNS yields a series of promising parses for a new character image, and to complete the parsing task, we identify the maximum a posteriori parse $k^* = \max_k \pi_k$, reporting the corresponding stroke configuration. Fig. 3.5(b) shows a visualization of the GNS predicted parses for 27 different raw images drawn from 3 unique character classes, plotted alongside ground-truth human parses (how the images were actually drawn) along with predicted parses from the BPL model. Compared to BPL, GNS parses possess a few unique desirable qualities. The first character class has an obvious segmentation to the human eye—evidenced by the consistency of human parses in all examples—and the GNS model replicates this consistency across all 9 predicted parses. In contrast, BPL predicts seemingly-unlikely parses for 2 of the examples shown. The second character is more complex, and it was drawn in different styles by different human subjects. The GNS model, which is trained on data from subjects with different styles, captures the uncertainty in this character by predicting a variety of unique parses. BPL, on the other hand, produces a single, ubiquitous segmentation across all 9 examples. In Fig. 3.6, we provide a larger set of parses from the GNS model for a diverse range of Omniglot characters.

3.6.3 Generating new exemplars

Given just one training image of a novel character concept, GNS produces new exemplars of the concept by sampling from the approximate conditional $P(I^{(2)}, \theta^{(2)} \mid I^{(1)})$ of Eq. 3.6. In Fig. 3.7(a) we show new exemplars produced by GNS for a handful of target images, plotted next to human productions (more examples in Fig. A.4). In the majority of cases, samples from the model demonstrate that it has successfully captured the causal structure and invariance of the target class. In contrast, deep generative models applied to the same task miss meaningful compositional and causal structure, producing new examples that are easily discriminated from human productions (Rezende et al., 2016; Hewitt et al., 2018) (see Fig. 3.2). In some cases, such as the third column of Fig. 3.7(a), samples from GNS exhibit sloppy stroke junctions and connections. Compared to BPL, which uses engineered symbolic relations to enforce rigid constraints at stroke junctions, GNS



(a) Generating new exemplars



(b) Generating new concepts

Figure 3.7: Generation tasks. (a) GNS produced 9 new exemplars for each of 5 target images, plotted here next to human and BPL productions. (b) A grid of 36 new character concepts sampled unconditionally from GNS, shown next to BPL samples.

misses some of these structural elements. Nevertheless, new examples from GNS appear strong enough to pass for human in many cases, which we would like to test in future work with visual Turing tests.

3.6.4 Generating new concepts (unconstrained)

In addition to generating new exemplars of a particular concept, GNS can generate new character concepts altogether, unconditioned on training images. Whereas the BPL model uses a complicated procedure for unconditional generation that involves a preliminary inference step and a supplemental nonparametric model, GNS generates new concepts by sampling directly from the type prior $P(\psi)$. Moreover, the resulting GNS productions capture more of the structure found in real characters than either the raw BPL prior (Fig 3.1, 3.7(b)) or the supplemental nonparametric BPL prior (Lake et al., 2015). In Fig. 3.7(b) we show a grid of 36 new character concepts sampled from our generative

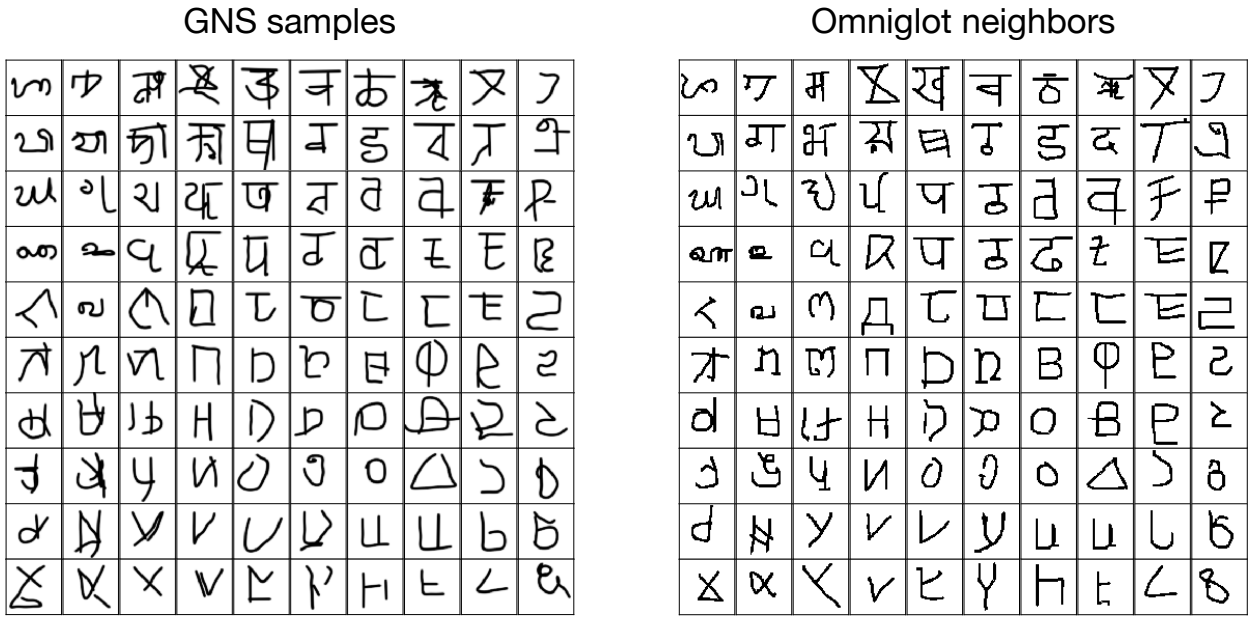


Figure 3.8: Generating new concepts (unconstrained). 100 new concepts sampled unconditionally from GNS are shown in a topologically-organized grid alongside a corresponding grid of “nearest neighbor" training examples. To identify nearest neighbors, we used cosine distance in the last hidden layer of a CNN classifier as a metric of perceptual similarity. The CNN was trained to classify characters from the Omniglot background set, a 964-way classification task.

model at reduced temperature setting $T = 0.5$ (Appendix A.3). The model produces characters in multiple distinct styles, with some having more angular, line-based structure and others relying on complex curves. In Fig. 3.8, we show a larger set of characters sampled from GNS, plotted in a topologically-organized grid alongside a corresponding grid of “nearest neighbor" training examples. In many cases, samples from the model have a distinct style and are visually dissimilar from their nearest Omniglot neighbor.

3.6.5 Marginal image likelihoods

As a final evaluation, we computed likelihoods of held-out character images by marginalizing over the latent type and token variables of GNS to estimate $P(I) = \int P(\psi, \theta, I) \partial\psi \partial\theta$. We hypothesized that our causal generative model of character concepts would yield better test likelihoods compared to deep generative models trained directly on image pixels. As detailed in Section 3.5.3, under the

Model	Image Size	LL	LL/dim
VHE	28×28	-61.2	-0.0496
SG	52×52	-134.1	-0.0781
GNS	105×105	-383.2	**-0.0348**

Table 3.3: Test log-likelihood bounds.

minimal assumption that our K posterior parses represent sharply peaked modes of the joint density, we can obtain an approximate lower bound on the marginal $P(I)$ by using Laplace's method to estimate the integral around each mode and summing the resulting integrals. In Table 3.3, we report average log-likelihood (LL) bounds obtained from GNS for a random subset of 1000 evaluation images, compared against test LL bounds from both the SG (Rezende et al., 2016) and the VHE (Hewitt et al., 2018) models. Our GNS model outperforms each alternative, reporting the best overall log-likelihood per dimension.

3.7 Discussion

We introduced a new generative neuro-symbolic (GNS) model for learning flexible, task-general representations of character concepts. We demonstrated GNS on the Omniglot challenge, showing that it performs a variety of inductive tasks in ways difficult to distinguish from human behavior. Some evaluations were still qualitative, and future work will further quantify these results using Visual Turing Tests (Lake et al., 2015).

Whereas many machine learning algorithms emphasize breadth of data domains, isolating just a single task across datasets, we have focused our efforts in this paper on a single domain, emphasizing depth of the representation learned. Human concept learning is distinguished for having both a breadth and depth of applications (Murphy, 2002; Lake et al., 2019), and ultimately, we would like to capture both of these unique qualities. We see our character model as belonging to a broader class of generative neuro-symbolic (GNS) models for capturing the data generation process. We

have designed our model based on general principles of visual concepts—namely, that concepts are composed of reusable parts and locations—and we will later describe how it generalizes to compositional visual concepts (Chapter 4) as well as real-world object concepts (Section 7.3). As in the human mind, machine learning practitioners have far more prior knowledge about some domains vs. others. Handwritten characters is a domain with strong priors (Babcock & Freyd, 1988; Longcamp et al., 2003a; James & Gauthier, 2009), implemented directly in the human mind and body. For concepts like these with more explicit causal knowledge, it is beneficial to include priors about how causal generative factors translate into observations, as endowed to our character model through its symbolic rendering engine. For other types of concepts where these processes are less clear, it may be appropriate to use more generic neural networks that generate concepts and parts directly as raw stimuli, using less symbolic machinery and prior knowledge. We anticipate that GNS can flexibly model concepts in both types of domains, although further experiments are needed to demonstrate this.

Our current token model for character concepts is much too simple, and we acknowledge a few important shortcomings. First, as shown in Fig. A.5, there are a number of scenarios in which the parses from a training character cannot adequately refit to a new example of the same character without a token model that allows for changes to discrete variables. By incorporating this allowance in future work, we hope to capture more knowledge in this domain and further improve performance. Furthermore, although our vision for GNS is to represent both concepts and background knowledge with neuro-symbolic components, the current token-level model uses only simple parametric distributions. In future work, we hope to incorporate token-level models that use neural network sub-routines, as in the type-level model presented here.

Chapter 4

Few-shot learning of structured visual concepts

4.1 Preface

The experiments of Chapters 2 & 3 provide preliminary support for generative neuro-symbolic (GNS) models of concept learning. Although the results of these experiments are promising, they are only a start; handwritten characters lack many of the interesting qualities found in other types of concepts that people grapple with ease. For example, the variability between different tokens of a character is limited to simple motor and affine noise. In a sense, a character type represents a *prototype* around which unique tokens vary in consistent ways. This contrasts with other conceptual domains, where tokens of a concept can vary in different ways depending on the type. For example, tokens of the concept "table" vary in their number of legs (1 vs. 4) or in the shape of their top (square vs. round), whereas tokens of "dresser" vary in their number of drawer levels and the style of drawer handle, among other things. Concepts like these call for a more powerful token-level model.

This chapter expands on the GNS framework and studies a new family of visual concepts with some of the distinguishing qualities discussed above. It is based off a collaborative work with fellow Ph.D. student Yanli Zhou, currently under peer review and available as a preprint (Zhou et al., 2023). For the study, Yanli developed a human behavioral experiment to investigate how people generalize structured visual concepts composed of simple shape primitives. She created a class of stimuli—dubbed “alien figures"—with are richly hierarchical, compositional and relational. Each alien figure concept represents a large class of stimuli with significant inter-token variability. Although highly variable, each concept has rigid rules about the ways that primitives are allowed to appear and combine with one another, suggestive of a specific form of compositionality. A rich diversity of composition rules are included in the behavioral experiment. Using the stimuli and behavioral results from Yanli’s experiment, I conduct a rigorous study to test whether GNS models can explain the ways that people learn and generalize structured visual concepts. I develop a new GNS model designed to represent alien figure concepts, as well as a novel meta-training paradigm that facilitates few-shot learning and helps explain how people generalize from limited examples. The GNS model is evaluated and compared against a fully-symbolic Bayesian model that manifests strong domain expertise.

Section 4.2 provides a brief overview of the stimuli and behavioral experiments presented in Zhou et al. (2023). It also describes a fully-symbolic computational model that was developed as a preliminary account of human behavior. Importantly, **the materials in Section 4.2 are not my work:** they are Yanli’s contribution to our collaborative paper. I include them only to provide the necessary context and background for my own work, which constitutes the remaining sections of this chapter.

4.2 Structured visual concepts

4.2.1 Stimuli

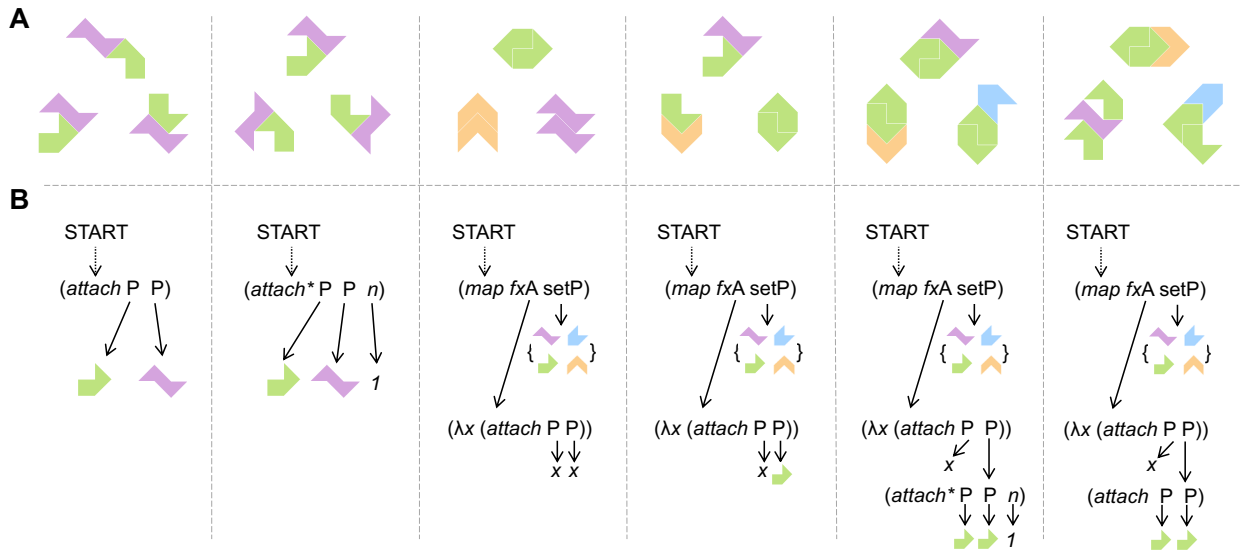


Figure 4.1: Examples of alien figure stimuli and concepts, derived from Zhou et al. (2023, Fig. 3). Each figure is a compound shape formed from 1-3 basic shape primitives. The figures are organized into concepts according to systematic formulas, visualized in (B) as simplified parse trees from a context-free grammar (Zhou et al., 2023), and the concepts are sampled to form trials for human experiments.

Zhou et al. (2023) present a human behavioral experiment designed to explore the ways that people generalize simple visual concepts from just a few examples. In the experiment, participants are presented with “alien figure" stimuli—compound shapes formed from 1-3 basic shape primitives (Fig. 4.1)—and prompted to reason about their characteristic structure via one of two distinct tasks. The stimuli are generated programmatically and grouped according to a variety of systematic formulas, resulting in a diverse collection of compositional concepts. Fig. 4.1 shows some examples of these concepts, providing a simplified parse tree for each concept that represents its syntactic structure according to a context-free grammar (Zhou et al., 2023). As one example, the third column describes a concept that includes any two-part token composed of two instances of a single primitive.

This concept is invariant to the primitive identity, as well as to the attachment between the first and second instance of the primitive, but it does not allow changes to the number of parts or the number of unique primitives used in the figure.

4.2.2 Few-shot learning tasks

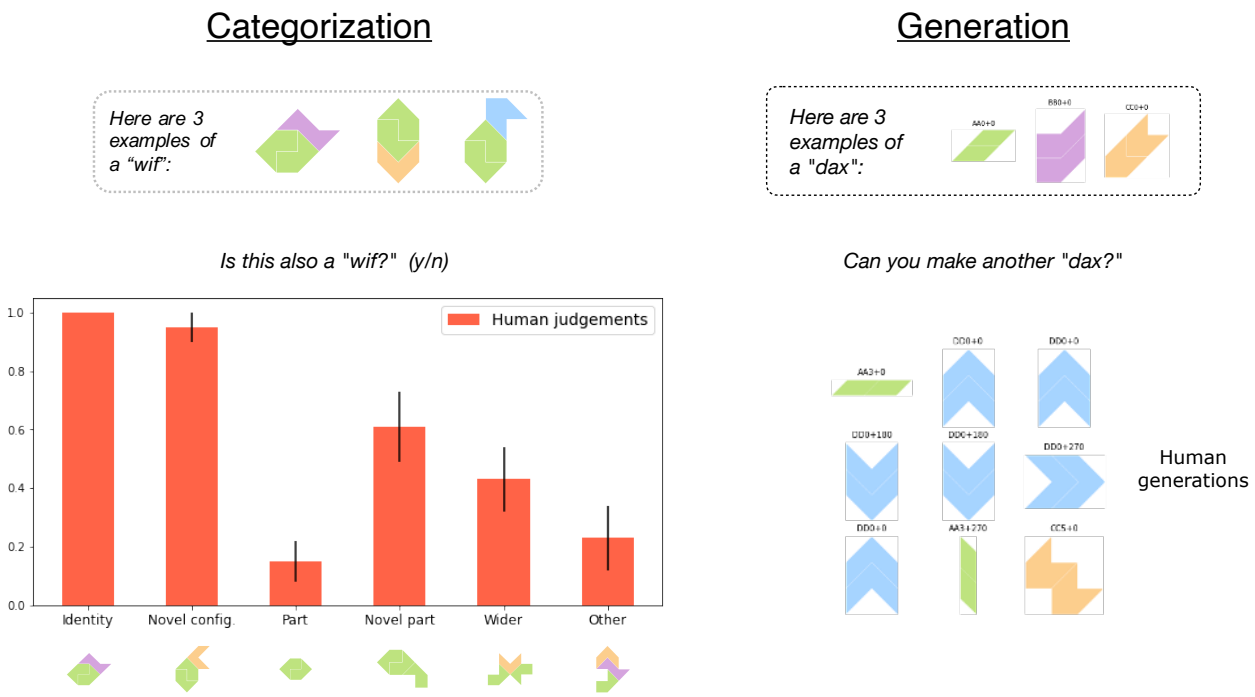


Figure 4.2: Categorization and generation tasks from Zhou et al. (2023). In both tasks, participants are first familiarized with a new alien figure concept through a collection of exemplars. In categorization, participants are then shown a set of query stimuli and asked to answer yes/no whether each is a member of the category. In generation, participants are instead asked to generate another example of the concept using by composing basic shape primitives with a web tool.

Human participants were recruited via Amazon's Mechanical Turk and asked to perform one of two tasks that evaluate their ability to learn and generalize alien figure concepts from limited examples (Fig. 4.2). In each task, the participant is first familiarized with a novel alien figure concept via a set of 1-6 exemplars. In the categorization task, the participant is then presented with a series of query stimuli and asked to answer yes/no as to whether they believe each query to be an

instance of the familiarized concept. In the generation task, the participant is not shown any new stimuli but is instead asked to generate another example of the concept. To generate an example, the participant is provided a web interface that enables them to construct alien figure compounds by selecting and piecing together primitives akin to children's building blocks.

4.2.3 Symbolic Bayesian model

Zhou et al. (2023) develop a fully-symbolic model as a preliminary account for human few-shot learning of alien figure concepts, using an approach inspired by probabilistic language of thought models (Goodman et al., 2008; Piantadosi et al., 2016). Concept learning is modeled as Bayesian reasoning over structured hypotheses in a probabilistic context-free grammar. Each hypothesis h defines criteria that must hold true for category membership, implicitly specifying a concept and its constituent tokens. Given a set of observed examples $X = \{x_1, ..., x_n\}$ the model uses Bayes rule to reason about the most probable a posteriori hypotheses,

$$P(h|X) \approx P(h)P(X|h). \tag{4.1}$$

The prior $P(h)$ is defined by a probabilistic grammar (Zhou et al., 2023, Section 1.2.2), and the likelihood $P(x|h)$ of new tokens given hypothesis h is a simple uniform distribution over the tokens that satisfy membership in h.

Inference for few-shot learning uses Metropolis-Hastings with subtree regeneration (Goodman et al., 2008) to sample from the posterior $P(h|X)$. The specific algorithm varies between the categorization and generation tasks. Categorization introduces a new likelihood $P(l_y = 1|h)$ to represent the probability of answering "yes" about whether a new token y is consistent with hypothesis h, and then derives a novel formula to model categorization responses from examples. Generation, on the other hand, uses the canonical posterior predictive distribution $P(y|X)$ for new

tokens y given observations X, obtained by marginalizing over available hypotheses:

$$P(y|X) = \sum_{h \in H} P(y|h)P(h|X). \tag{4.2}$$

The probabilistic grammar for $P(h)$ has a number of hyperparameters, and these parameters are optimized separately in each task for overall fit to human data from the task. Additional details about the model are provided in Sections 1.2 & 2.2 of Zhou et al. (2023).

4.3 Generative neuro-symbolic (GNS) model

Symbolic probabilistic models like the one from Section 4.2.3 provide an elegant and interpretable account of human behavior; however, these models make simplifying and rigid parametric assumptions, and as result, they often leave portions of the data unexplained. For example, the Bayesian program induction model assumes that all constituent tokens x_i of a hypothesis h are sampled with equal probability. This assumption appears at odds with humans, who at times exhibit a preference for certain tokens over others within a particular grouping (Zhou et al., 2023, Fig. 7C&D). Although there may be an ad-hoc rule to explain each behavioral nuance like this, engineering such primitives would involve a considerable effort, and the complexity of the resulting model could quickly grow out of hand. Alternatively, we could let the data speak for itself by integrating more powerful data-driven modeling components.

In pursuit of a more complete computational account with much of the same structure and interpretablility, we propose to model human concepts of alien figures as neuro-symbolic probabilistic programs. This paradigm, known as Generative Neuro-Symbolic (GNS) Modeling, was shown to provide an effective framework for understanding another type of compositional visual concept: handwritten letters from different alphabets (Chapters 2 & 3). As in the fully-symbolic Bayesian model, the aim of GNS is to infer the best causal generative process for explaining the visual

examples. Unlike symbolic models, GNS further represents nonparametric statistical relationships between parts in a token, and between tokens in an observation, providing a more flexible model with fewer a priori assumptions. Moreover, a GNS model can be estimated directly from training data, providing an effective data-driven approach. An important component of our approach is that we train GNS to mimic the Bayesian program induction model by using the Bayesian model to generate some of its training data, while also including real human data so that GNS can go further to capture additional structure in human behavior.

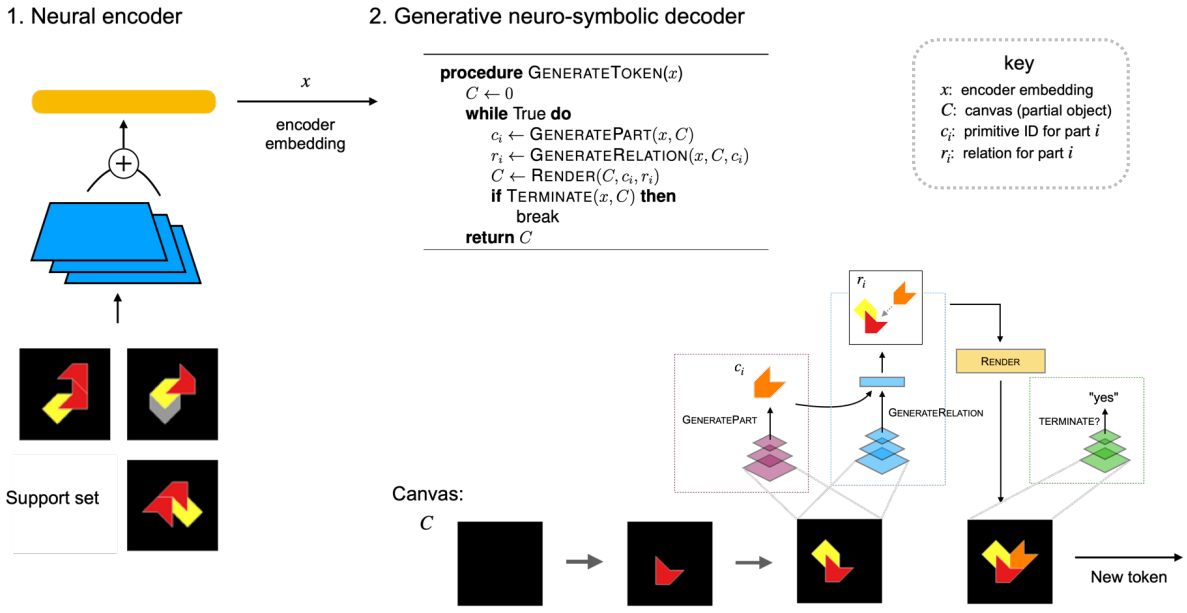


Figure 4.3: Overview of GNS model. A neural encoder first reads each support example with a convolutional neural network (CNN) and aggregates the resulting vectors into a single, fixed-sized embedding. This encoder embedding is then passed to a GNS decoder–expressed as probabilistic program `GenerateToken`–that generates new tokens one part at a time, using an image canvas C as memory. At each part iteration i, the current canvas C and encoder embedding x are first fed to subroutine `GeneratePart` which generates the primitive ID c_i of the next part. Next, C, x and c_i are passed to subroutine `GenerateRelation` which samples a relation specification r_i for the part. Finally, a symbolic renderer updates the canvas according to c_i and r_i, and subroutine `Terminate` decides whether to terminate the token.

A depiction of the proposed GNS model is given in Fig. 4.3. Similar to our previous model of handwritten characters (Chapters 2 & 3), our GNS model of alien figures uses the control flow of a probabilistic program, coupled with an external image memory, to represent the causal

process of generating new concepts. Through repeated calls to subroutines `GeneratePart` and `GenerateRelation` the model maintains a representation that is *compositional*, providing and appropriate inductive bias for compositional generalization. Each of these modular subroutines is expressed as a neural network that generates symbolic outputs conditioned on the current program state (Fig. 4.4). New from prior work, we augment the GNS model with an image encoder to account for the ways that people induce conceptual representations from exemplars in the current behavioral experiment. With this addition, we can use our GNS model as a proxy to the Bayesian model's posterior predictive distribution (Zhou et al., 2023, Eq. 5). Given a set of support exemplars, the encoder first reads each exemplar using a convolutional neural network (CNN) and then aggregates the individual responses to form a single vector embedding of the set. This embedding is passed to the decoder and used to condition a generative model for new tokens. Both the encoder and the decoder use a coloring scheme for alien figure images that associates each primitive from our primitive bank with a unique RGB color.

4.3.1 Encoder

The support encoder (Fig. 4.3, left) consists of a convolutional neural network (CNN) backbone and a transformer aggregator. The CNN first reads each exemplar in the support set, represented as an 80×80 RGB image, and encodes the image to a 256-dimensional vector. The sequence of CNN vectors is then fed to a transformer encoder, which processes the variable-length sequence and outputs an aggregate vector encoding of the set.

4.3.2 Decoder

Our GNS decoder (Fig. 4.3, right) generates new tokens by sampling a sequence of symbolic primitives $\{\theta, c_{1:\kappa}, r_{1:\kappa}, \}$ which together specify a unique instance of an alien figure concept with κ parts. Part assignments c_i convey the category of the i^{th} part, chosen from a dictionary of 9 basic

primitive categories, and relations r_i specify how the i^{th} part attaches to previously-generated parts, with r_1 assigned to null. Each attachment specification r_i encompasses 3 unique sub-choices: an index j of the previous part onto which the current part i will attach, and a choice of polygon sides s_j and s_i for the previous and current part that will touch at the point of attachment.

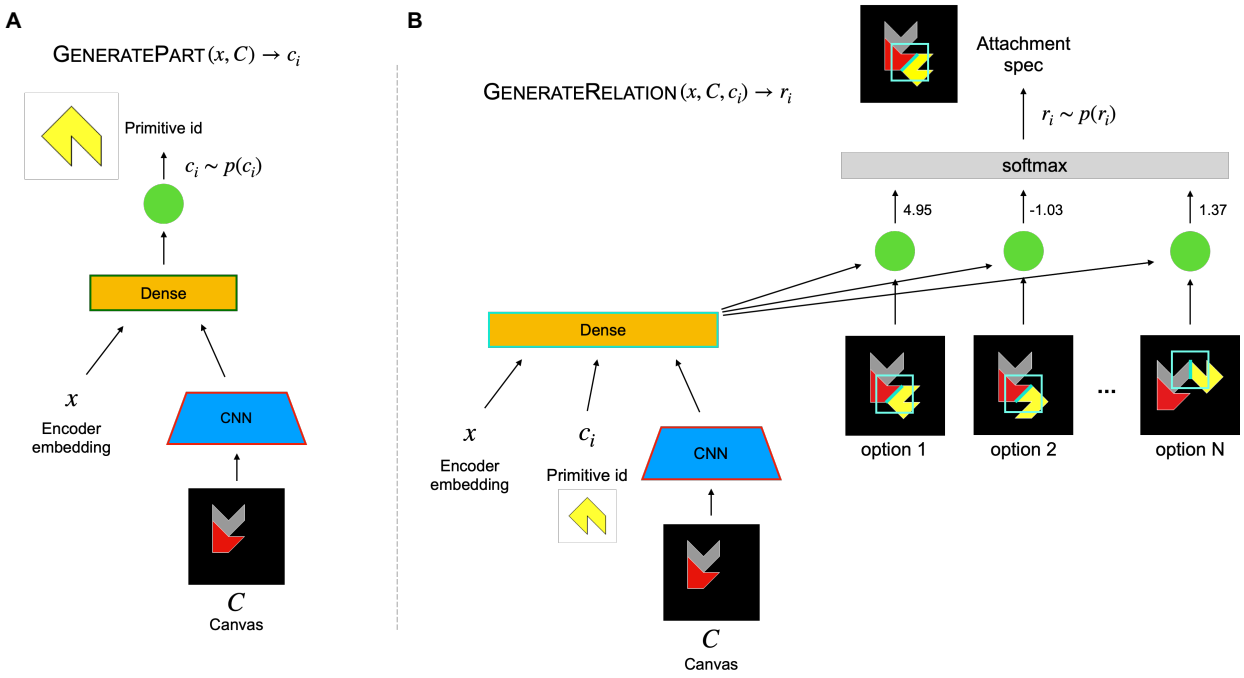


Figure 4.4: GNS Subroutines. (A) Subroutine `GeneratePart` first reads the image canvas with a CNN and concatenates the response with encoder embedding x. The combined vector is then processed by a dense layer and passed to a softmax prediction head that yields a categorical distribution to sample the next primitive ID c_i. (B) Subroutine `GenerateRelation` similarly reads the canvas with a CNN, this time concatenating with both the encoder embedding x as well as primitive ID c_i from `GeneratePart`. The combined vector is processed by a dense layer and then passed to a relation prediction head that yields a probability distribution to sample the next relation r_i (see Fig. B.1 for additional details).

The generative process to sample a new token conditioned on support embedding x proceeds as follows. We first initialize an empty image canvas, C, that will maintain the state of the sample. Next, we sample a global orientation θ for the token from subroutine `GenerateOrientation`. This is an additional neural network module that is used only once at the start of the sample and it selects from 4 discrete orientation choices. From there, we iteratively sample the next part and

next relation from subroutines `GeneratePart` and `GenerateRelation` until a termination is reached. Each of these sample steps conditions on the support, as well as the current partial-object, by reading x and C as neural network inputs. This design enables the model to capture complex correlations that permeate through multiple parts of an object, or that connect a new object to support examples. At the end of each iteration, we update our canvas C with the latest partial-object using a symbolic image renderer and pass the new canvas to subroutine `Terminate`, a neural network that decides whether to terminate the object or continue with another part.

The architectures of the neural networks for `GeneratePart` and `GenerateRelation` are depicted in Fig. 4.4. In `GeneratePart`, a CNN embeds the current image canvas to a vector and concatenates it with the encoder embedding. The combined vector is then processed by a fully-connected (dense) layer, and a softmax layer then predicts a categorical distribution for the primitive ID of the next part. In `GenerateRelation`, a CNN similarly encodes the image canvas, this time concatenating the resulting vector with both the encoder embedding as well as a discrete embedding of the primitive ID chosen in the previous step. The concatenated vector is then processed by a dense layer and fed to an attention-style prediction head. Using this input and an attention-style weighting scheme, the prediction head outputs a distribution over discrete choices for how and where the new part will attach to previous ones in the canvas (Fig. B.1).

4.4 Training with meta-learning

The objective of few-shot generation is to generate new tokens of a concept given a limited set of support exemplars. In the Bayesian setting, this task is modeled as sampling from the posterior predictive probability $p(y \mid X)$ of a new token y given a support set $X = \{x_1, ..., x_n\}$ consisting of n exemplars. Our GNS model provides a nonparametric analogue to the posterior predictive that can be estimated directly from training data, written $p(y \mid X) \approx f_\theta(y; X)$, where f represents the model approximation parameterized by θ. To train GNS effectively, we borrow a paradigm from AI

procedure P
$h \sim p(h)$ ▷ Sample formula hypothesis from prior
$S = x_1, ..., x_n \sim p(x \mid h)$ ▷ Sample support set from formula
$Q = x'_1, ..., x'_n \sim p(x \mid h)$ ▷ Sample query set from formula
return S, Q

procedure R
$S \sim \text{Uniform}(\Phi)$ ▷ Sample support set from human trials
$h \sim p(h \mid S)$ ▷ Sample formula hypothesis from posterior
$Q = x'_1, ..., x'_n \sim p(x \mid h)$ ▷ Sample query set from formula
return S, Q

procedure H
$S, Q \sim \text{Uniform}(\Phi)$ ▷ Sample support & query sets from human trials
return S, Q

Figure 4.5: Data distributions for meta-learning.

known as *meta-learning* (Hospedales et al., 2022). Each input or "episode" provided to the model consists of 1) a set of support tokens, a.k.a. exemplars, and 2) a set of query tokens for the model to evaluate. Through these episodes the model *learns-to-learn*, capturing overarching patterns that connect queries to support and learning to quickly grasp new concepts from exemplars.

As with any statistical estimator that uses neural networks, our GNS model calls for a sizeable training dataset to avoid overfitting and ensure adequate generalization. The behavioral dataset from our generation task consists of just 155 trials in total, an insufficient amount of data by itself. To fill in the gap, we use our symbolic Bayesian model to bootstrap GNS training with a vast supply of synthetic meta-learning data. Specifically, we use the Bayesian model to form two distinct distributions for generating training data (Fig. 4.5). In the first distribution P, episodes are generated by first sampling a hypothesis h from the prior and then sampling a support set S and query set Q from the likelihood $p(X \mid h)$ (Zhou et al., 2023, Eq. 3). In a second distribution R, episodes are generated by sampling a support set S uniformly from the human experiment and then sampling query Q via the posterior of the Bayesian model. In addition to these two synthetic data distributions, we also use real human data, dubbed distribution H, as part of the training mix. Episodes from H are sampled uniformly from the human experiment.

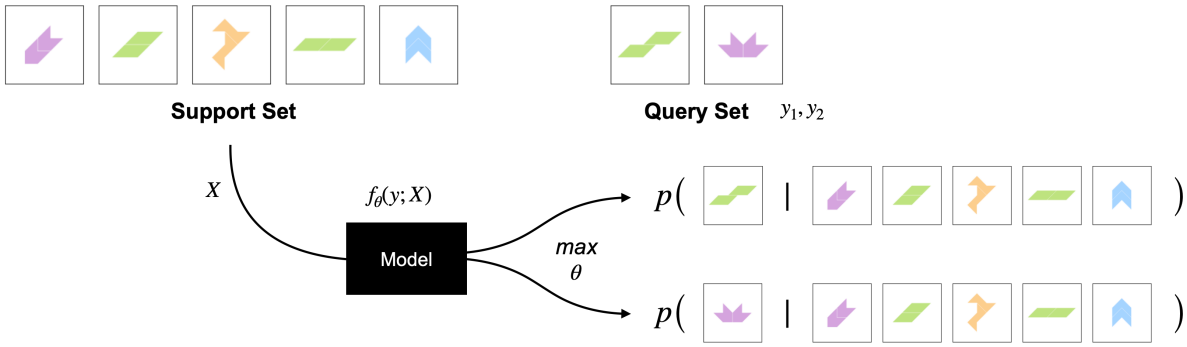


Figure 4.6: Meta-learning episodes. Each episode consists of 1) a support set X of 1-6 examples that demonstrate the concept, and 2) a query set of additional tokens for evaluation $y_1, y_2, \ldots$. The GNS model is trained to maximize the conditional log-likelihood of each query token given the support examples.

In addition to the P, R and H distributions described above, we make use of one additional data distribution, C, which provides concerted training of two inductive biases that were prominent in human behavior during the generation task, and yet that the Bayesian program induction model is unable to capture. In trials where the support exemplars convey a partial pattern with one item apparently left out, participants exhibit two salient inductive biases, both of which are noted in Zhou et al. (2023, Section 2.3.1(ii)). First, people demonstrate a strong preference to complete the pattern by generating the last remaining item. We refer to this tendency as the *complete-the-pattern* bias, and it occurs an aggregate 59% of the time in applicable trials. When not completing the pattern, people exhibit a second, slightly weaker inductive bias that we denote the *reconfigure* bias. This bias is characterized by a preference to take the familiarized primitives and reconfigure them into a novel, multi-part object. Participants exhibit the reconfigure bias an aggregate 14% of the time in the applicable trials. Our concerted bias training distribution, C, is designed to teach the GNS model these biases and help guide it toward generalizing in more human-like ways. Appendix B.2 provides details about how episodes are generated from C.

4.5 Experiments

Our first experiment is designed to test whether the GNS model can successfully learn to generate new tokens from exemplars, and to determine what training distributions are most important for learning this task. For the experiment, we constructed a test set of human data consisting of 1 randomly-selected trial from each trial type in the generation task (see Zhou et al. (2023, Section 2.1.1) for details on trials & trial types). The remaining 4 trials of each type are provided for model training. By reserving a portion of the human data for test time, we can use log-likelihood evaluations to assess whether the GNS model generalizes to novel trials with unseen behavioral data, and to compare the behavioral account of GNS to that of the Bayesian model.

Model	log-likelihood	t-statistic (p-value)
Bayesian	-4.741	-
GNS (P/R/H/C)	**-4.444**	6.197 (0.000)
GNS (P/R/H)	-4.535	4.549 (0.000)
GNS (P/R)	-4.645	2.490 (0.013)
GNS (P)	-4.930	-2.739 (0.006)

Table 4.1: Holdout log-likelihoods. For each model, the average log-likelihood per human token is reported in the first column. For each GNS model, we perform a paired t-test to test for improvement over the Bayesian model. The full GNS model, and all but one lesion model, show an improved behavioral fit over the Bayesian model, fortified by significant t-test results.

Our full GNS model, GNS (P/R/H/C), uses a mixture of all four training distributions described in the previous section. This represents our most comprehensive training environment, and we anticipate that the resulting model will outperform alternatives that receive only a subset of the proposed training distributions. We test a series of these alternatives. The first, GNS (P/R/H), receives all but the bias training distribution C. In addition, we also evaluated two lesions that receive only synthetic data from the Bayesian model. One of these, GNS (P/R), receives data from both of the two synthetic generators. The other, GNS (P), uses only the forward-sampling modality P. Each of our models is trained using minibatches of 60 meta-learning episodes (Appendix B.2).

Log-likelihood results for held-out human data are shown in Table 4.1. When evaluating test log-likelihoods, we mix the model distribution with a naive lapse distribution using weight α that is independently tuned for each model (Section B.3). Our full GNS model, GNS (P/R/H/C), performs the strongest on held-out data and shows a considerable improvement in log-likelihood over the Bayesian model. The improvement is further validated by a significant paired t-test that looks at per-token difference in log-likelihood $\ell(\theta) - \ell(\theta_0)$ between the GNS model, θ, and Bayesian model θ_0 [t(336) = 6.197, p < 0.001]. After lesioning the bias training distribution, our GNS (P/R/H) model still exhibits a significant log-likelihood improvement over Bayesian program induction, although the gain is smaller. The simplest lesioned model, GNS (P), performs the weakest on held-out data and does not show a significant improvement over the Bayesian model. This result matches our intuition: the space of possible episodes generated from P is vast, and so it is unlikely that the model will receive sufficient experience with the types of support sets that are relevant to our human experiment. Our second lesion, GNS (P/R), is the first to outperform the symbolic Bayesian model and show a statistically significant improvement in log-likelihood. Like GNS (P), this model is trained solely on synthetic data from the Bayesian model; however, the way that episodes are sampled in R—by selecting a support S from the human experiment and then sampling query Q from the Bayesian posterior—ensures that a sufficient amount of relevant training experience is provided.

To help understand how and where our full GNS model outperforms Bayesian program induction, Fig. 4.7 shows some of the top-performing examples where the log-likelihood improvement is largest (a more exhaustive set of best and worst examples is provided in Fig. B.2). The GNS model does particularly well with the two-part concept from rows 1, 2, and 3. In this trial, the size principle pushes the Bayesian model to assign most posterior weight to an attachment-specific hypothesis, so when a new token is shown with a different attachment, it loses out. GNS also outperforms on the completion bias example from row 4, a result that is expected since the model receives explicit completion bias training from distribution C. In row 5, the Bayesian model assigns a majority of

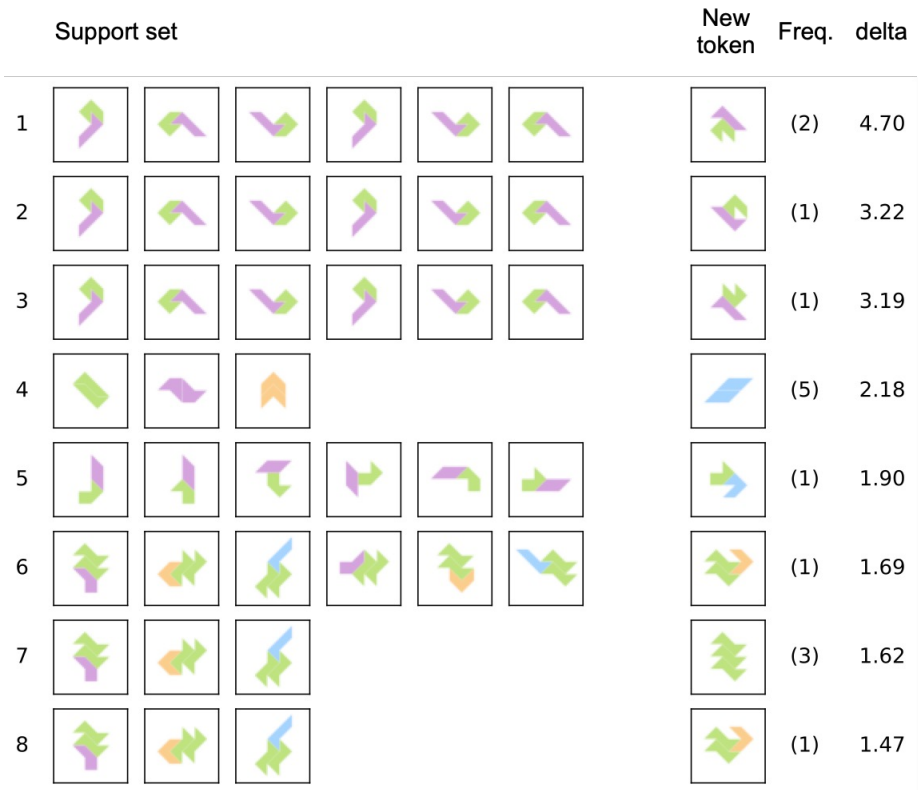


Figure 4.7: A subset of most-improved examples, measured by ℓ(GNS) - ℓ(Bayes).

posterior weight to a primitive-specific hypothesis, and it therefore suffers on the human-generated token that uses a new primitive. The concept from rows 6, 7 and 8 has a salient visual compound that likely guides stronger generalization in human participants, but the Bayesian model is not aware of the compound. The GNS model, however, is capable of picking up on this visual pattern and mirroring human generalization.

To further understand how and whether the GNS model provides an improved account of human inductive biases, we conducted an additional experiment designed to give a more in-depth look at the *complete-the-pattern* and *reconfigure* biases discussed in Section 4.4 & Zhou et al. (2023, Section 2.3.1). We emphasize these two biases in particular because a) they are the most prevalent inductive biases that we find in the human behavioral data, and b) they are not currently well-explained by the Bayesian model. To evaluate whether the GNS model can capture these biases, we created

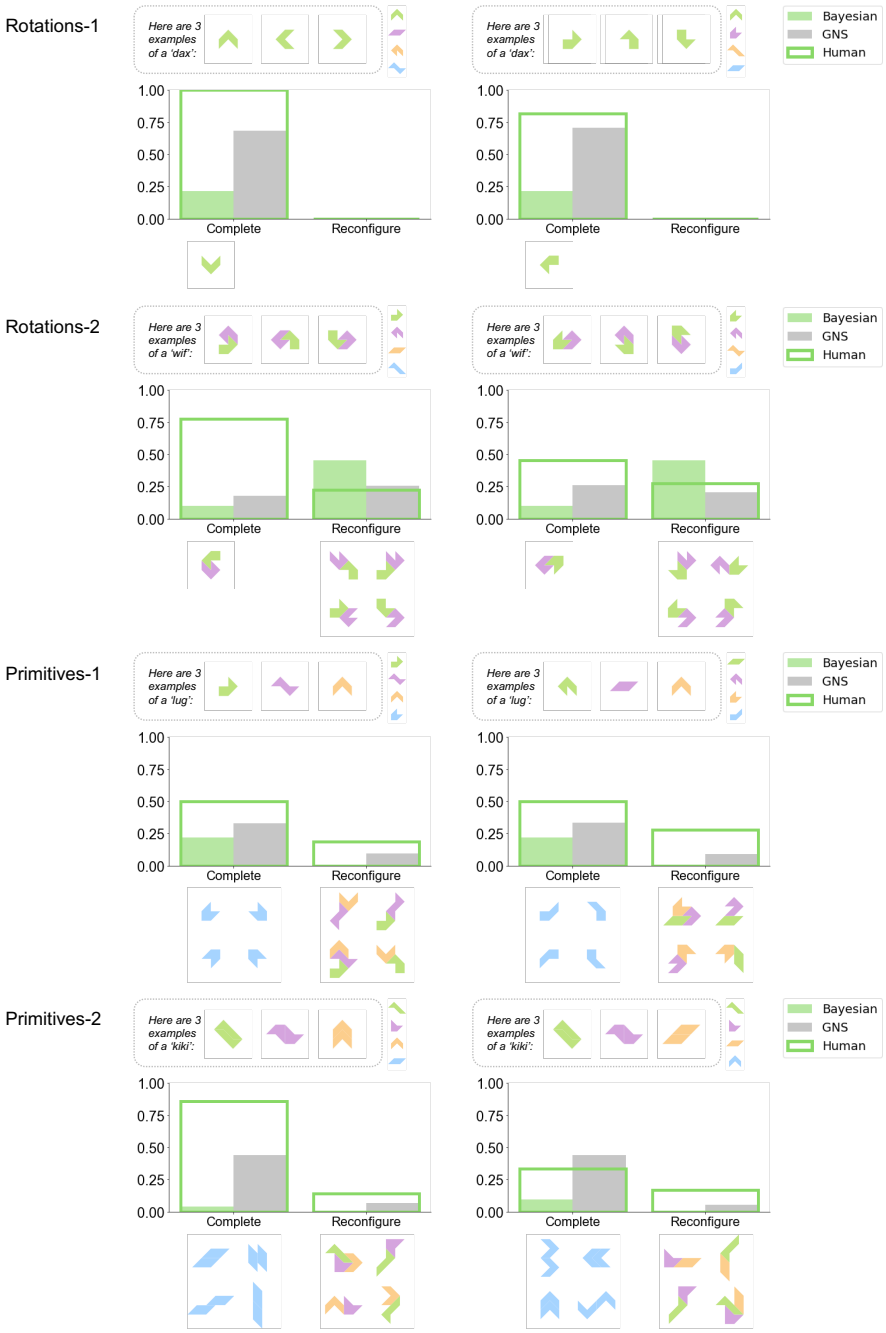


Figure 4.8: Inductive biases captured by the GNS and Bayesian models. Two trials are shown from each of four trial types with the partial-pattern property. Bars convey the marginal model probability of generating a new token that matches the target bias, and the empirical human frequency of doing so. In each trial, GNS exhibits a stronger completion bias vs. the Bayesian model that more closely matches human behavior. Moreover, the GNS model provides a closer match to human frequency for the *reconfigure bias*, assigning a non-zero probability where Bayes does not and showing a more modest probability where Bayes overpredicts.

a test set with all 19 trials that contain the partial-pattern property discussed, as well as 7 other randomly-selected trials from the generation task. We then trained the full GNS (P/R/H/C) model using only the remaining trials for the human distribution H. Fig. 4.8 shows the strength of the GNS model's inductive biases for a selection of test trials after training, comparing against both the Bayesian model and humans. The Rotations-N trial type consists of N-part tokens with a rotation pattern, and Primitives-N consists of N-part tokens with a primitive assignment pattern. People consistently exhibit a strong completion bias across different trial types, and the GNS model largely replicates this bias, showing a marginal probability for completion tokens that is often much closer to the human frequency compared with the Bayesian model. In addition, the GNS model's reconfigure bias matches humans in strength more closely than the Bayesian model, showing more accurate probabilities where the Bayesian model overpredicts in Rotations-2 trials, and where it underpredicts in Primitives-1 trials.

4.6 Discussion

This chapter demonstrates that generative neuro-symbolic (GNS) models can provide an effective means to understand and simulate human behavior in few-shot generation of structured visual concepts. When trained with a novel meta-learning scheme that mixes synthetic and real human data, our GNS model successfully mimics the symbolic Bayesian model and goes beyond to capture additional human biases that were not previously well explained. Our full GNS model shows a considerable improvement in likelihood of held-out participant data, and it provides an improved account for two salient inductive biases that participants exhibit: the *complete-the-pattern* bias and the *reconfigure* bias. In addition to these salient inductive biases, the GNS model also provides an account for a collection of one-off behaviors that do not fit into a larger bias category (Fig. 4.7).

Chapter 5

Learning inductive biases with simple neural networks

5.1 Preface

This chapter studies the acquisition of inductive biases in neural networks and is based off of our paper Feinman & Lake (2018). At the time of publication, a recent study had just come out highlighting exciting new connections between neural network models trained for object recognition and classical results from developmental psychology (Ritter et al., 2017). These results opened many new questions about whether and how neural networks learn in ways similar to humans. Our study builds from Ritter et al. (2017) and addresses a range of unanswered questions about how neural networks acquire inductive biases that are influential in child learning. Using a paradigm from developmental psychology Smith et al. (2002) we study the conditions required for neural networks to acquire the shape bias–a preference to organize objects by shape vs. other observed attributes–from stimuli containing basic patterns and shapes. These controlled stimuli allow us to systematically vary the quantity and form of the experience provided. We found that simple

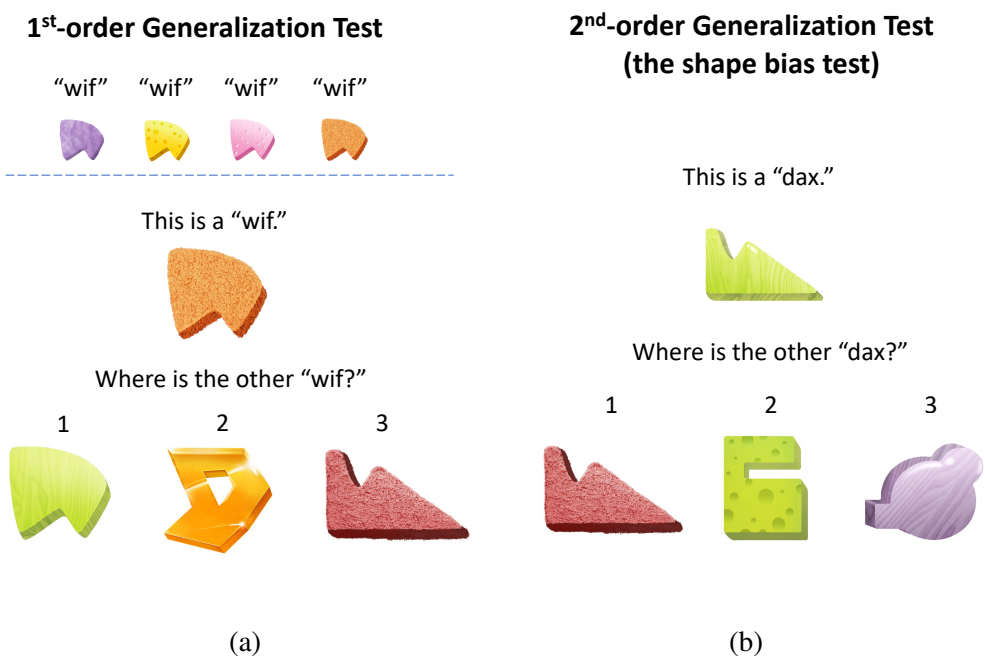
neural networks develop a shape bias after seeing as few as 3 examples of just 4 object categories. Moreover, the development of these biases predicts the onset of vocabulary acceleration in neural networks, consistent with the developmental processes in children.

This chapter is the first of two works studying how neural network models acquire inductive biases and use them to support future learning ("learning-to-learn"). Chapter 6 builds on this work and develops a new framework for modeling learning-to-learn in neural networks as hierarchical Bayesian inference.

5.2 Introduction

Humans possess the remarkable ability to learn a new concept from seeing just a few examples. A child can learn the meaning of a new word such as "fork" after observing only one or a handful of different forks (Bloom, 2000). In contrast, state-of-the-art artificial learning systems use hundreds or thousands of examples per class when learning to recognize the same objects (e.g., Krizhevsky et al., 2012; Szegedy et al., 2015). Consequently, significant effort is ongoing to understand what cognitive and neural mechanisms enable efficient concept learning (Lake et al., 2017). In this chapter, we perform a series of developmentally-informed neural network experiments to study the computational basis of efficient word learning. An open-source implementation of the experiments is available at `http://github.com/rfeinman/learning-to-learn`.

If a learner extrapolates beyond the data, then another source of information must make up the difference; prior knowledge or "inductive biases" must help constrain the space of models considered by the learner (Tenenbaum et al., 2011; Michalski et al., 2013; Lake et al., 2017). For example, children make use of the shape bias–the assumption that objects with the same name will tend to have the same shape–when learning new object names, and thus they attend to shape more often than color, material and other properties when generalizing a novel name to new examples (Fig. 5.1(b)) (Landau et al., 1988). Similarly, children assume that object names are mutually exclusive,



(a) (b)

Figure 5.1: Shape bias generalization tests. The 1st-order test, shown in (a), assesses if a child has learned to generalize a familiar object name to a novel exemplar according to shape. This is the first step of shape bias development. The 2nd-order test, shown in (b), assesses if the child has learned to generalize a novel name to a novel exemplar by shape, the second and final step of shape bias development.

i.e. that a novel name probably refers to a novel object rather than a familiar object (Markman & Wachtel, 1988). Although the origin of inductive biases is not always clear, results show that children, adults and primates can "learn-to-learn" or form higher-order generalizations that improve the efficiency of future learning (Harlow, 1949; Smith et al., 2002; Dewar & Xu, 2010).

Researchers have proposed a number of computational models to explain how inductive biases are acquired and harnessed for future learning. Hierarchical Bayesian Models (HBMs) enable probabilistic inference at multiple levels simultaneously, allowing the model to learn the structure of individual concepts while also learning about the structure of concepts in general (A. Gelman et al., 2013; Kemp et al., 2007; Salakhutdinov et al., 2012). These models have been used to explain various forms of "learning-to-learn," including learning a shape bias (Kemp et al., 2007). However,

it is currently difficult to apply HBMs to the type of high-dimensional visual and auditory stimuli that children receive; there have been successes (Salakhutdinov et al., 2013; Lake et al., 2015), but neural networks are still the most general solution to learning effectively from many different forms of raw data (LeCun et al., 2015). Utilizing this property, here we use neural networks to study learning-to-learn in different settings of varying stimulus complexity, with the goal of isolating the fundamentals of the learning dynamics.

Most related to our work here are studies by Colunga & Smith (2005) and Ritter et al. (2017) investigating neural network accounts of shape bias development. Colunga & Smith (2005) showed that a simple recurrent neural network, trained via Hebbian learning, can acquire a shape bias for solid objects and a material bias for non-solid objects. These simulations demonstrate that neural networks can form different expectations for different kinds of objects, but they raise many new questions regarding the conditions required to develop these types of biases. For example, the authors used highly simplified bit-vector data, and it is unclear whether their findings generalize to more complex or realistic stimuli. Furthermore, the authors did not systematically vary the quantity of experience provided to the networks, and thus we do not know the exact conditions in which biases arise and whether these networks can compete with the strong sample efficiency of HBMs (Kemp et al., 2007). In a recent study, Ritter et al. (2017) found that performance-optimized deep neural networks (DNNs) develop the shape bias when trained on the popular ImageNet object recognition dataset consisting of raw naturalistic images. These results highlight an exciting possible connection between large-scale DNNs and developmental psychology, though many questions still remain. ImageNet–which contains about 1200 labeled examples of 1000 different object categories–is a poor proxy for the experience of a developing child, who typically develops a shape bias with no more than 50-100 object words in her vocabulary (Gershkoff-Stowe & Smith, 2004). Whether these networks can acquire the shape bias with more appropriate training sets is unclear. Furthermore, although the development of the shape bias is known to predict the onset of vocabulary acceleration in children (Gershkoff-Stowe & Smith, 2004), we do not know whether the same holds for DNNs.

In a related study, Hill et al. (2017) trained a neural network agent to navigate around a virtual 3D world and collect objects according to name-based language commands. Although the authors draw inspiration from developmental psychology, the agent in this experiment is asked to learn a variety of tasks simultaneously: visual perception, language comprehension and navigation. Further work is necessary to isolate the dynamics of learning-to-learn in neural networks.

We investigate the development and influence of inductive biases in neural networks using artificial object stimuli that allow us to systematically vary the quality and form of the experience provided. Specifically, we use an experimental paradigm from developmental psychology (Smith et al., 2002) to train and evaluate the networks. Beginning with simple bit-vector data akin to Colunga & Smith (2005), we systematically vary the number of categories and the number of examples in the training set, each time evaluating the resulting network for two different forms of generalization (Fig. 5.1). Parallel experiments are then performed with raw image data, where each image consists of a 2D object with a particular shape, color and texture. For both the bit-vector and image stimuli, we analyze the perceptual similarity of our corresponding networks as a function of stimulus distance along shape and color dimensions, gauging the parametric sensitivities to these attributes. In a final set of experiments, we examine the dynamics of learning-to-learn by analyzing the relationship between shape bias acquisition and the rate of word learning, mirroring an analogous study from developmental psychology (Gershkoff-Stowe & Smith, 2004).

5.3 Experimental Paradigm

We set out to train neural networks with a learning paradigm used to guide toddlers to the shape bias (Smith et al., 2002). In this paradigm, the learner acquires new object names that are organized exclusively by shape, such that different instances of the same object category are identical in shape but contrast sharply in color and material. This is reflective of the fact that a child's early noun vocabulary consists predominantly of shape-based categories (Samuelson & Smith, 1999), although

not with the same purity as provided in the shape bias training. As in previous computational modeling work (Kemp et al., 2007; Colunga & Smith, 2005), we focus on purified training with shape-based categories, since it provides a controlled test of the artificial learner's ability to make higher-order generalizations across varying quantities of training experience.

In Smith et al. (2002), 17-month-old children were taught 4 new object names ("wif", "dax", etc.) over 7 weeks via weekly play sessions. Objects in the study were 3D formations constructed of various materials; each object contained a specific shape, color and texture (material), and their names were organized strictly by shape. During weekly sessions, children played with each object while an adult announced its name repeatedly. By the end of the study, the children had acquired a shape bias–i.e., they had formed the inductive bias that a novel name should be generalized by shape as opposed to color or texture. A control group of children who did not partake in the play sessions did not form this bias.

We use the training paradigm of Smith et al. (2002) to study inductive bias learning in neural networks with artificial object datasets. We first perform our computational experiments with abstract bit-vector stimuli, followed by experiments with raw image data. Each constructed object is assigned a shape, color and texture. We train simple neural networks to label objects with category names based on shape. To understand the necessary conditions for successful inductive bias learning, training is performed with various dataset sizes, varying both the number of categories and the number of examples of each category provided to the network. We evaluate the generalization capabilities of the network for each training set using 2 generalization tests modeled after the 2 tests of Smith et al. (2002), depicted in Fig. 5.1.

1st-order generalization test. For this test, toddlers are first presented with an exemplar object that they have seen during training ("wif" in Fig. 5.1a). Then, they are presented with 3 test objects that they have not seen: 1 that matches the exemplar in shape (item 1 in Fig. 5.1a), 1 that matches in color (item 2), and 1 that matches in texture (item 3). For each potential match, the other 2 stimulus attributes are novel. The toddlers are asked to select which of the 3 test objects share the same name

as the exemplar. Performance is measured as the fraction of trials in which the child selected the correct object, i.e. the shape match. To simulate this test, we create an evaluation set containing groupings of 4 sample objects: an exemplar, a shape match, a color match, and a texture match. The activations of our network's hidden layer are obtained in response to each object. We then evaluate the cosine similarity between the activations of the exemplar and each test object to determine which object the network perceives to be most similar. Accuracy is defined as the fraction of groupings for which the correct (shape-similar) object is chosen.

2nd-order generalization test. For this test, toddlers are first presented with an exemplar object that has a novel label (e.g., "teema") as well as a novel shape, color and texture. From there, the trial proceeds similarly to those of the 1st-order: a shape match, color match and texture match are presented, and the child must select which test object she believes to share a name with the exemplar. All shapes, colors and textures are novel to the child in this test. We simulate the 2nd-order test with artificial object stimuli similarly to the 1st-order case, again using last hidden layer features to evaluate perceptual similarity.

In all simulations, we record accuracy over 1000 simulated test trials as the performance metric for each generalization.

5.4 Experiment 1: Multilayer perceptron trained on synthetic objects

Our first experiment aims to study inductive bias learning in its purest form, using synthetic stimuli with maximal control. Objects are abstract binary patterns, divided into 3 input pools of 20 binary units each (representing the shape, color and texture of the objects; see Fig. 5.2). We varied the number of categories and number of examples per category in the training set. For datasets with N categories and K examples, we randomly generate N shape patterns, N color patterns, and N texture patterns. For all 3 attributes, each pattern is replicated K times, ensuring equal entropy

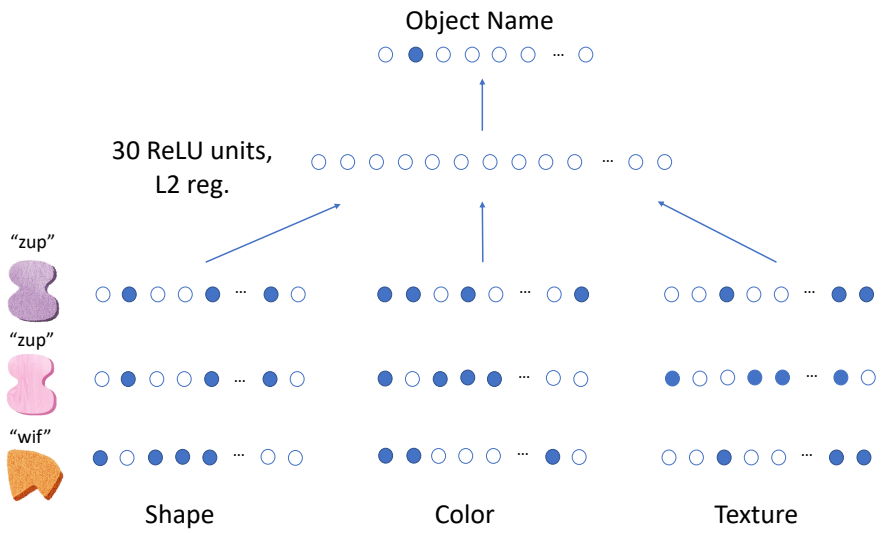


Figure 5.2: Multilayer perceptron architecture. Shape, color and texture attribute vectors are concatenated and fed to a 30-unit hidden layer, followed by a classification layer. 3 example input objects are shown (only one is presented at a time to the network).

across the 3. The shape patterns are then aligned with object labels, and the remaining 2 attributes are permuted randomly to create the dataset. A holdout set of shapes, colors and textures is retained for the generalization tests.

We train a multilayer perceptron (MLP) to name objects, as shown in Fig. 5.2. The network has an input layer of 60 units, a hidden layer of 30 rectified linear units (ReLUs) with L2 regularization, and a softmax output layer to classify the object by name. The softmax layer has N units (1 for each label). We train the network for 200 epochs using negative log-likelihood loss, RMSProp, and batch size min(32, $\frac{N*K}{5}$). For details about the selection of architectures and training parameters in Experiments 1 & 2, see Appendix C.

Results. Initially, as would be expected given the data format, shape is treated the same as other attributes. In the 2$^{\text{nd}}$-order generalization test, a randomly initialized network selects test objects with the following ratios, on average (50 trials): shape 35%, color 33% and texture 32%. We then trained the network with various dataset sizes. Results for the 1$^{\text{st}}$- and 2$^{\text{nd}}$-order generalizations are shown in Fig. 5.3, where each setup is an average over 10 networks with different random

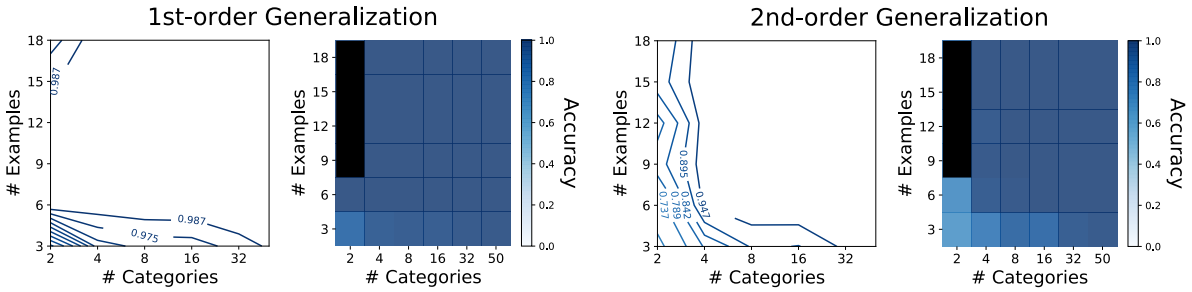
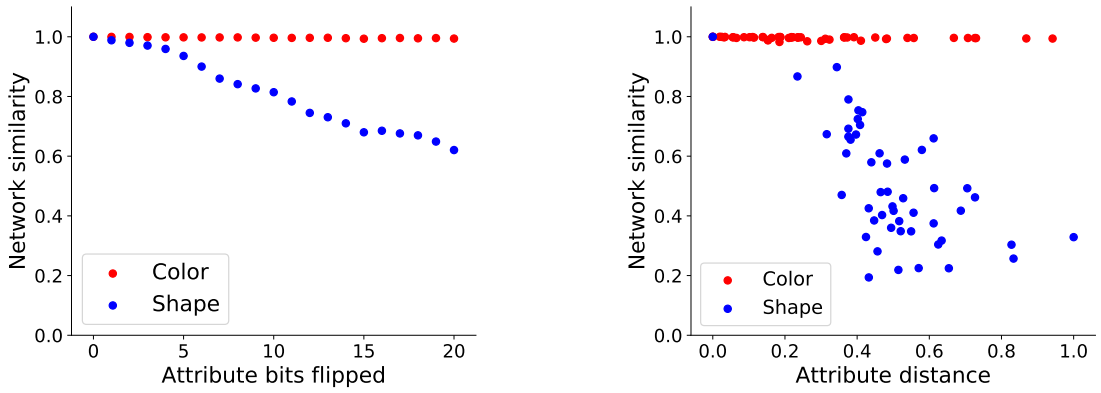


Figure 5.3: MLP generalization results for shape bias training with various training set sizes. The number of categories and number of examples per category provided to the network are shown on the x and y axes, respectively. Plots show accuracy over 1000 trials of the specified generalization test, averaged from 10 training runs. The same data is shown in both contour and heatmap format. With 2 categories, only 8 unique examples are feasible; thus, N/A results are blacked out.

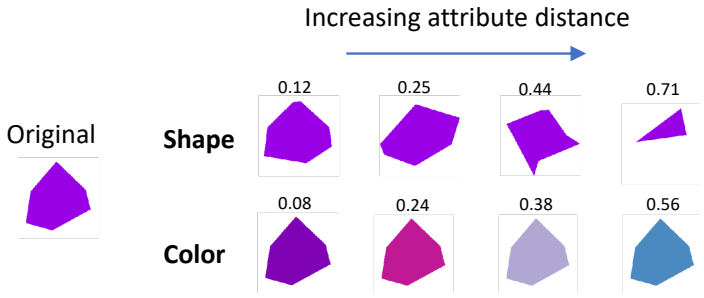
seeds. We note that acquisition of the 1st-order generalization requires less data than that of the 2nd-order, as predicted by the 2-phase hypothesis (Smith et al., 2002). Success in the 1st-order test indicates that the network is learning successfully and generalizing to new examples of the training classes. Networks that achieve an accuracy of 0.7 or higher on the 2nd-order test show a substantial shape bias, and the MLP reaches this threshold at the following points: N=2 & K=6 (accuracy 0.71) and N=4 & K=3 (accuracy 0.80). These results reproduce the general pattern of the Hierarchical Bayesian Model (HBM) in Kemp et al. (2007) and toddlers in Smith et al. (2002), who neared the 0.7 shape bias threshold with N=4 & K=2 (although the toddlers also receive external experience). In contrast, Colunga & Smith (2005) used N=10 & K=100 to obtain the shape bias in their networks, using similar abstract patterns. Although HBMs are often noted for their data efficiency, in this case, the neural network was competitive for making 2nd-order generalizations from limited data.

As another way of demonstrating the learned sensitivity to shape, we perform parametric manipulations of the stimuli. Using an MLP trained with N=4 & K=6, we probe the shape bias by selecting a novel test stimulus and systematically flipping bits, recording the network similarity between the modified stimulus and the original. For comparison, a similar test is also performed



(a) MLP

(b) CNN



(c) Distance along stimulus dimensions

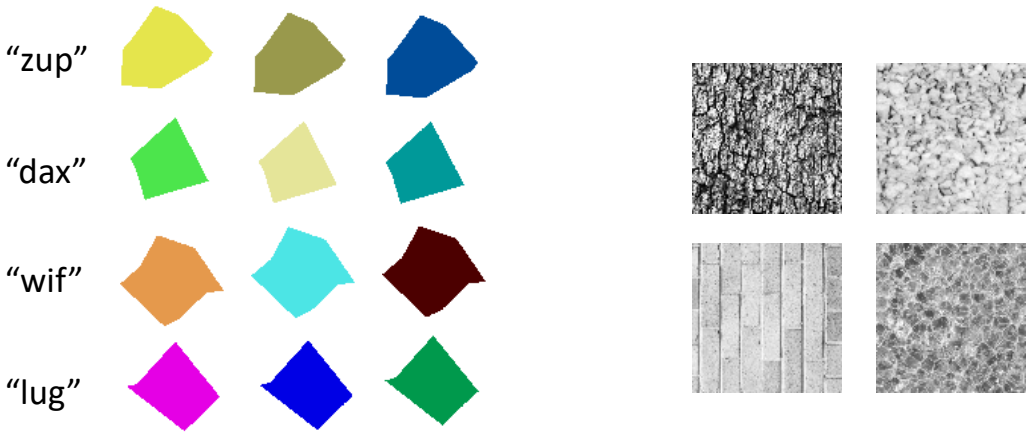
Figure 5.4: Perceptual (network) similarity as a function of physical (attribute) distance. A test stimulus is systematically altered along either its shape or color dimension. Network similarity scores are computed between the original stimulus and its altered counterpart.

with color. Results are shown in Fig. 5.4(a) for 1 test stimulus. Clearly, the network is far more sensitive to changes in shape than changes in color.

5.5 Experiment 2: Convolutional network trained on synthetic objects

Our first experiment used highly simplified training stimuli for maximal experimental control. One strength of modern neural network architectures is that they can learn effectively from data in

raw and complex forms, a fact we take advantage of in developing Experiment 2. Here we ask whether similar learning-to-learn results can be achieved using synthetic object stimuli presented as raw images. This setup presents a more challenging learning problem for the neural network, in terms of making both 1st- and 2nd-order generalizations, since understanding shape requires making abstractions that go substantially beyond separating a pool of input units that directly encode the dimension, as in Experiment 1.



(a) (b)

Figure 5.5: Training stimuli for Experiment 2. (a) novel objects with various shapes and colors (the first 3 input channels). (b) a few examples of textures that might be found in the 4th input channel.

The stimuli are constructed as follows. Each object is a 2D shape of a specified color placed over white background (200×200). Texture is represented in a fourth image channel, independent of RGB space. This design choice was made to avoid an initial shape bias; with texture overlaid in RGB, a randomly initialized network exhibits the shape bias. Furthermore, the participants in Smith et al. (2002) physically touch each object during play, indicating that they have access to additional non-visual information.

Examples of our objects are shown in Fig. 5.5. Object shapes are polygons of random order (uniform 3-10) and randomly sampled vertices, with preference given to points near image boundaries in order to ensure visible-sized objects. Colors are generated to span the RGB vector space with even separation. We use black and white textures from the Brodatz database (Brodatz, 1966)

for our texture categories. A holdout set of shapes, colors and textures is again retained for testing.

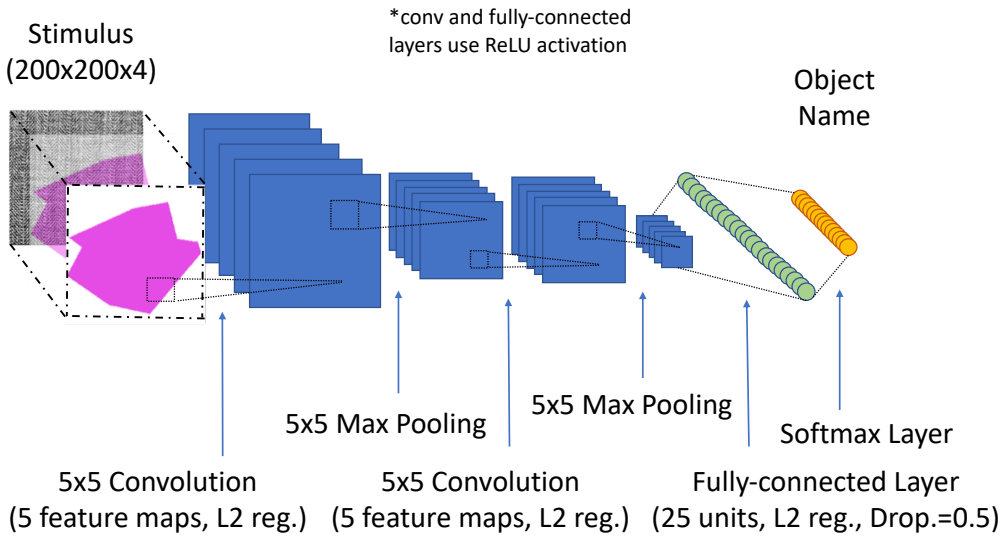


Figure 5.6: Convolutional network architecture. The network receives 4-channel image stimuli and is trained to label the object in the image with a category name that is based on shape.

We train a multi-layer convolutional neural network (CNN) (LeCun et al., 2015) consisting of two convolution layers with five feature maps, each followed by a max pooling layer. A depiction of this architecture is shown in Fig. 5.6. The last pooling layer is followed by a fully-connected layer of 25 ReLU units, and the softmax layer again varies in size according to the number of categories. Both the convolutional layers and the fully-connected layer use L2 regularization, the latter also with dropout=0.5. Each object is randomly shifted around image space by a small offset (train and test alike). Training details mimic the MLP, but with 400 epochs.

Results. The randomly initialized network makes 2nd-order selections with the following ratios: shape 38%, color 42% and texture 20%. We trained the network using varying dataset sizes, as with our MLP. Results are shown in Fig. 5.7. Similarly to the MLP, acquisition of the 1st-order generalization requires less data than that of the 2nd-order, supporting the notion that learning the training classes is a simpler task than forming higher-order generalizations. Using the same shape bias threshold of 0.7 2nd-order score, we find a number of important transition points: N=32 &

K=3 (accuracy 0.74), N=8 & K=6 (accuracy 0.75), and N=4 & K=12 (accuracy 0.70). The CNN is thus capable of learning a shape bias from as few as 6 examples of 8 categories, a significant feat given the scale of the input. Notably, the network is able to learn this bias with much less data than Colunga & Smith (2005) using a data form that is significantly more complex. The CNN of Ritter et al. (2017) used roughly N=1000 & K=1200, and developed a shape bias of 0.68 on a shape and color-only task. A key takeaway from our results is that, with concentrated training effort, it is possible to learn this bias from much less data using high-dimensional color images.

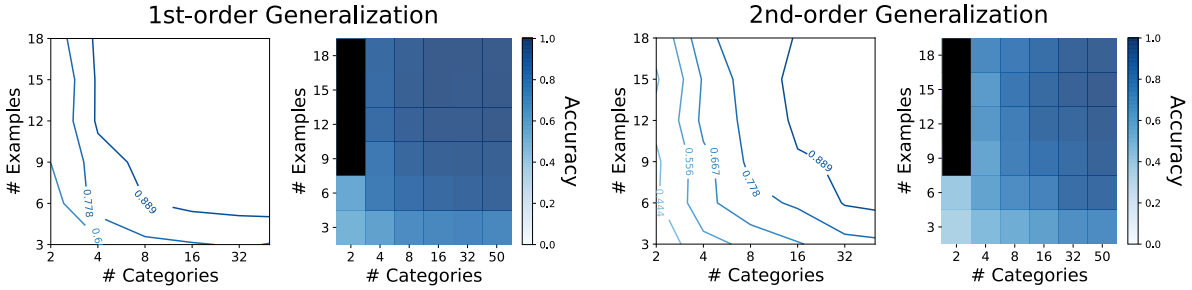


Figure 5.7: CNN generalization results for shape bias training with various training set sizes. Results show the average of 10 training runs. See Fig. 5.3 for details.

As in Experiment 1, we also parametrically manipulate the CNN's input to analyze its sensitivity to changes along different stimulus dimensions, using a CNN trained with N=30 & K=10. Distance in shape space is quantified as the Modified Hausdorff Distance (Dubuisson & Jain, 1994) between the shape pair, and in color space as the cosine similarity of the RGB vector pair. Beginning with an exemplar object stimuli, we sample 50 secondary shapes and order them by their distance from the exemplar. We then modify the shape of the exemplar parametrically by stepping along this list, recording network similarities between the original and modified versions in each case. A mirroring experiment is then performed with color; in each case, only 1 attribute is altered at a time. Results are shown in Fig. 5.4(b) & 5.4(c). We find that, much like the MLP, our CNN shows a strong sensitivity to shape but not color.

For the sake of comparison, in Experiment 2 we also trained our CNN to label objects with

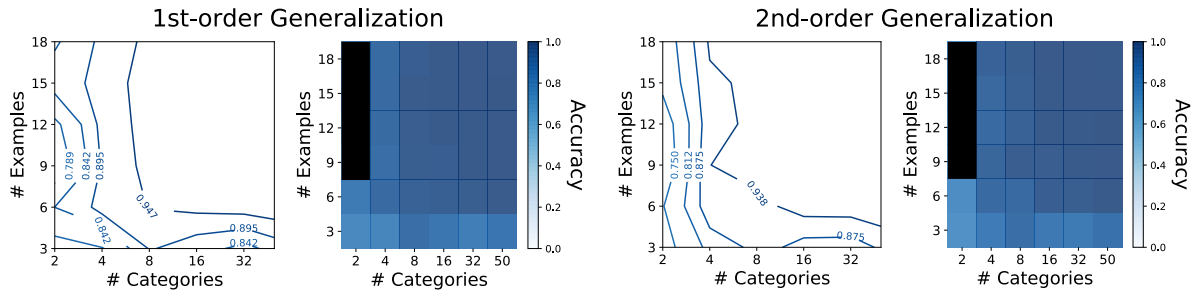
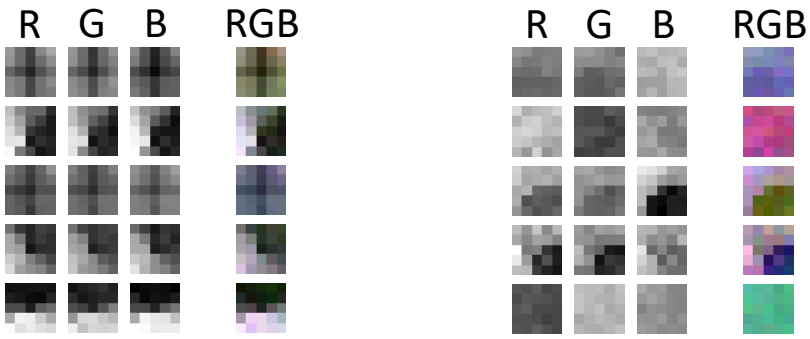


Figure 5.8: CNN generalization results for color bias training with various training set sizes. The network is trained to label objects with category names based on color. In this case, the generalization tests evaluate the fraction of times that the color match is selected. Results in each grid show the average of 10 training runs.

names organized by color. Our goal was to compare the required sample complexity for color bias training with that of shape bias training, and to evaluate whether color bias development follows a similar 2-step process. All dataset parameters mirrored those of shape training, except that the object labels were aligned with the color attribute of each training image. Performance on the generalization tests was measured as the fraction of trials for which the network selects the color match. Results for CNN color bias training are shown in Fig. 5.8. Notably, the color-trained CNN requires a smaller sample complexity to achieve 0.7 accuracy on the 2$^{\text{nd}}$-order test, reaching a score of 0.73 with N=2 & K=3. Furthermore, this network does not appear to follow the 2-step process of bias development; results for 1$^{\text{st}}$- and 2$^{\text{nd}}$-order generalizations look near-identical to one another. In order to identify a stimulus as a member of a particular color category, the network needs only to find a single pixel of that color, a task that is much simpler than representing and identifying shape. Representing color requires a simple 3D space. By learning to isolate and preserve this space in the hidden layers, the network can easily generalize to novel colors, hence the early 2$^{\text{nd}}$-order results.

We inspected the learned representations of both a shape-trained and a color-trained CNN, trained with N=50 & K=18, by visualizing the first-layer convolution filters of each network (Fig. 5.9). As we would expect, filters of the shape-trained CNN look identical across R, G and B channels, as this network needs no sensitivity to color. In contrast, filters of the color-trained CNN



(a) Shape (b) Color

Figure 5.9: Visualizing RGB channels of learned first-layer convolution filters. (a) Filters from the CNN trained with explicit shape bias training (N=50 & K=18). Each row corresponds to 1 of the 5 filters. The first 3 channels are shown in the 'R', 'G' and 'B' columns, respectively. These 3 channels are shown together in a 4th column, labeled 'RGB'. (b) Filters from the CNN trained to label objects with category names based on color. In both (a) and (b), only channels 1-3 of the 4 are shown.

vary across channels, indicating that the network has learned a selectivity for color.

5.6 Experiment 3: The onset of vocabulary acceleration

Our previous experiments confirm that simple neural networks can develop the shape bias from a relatively small number of categories and examples. It remains unclear, however, how the dynamics of bias acquisition relate to the dynamics of word learning. Gershkoff-Stowe & Smith (2004) showed that the development of the shape bias in toddlers predicts the onset of vocabulary acceleration during early word learning, a phase that begins at ages 16-20 months. Studying 8 children during regular lab sessions at 3-week intervals, the authors found that increasing attention to shape was correlated with increasing rate of vocabulary acquisition in participants. Fig. 5.11(a) shows the individual growth curves of vocabulary size and shape response for each child. The former variable is measured as the cumulative number of nouns in the child's vocabulary, and the latter as the cumulative number of times that the child has selected the shape match in a shape bias task akin to the 2nd-order test. Although the vocabulary curve shows cumulative nouns in whole, the authors

also recorded cumulative “count nouns" for each participant, a subset of nouns that is well organized by shape. We focus on the statistics reported for count nouns, as this subset reflects the type of vocabulary that is influenced by the shape bias.

The authors found a few interesting correlations: **1)** a correlation between increase in cumulative shape choices and increase in cumulative count nouns across sessions for an individual participant, averaged over participants [average $r = .75$; $p < .05$ for each], and **2)** a correlation between average increase in shape choices over the whole experiment and average increase in count nouns, computed across participants [$r = .81$; $p < .05$].

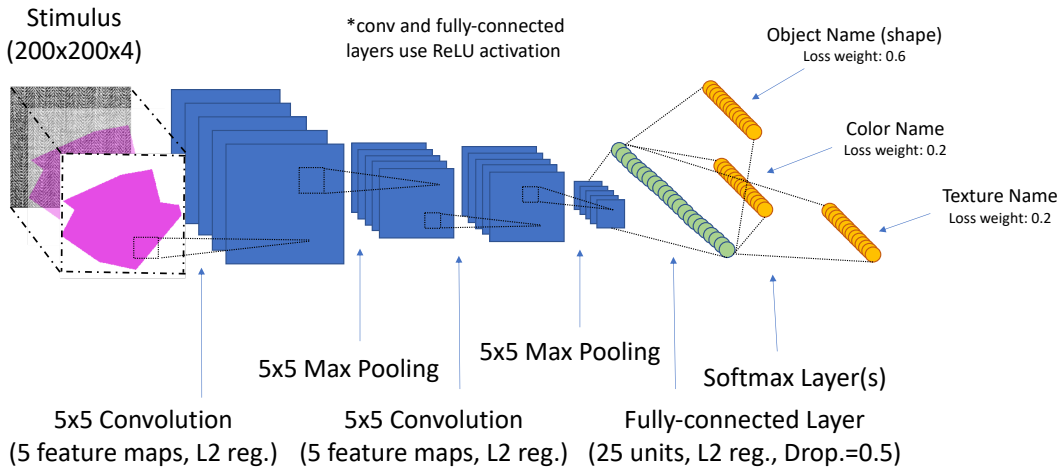


Figure 5.10: CNN architecture for Experiment 3. The architecture mimics the original CNN of Experiment 2, with the exception of the softmax layer. Here, there are 3 softmax layers (1 for each shape, color and texture), each of which extends from the fully-connected layer.

Methods. Inspired by this study, we train a CNN using our raw image data with the goal of evaluating related correlation metrics for our networks. The participants of Gershkoff-Stowe & Smith (2004) were not explicitly trained for the shape bias like those of Smith et al. (2002); they received natural experience in a home setting, which may have included some words organized by attributes other than shape. Therefore, we train our CNN to simultaneously label the object’s name, which correlates with shape, as well as its color and texture names. We use a modified version of

the CNN architecture from Section 5.5 with three softmax layers, one for each label dimension (Fig. 5.10). Each softmax layer branches independently from the fully-connected layer and has its own negative log-likelihood loss. The number of categories along each label dimension, and the loss weight assigned to its softmax layer, are determined according to the natural statistics of the early human lexicon (Samuelson & Smith, 1999).[1] The chosen ratios are as follows: 60-20-20 shape-color-texture names (36, 12 and 12 categories, respectively). Thus, the overall loss is $L = 0.6 \cdot L_{shape} + 0.2 \cdot L_{color} + 0.2 \cdot L_{texture}$. We use 10 examples of each shape, and colors and textures are assigned at random to each stimuli from their 12 categories. We keep a cumulative count of the number of count nouns in the network's vocabulary, defined as the number of shape categories for which the network has achieved 80% or greater accuracy on the training set. We also keep a cumulative count of shape choices the network makes in a 500-trial 2$^{\text{nd}}$-order test. This process is repeated with 20 networks, using a different random seed for each network.

Results. We inspect the "early" word learning period for our networks, defined as the period in which the average vocabulary size across the 20 networks is less than or equal to $2/3$ the total number of count nouns. Beyond this period, which we find to include the first 30 training epochs, the network's learning begins to flatten. We divide this period into 10 "sessions," evenly spaced by 3 epochs. The learning curves of our networks are shown in Fig. 5.11(b). We compute correlation metrics for our networks that are analogous to those of the child study. Looking at increases across the sessions of a single network (metric **1**), we find an average correlation of $r = .53$ between increase in cumulative shape choices and increase in cumulative count nouns [$p < .05$ for each]. Further, looking at average increases across the entire 10-session period for each network (metric **2**), we find a correlation of $r = 0.76$ [$p < .001$] across the 20 networks.

These analyses confirm that the dynamics of shape bias acquisition and early word learning show a considerable dependency on one another in our CNNs, a phenomenon that is mirrored in the

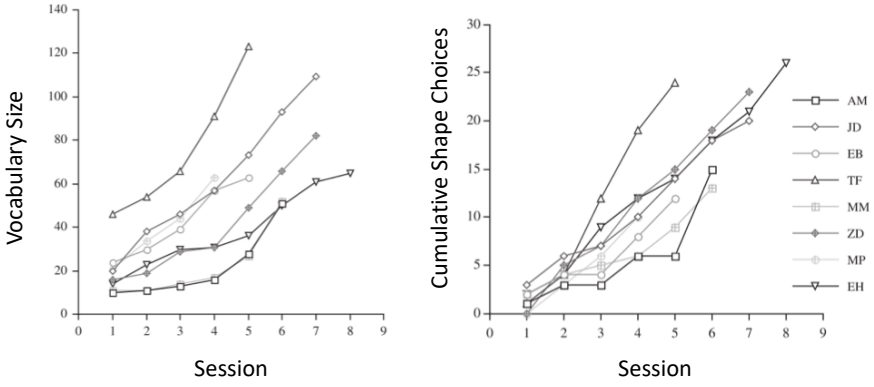
[1] Children are taught object, color and material names independently. Loss weighting provides a good analog to this for CNN training. Assigning a weight of 0.6 to object name labeling mirrors presenting this type of name 60% of the time in training.

early word learning of human children.

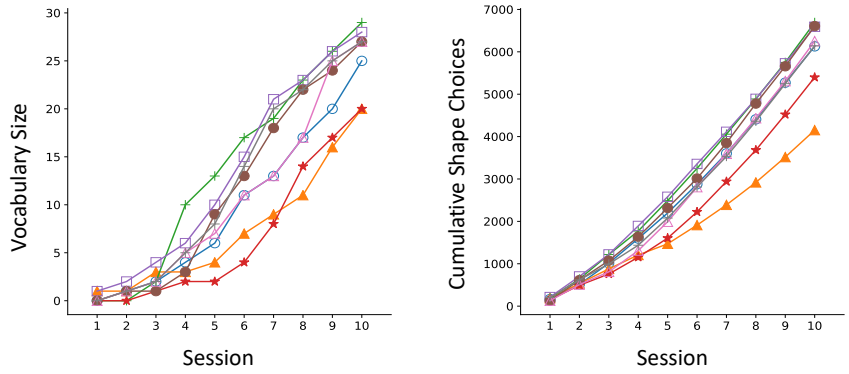
5.7 Discussion

Using a set of controlled synthetic experiments, our work provides novel insights about the environmental conditions that enable learning-to-learn in neural networks. Building on the work of Colunga & Smith (2005) and Ritter et al. (2017), Experiment 1 showed that simple neural networks can learn a shape bias from stimuli presented as abstract bit patterns with as few as 3 examples of 4 categories. Experiment 2 showed that simple convolutional neural net architectures trained on high-dimensional images can learn a shape bias with as few as 6 examples of 8 object categories. Although Hierarchical Bayesian Models (HBMs) are often noted for their data efficiency, our results indicate that neural networks can approach both HBMs (Kemp et al., 2007) and children (Smith et al., 2002) in the amount of data required to develop a shape bias. Moreover, we show that the complexity of the data (e.g., binary patterns vs. synthetic images) influences the dynamics of learning, and that neural networks are a powerful tool for understanding these types of interactions.

The development of the shape bias in children is known to correlate with accelerated word learning (Gershkoff-Stowe & Smith, 2004), a phenomenon that Experiment 3 confirmed can be mirrored in neural networks. One implication of this finding is that it may be possible to train large-scale image recognition models more efficiently after initializing these models with shape bias training. In future work, we hope to investigate this hypothesis with ImageNet-scale DNNs, using an initialization framework designed with the intuitions garnered here.



(a) Children



(b) CNN

Figure 5.11: Learning curves for shape bias and vocabulary. (a) shows the learning curves of the 8 children participants from Gershkoff-Stowe & Smith (2004). Participants were studied over the course of 5-8 lab sessions. Curves are shown for vocabulary size (left) and cumulative shape choices (right). Here, vocabulary includes all noun types. (b) shows analogous plots for our CNNs. 8 networks are shown, randomly sampled from the total 20 for the sake of visibility. Here, vocabulary is measured only for shape-based object names.

Chapter 6

Learning a smooth kernel regularizer for convolutional neural networks

6.1 Preface

This chapter is based off of Feinman & Lake (2019) and is the second of two works studying learning-to-learn and inductive bias acquisition in neural network models. The first work, presented in Chapter 5, studied inductive biases that arise naturally with existing architectures and algorithms. This chapter presents a modification of the standard neural network toolkit that incorporates new priors and new techniques for learning-to-learn. The learned weights of convolutional neural networks (CNNs) trained on large datasets for object recognition contain a substantial amount of structure. These representations have parallels to simple cells in the primary visual cortex, where receptive fields are smooth and contain many regularities. Incorporating smoothness constraints over the kernel weights of modern CNN architectures is a promising way to improve their sample complexity. In this chapter, we propose a smooth kernel regularizer that encourages spatial correlations in convolution kernel weights. The correlation parameters of this regularizer are learned from previous

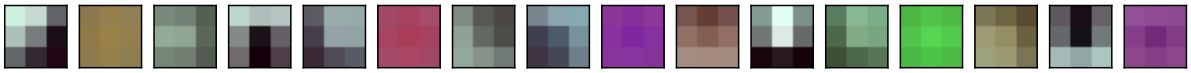
experience, yielding a method with a hierarchical Bayesian interpretation. We show that our correlated regularizer can help constrain models for visual recognition, improving over an L2 regularization baseline.

6.2 Introduction

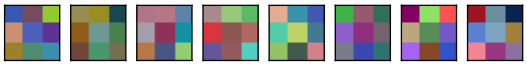
Convolutional neural networks (CNNs) are powerful feed-forward architectures inspired by mammalian visual processing capable of learning complex visual representations from raw image data (LeCun et al., 2015). These networks achieve human-level performance in some visual recognition tasks; however, their performance often comes at the cost of hundreds or thousands of labelled examples. In contrast, children can learn to recognize new concepts from just one or a few examples (Bloom, 2000; F. Xu & Tenenbaum, 2007), evidencing the use of rich structural constraints (Lake et al., 2017). By enforcing structure on neural networks to account for the regularities of visual data, it may be possible to substantially reduce the number of training examples they need to generalize. In this paper, we introduce a soft architectural constraint for CNNs that encourages smooth, correlated structure on their convolution kernels through transfer learning. We see this as an important step towards a general, off-the-shelf CNN regularizer that operates independently of previous experience. We have released an open-source implementation of the experiments at `https://github.com/rfeinman/SK-regularization`.

The basis for our constraint is the idea that the weights of a convolutional kernel should in general be well-structured and smooth. The weight kernels of CNNs that have been trained on the large-scale ImageNet object recognition task contain a substantial amount of structure. These kernels have parallels to simple cells in primary visual cortex, where smooth receptive fields implement bandpass oriented filters of various scale (Jones & Palmer, 1987).

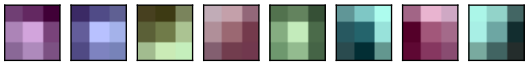
The consistencies of visual receptive fields are explained by the regularities of image data. Locations within the kernel window have parallels to locations in image space, and images are



(a) VGG16 layer-1 kernels



(b) i.i.d. Gaussian (L2-reg)



(c) correlated Gaussian (SK-reg)

Figure 6.1: Kernel priors for VGG16. The layer-1 convolution kernels of VGG16, shown in (a), possess considerable correlation structure. An i.i.d. Gaussian prior that has been fit to the VGG layer-1 kernels, samples from which are shown in (b), captures little of the structure in these kernels. A correlated multivariate Gaussian prior, samples from which are shown in (c), captures the correlation structure of these kernels well.

generally smooth (Li, 2009). Consequently, smooth, structured receptive fields are necessary to capture important visual features like edges. In landmark work, Hubel & Wiesel (1962) discovered edge-detecting features in the primary visual cortex of cat. Since then, the community has successfully modeled receptive fields in early areas of mammalian visual cortex using Gabor kernels (Jones & Palmer, 1987). These kernels are smooth and contain many spatial correlations. In later stages of visual processing, locations of kernel space continue to parallel image space; however, inputs to these kernels are visual features, such as edges. Like earlier layers, these layers also benefit from smooth, structured kernels that capture correlations across the input space. Geisler et al. (2001) showed that human contour perception–an important component of object recognition–is well-explained by a model of edge co-occurrences, suggesting that correlated receptive fields are useful in higher layers of processing as well.

Despite the clear advantages of structured receptive fields, constraints placed on the convolution kernels of CNNs are typically chosen to be as general as possible, with neglect of this structure. L2 regularization–the standard soft constraint applied to kernel weights, which is interpreted as a zero-mean, independent identically distributed (i.i.d.) Gaussian prior–treats each weight as an independent random variable, with no correlations between weights expected a priori. Fig. 6.1

shows the layer-1 convolutional kernels of VGG16, a ConvNet trained on the large-scale ImageNet object recognition task (Simonyan & Zisserman, 2015). Fig. 6.1(b) shows some samples from an i.i.d. Gaussian prior, the equivalent of L2 regularization. Clearly, this prior captures little of the correlation structure possessed by the kernels.

A simple and logical extension of the i.i.d. Gaussian prior is a correlated multivariate Gaussian prior, which is capable of capturing some of the covariance structure in the convolution kernels. Fig. 6.1(c) shows some samples from a correlated Gaussian prior that has been fit to the VGG16 kernels. This prior provides a much better model of the kernel distribution. In this paper, we perform a series of controlled CNN learning experiments using a smooth kernel regularizer–which we denote "SK-reg"–based on a correlated Gaussian prior. The correlation parameters of this prior are obtained by fitting a Gaussian to the learned kernels from previous experience. We compare SK-reg to standard L2 regularization in two object recognition use cases: one with simple silhouette images, and another with Tiny ImageNet natural images. In the condition of limited training data, SK-reg yields improved generalization performance.

6.3 Background

Our goal in this paper is to introduce new a priori structure into CNN receptive fields to account for the regularities of image data and help reduce the sample complexity of these models. Previous methods from this literature often require a fixed model architecture that cannot be adjusted from task to task. In contrast, our method enforces structure via a statistical prior over receptive field weights, allowing for flexible architecture adaption to the task at hand. Nevertheless, in this section we review the most common approaches to structured vision models.

A popular method to enforce structure on visual recognition models is to apply a fixed, pre-specified representation. In computational vision, models of image recognition consist of a hierarchy of transformations motivated by principles from neuroscience and signal processing (e.g., Serre

et al., 2007; Bruna & Mallat, 2013). These models are effective at extracting important statistical features from natural images, and they have been shown to provide a useful image representation for SVMs, logistic regression and other “shallow" classifiers when applied to recognition tasks with limited training data. Unlike CNNs, the kernel parameters of these models are not learned by gradient descent. As result, these features may not be well-adapted to the specific task at hand.

In machine learning, it is commonplace to use the features from CNNs trained on large object recognition datasets as a generic image representation for novel vision tasks (Donahue et al., 2014; Razavian et al., 2014). Due to the large variety of training examples that these CNNs receive, the learned features of these networks provide an effective representation for a range of new recognition tasks. Some *meta-learning* algorithms use a similar form of feature transfer, where a feature representation is first learned via a series of classification episodes, each with a different support set of classes (e.g., Vinyals et al., 2016). As with pre-specified feature models, the representations of these feature transfer models are fixed for the new task; thus, performance on the new task may be sub-optimal.

Beyond fixed feature representations, other approaches use a pre-trained CNN as an initialization point for a new network, following with a fine-tuning phase where network weights are further optimized for a new task via gradient descent (e.g., Girshick et al., 2014; Girshick, 2015). By adapting the CNN representation to the new task, this approach often enables better performance than fixed feature methods; however, when the scale of the required adaptation is large and the training data is limited, fine-tuning can be difficult. Finn et al. (2017) proposed a modification of the pre-train/fine-tune paradigm called model-agnostic meta-learning (MAML) that enables flexible adaptation in the fine-tuning phase when the training data is limited. During pre-training (or *meta-learning*), MAML optimizes for a representation that can be easily adapted to a new learning task in a later phase. Although effective for many use cases, this approach is unlikely to generalize well when the type of adaptation required differs significantly from the adaptations seen in the meta-learning episodes. A shared concern for all pre-train/fine-tune methods is that they require a

fixed model architecture between the pre-train and fine-tune phases.

The objective of our method is distinct from those of fixed feature representations and pre-train/fine-tune algorithms. In this paper, we study the structure in the learned parameters of vision models, with the aim of extracting general structural principles that can be incorporated into new models across a broad range of learning tasks. SK-reg serves as a parameter prior over the convolution kernels of CNNs and has a theoretical foundation in Bayesian parameter estimation. This approach facilitates a CNN architecture and representation that is adapted to the specific task at hand, yet that possesses adequate structure to account for the regularities of image data. The SK-reg prior is learned from previous experience, yielding an interpretation of our algorithm as a method for hierarchical Bayesian inference.

Independently of our work, Atanov et al. (2019) developed the *deep weight prior*, an algorithm to learn and apply a CNN kernel prior in a Bayesian framework. Unlike our prior, which is parameterized by a simple multivariate Gaussian, the deep weight prior uses a sophisticated density estimator parameterized by a neural network to model the learned kernels of previously-trained CNNs. The application of this prior to new learning tasks requires variational inference with a well-calibrated variational distribution. Our goal with SK-reg differs in that we aim to provide an interpretable, generalizable prior for CNN weight kernels that can be applied to existing CNN training algorithms with little modification.

6.4 Bayesian interpretation of regularization

From the perspective of Bayesian parameter estimation, the L2 regularization objective can be interpreted as performing *maximum a-posteriori* inference over CNN parameters with a zero-mean, i.i.d. Gaussian prior. Here, we review this connection, and we discuss the extension to SK-reg.

L2 regularization. Assume we have a dataset $X = \{x_1, ..., x_N\}$ and $Y = \{y_1, ..., y_N\}$ consisting of N images x_i and N class labels y_i. Let θ define the parameters of the CNN that we

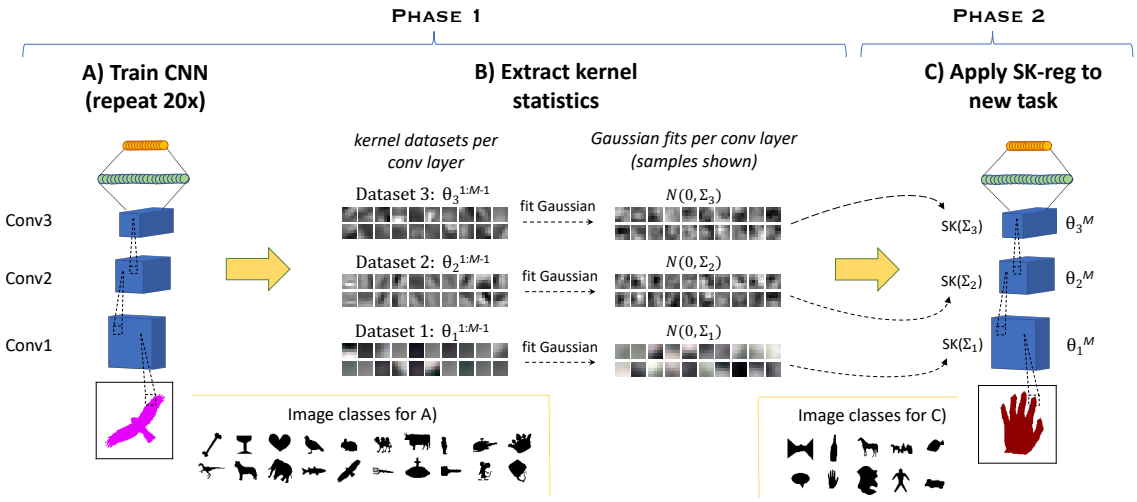


Figure 6.2: SK-reg workflow. A) First, a CNN is trained repeatedly (20x) on an object recognition task. B) Next, the learned parameters of each CNN are studied and statistics are extracted. For each convolution layer, kernels from the multiple CNNs are consolidated, yielding a kernel dataset for the layer. A multivariate Gaussian is fit to each kernel dataset. C) SK-reg is applied to a fresh CNN trained on a new learning task with limited training data (possibly with a different architecture or numbers of kernels), using the resulting Gaussians from each layer.

wish to estimate. The L2 regularization objective is stated as follows:

$$\tilde{\theta} = \arg\max_{\theta} \log p(Y \mid \theta; X) - \lambda * \theta^T \theta. \tag{6.1}$$

Here, the first term of our objective is our prediction accuracy (classification log-likelihood), and the second term is our L2 regularization penalty.

From a Bayesian perspective, this objective can be thought of as finding the *maximum a-posteriori* (MAP) estimate of the network parameter posterior $p(\theta \mid Y; X) \propto p(Y \mid \theta; X) * p(\theta)$, leading to the optimization problem

$$\tilde{\theta} = \arg\max_{\theta} \log p(Y \mid \theta; X) + \log p(\theta). \tag{6.2}$$

To make the connection with L2 regularization, we assume a zero-mean, i.i.d Gaussian prior over

the parameters θ of a weight kernel, written as

$$p(\theta) = \frac{1}{Z}\exp\big(-\frac{1}{2\sigma^2}\theta^T\theta\big). \tag{6.3}$$

With this prior, Eq. 6.2 becomes

$$\tilde{\theta} = \arg\max_{\theta} \log p(Y \mid \theta; X) - \frac{1}{2\sigma^2}\theta^T\theta,$$

which is the L2 objective of Eq. 6.1, with $\lambda = \frac{1}{2\sigma^2}$.

SK regularization. The key idea behind SK-reg is to extend the L2 Gaussian prior to include a non-diagonal covariance matrix; i.e., to add correlation. In the case of SK-reg, the prior over kernel weights θ of Eq. 6.3 becomes

$$p(\theta) = \frac{1}{Z}\exp\big(-\frac{1}{2}\theta^T\Sigma^{-1}\theta\big)$$

for some covariance matrix Σ, and the new objective is written

$$\tilde{\theta} = \arg\max_{\theta} \log p(Y \mid \theta; X) - \lambda * \theta^T\Sigma^{-1}\theta. \tag{6.4}$$

Hierarchical Bayes. When Σ is learned from previous experience, SK-reg can be interpreted as approximate inference in a hierarchical Bayesian model. The SK regularizer for a CNN with C layers, $\boldsymbol{\Sigma} = \{\Sigma_1, \ldots, \Sigma_C\}$, assumes a unique zero-mean Gaussian prior $\mathcal{N}(\theta_i; 0, \Sigma_i)$ over the weight kernels for each convolutional layer, $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_C\}$. Due to the regularities of the visual world, it is plausible that effective general priors exist for each layer of visual processing. In this paper, transfer learning is used to fit the prior covariances $\boldsymbol{\Sigma}$ from previous datasets $X^{1:M-1}$ and $Y^{1:M-1}$, which informs the solution for a new problem X^M and Y^M, yielding the hierarchical Bayesian interpretation depicted in Fig. 6.3. Task-specific CNN parameters $\boldsymbol{\theta}^{1:M}$ are drawn from a common

$\boldsymbol{\Sigma}$, and $\boldsymbol{\Sigma}$ has a hyperprior specified by β. Ideal inference would compute $p(Y^M|Y^{1:M-1}; X^{1:M})$, marginalizing over $\boldsymbol{\theta}^{1:M}$ and $\boldsymbol{\Sigma}$.

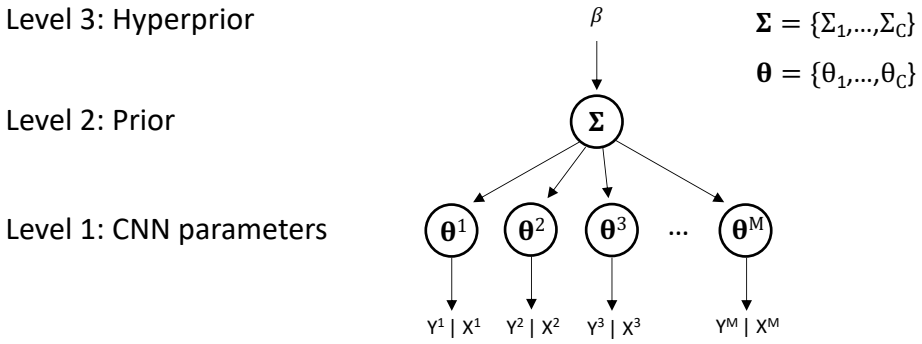


Figure 6.3: A hierarchical Bayesian interpretation of SK-reg. A point estimate of prior parameters $\boldsymbol{\Sigma}$ is first computed with MAP estimation. Next, this prior is applied to estimate CNN parameters $\boldsymbol{\theta}^j$ in a new task.

We propose a very simple empirical Bayes procedure for learning the kernel regularizer in Eq. 6.4 from data. First, $M-1$ CNNs are fit independently to the datasets $X^{1:M-1}$ and $Y^{1:M-1}$ using standard methods, in this case optimizing Eq. 6.1 to get point estimates $\tilde{\boldsymbol{\theta}}^{1:M-1}$. Second, a point estimate $\tilde{\boldsymbol{\Sigma}}$ is computed by maximizing $p(\boldsymbol{\Sigma}|\tilde{\boldsymbol{\theta}}^{1:M-1}; \beta)$, which is a simple regularized covariance estimator. Last, for a new task M with training data X^M and Y^M, a CNN with parameters $\boldsymbol{\theta}^M$ is trained with the SK-reg objective (Eq. 6.4), with $\boldsymbol{\Sigma} = \tilde{\boldsymbol{\Sigma}}$.

This procedure can be compared with the hierarchical Bayesian interpretation of MAML (Grant et al., 2018). Unlike MAML, our method allows flexibility to use different architectures for different datasets/episodes, and the optimizer for $\boldsymbol{\theta}^M$ is run to convergence rather than just a few steps.

6.5 Experiments

We evaluate our approach within a set of controlled visual learning environments. SK-reg parameters Σ_i for each convolution layer θ_i are determined by fitting a Gaussian to the kernels acquired from an earlier learning phase. We divide our learning tasks into two unique phases, applying the same

CNN architecture in each case. We note that our approach does not require a fixed CNN architecture across these two phases; the number of feature maps in each layer may be easily adjusted. A depiction of the two learning phases is given in Fig. 6.2.

Phase 1. The goal of phase 1 is to extract general principles about the structure of learned convolution kernels by training an array of CNNs and collecting statistics about the resulting kernels. In this phase, we train a CNN architecture to classify objects using a sufficiently large training set with numerous examples per object class. Training is repeated multiple times with unique random seeds, and the learned convolution kernels are stored for each run. During this phase, standard L2 regularization is applied to enforce a minimal constraint on each layer's weights (optimization problem of Eq. 6.1). After training, the convolution kernels from each run are consolidated, holding each layer separate. A multivariate Gaussian is fit to the centered kernel dataset of each layer, yielding a distribution $N(0, \Sigma_i)$ for each convolution layer i. To ensure the stability of the covariance estimators, we apply shrinkage to each covariance estimate, mixing the empirical covariance with an identity matrix of equal dimensionality. This can be interpreted as a hyperprior $p(\mathbf{\Sigma}; \beta)$ (Fig. 6.3) that favors small correlations. The optimal mixing parameter is determined via cross-validation.

Phase 2. In phase 2, we test the aptitude of SK-reg on a new visual recognition task, applying the covariance matrices Σ_i obtained from phase 1 to regularize each convolution layer i in a freshly-trained CNN (optimization problem of Eq. 6.4). In order to adequately test the generalization capability of our algorithm, we use a new set of classes that differ from the phase 1 classes in substantial ways, and we provide just a few training examples from each class. Performance of SK-reg is compared against standard L2 regularization.

6.5.1 Silhouettes

As a preliminary use case, we train our network using the binary shape image dataset developed at Brown University[1], henceforth denoted "Silhouettes." Silhouette images are binary masks that

[1]The binary shape dataset is available in the "Databases" section at http://vision.lems.brown.edu

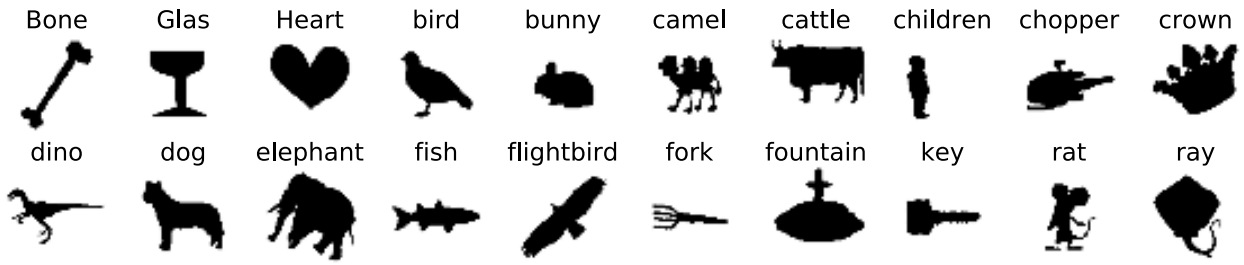


Figure 6.4: Exemplars of the phase 1 silhouette object classes.

Layer	Window	Stride	Features	λ
Input (200x200x3)				
Conv2D	5x5	2	5	0.05
MaxPooling2D	3x3	3		
Conv2D	5x5	1	10	0.05
MaxPooling2D	3x3	2		
Conv2D	5x5	1	8	0.05
MaxPooling2D	3x3	1		
FullyConnected			128	0.01
Softmax				

Table 6.1: CNN architecture. Layer hyperparameters include window size, stride, feature count, and regularization weight (λ). Dropout is applied after the last pooling layer and the fully-connected layer with rates 0.2 and 0.5, respectively.

depict the structural form of various object classes. Simple shape-based stimuli such as these provide a controlled learning environment for studying the inductive biases of CNNs (Feinman & Lake, 2018). We select a set of 20 well-structured silhouette classes for phase 1, and a set of 10 unique, well-structured classes for phase 2 that differ from phase 1 in their consistency and form. The images are padded to a fixed size of 200×200.

During phase 1, we train our network to perform 20-way object classification. Exemplars of the phase 1 classes are shown in Fig. 6.4. The number of examples varies for each class, ranging from 12 to 49 with a mean of 24. Class weighting is used to remedy class imbalances. To add complexity to the silhouette images, colors are assigned randomly to each silhouette before training. During training, random translations, rotations and horizontal flips are applied at each training epoch to improve generalization performance.

(a) First-layer kernels

(b) Gaussian samples

Figure 6.5: Learned first-layer kernels vs. Gaussian samples. (a) depicts some of the learned first-layer kernels acquired from phase 1 silhouette training. For comparison, (b) shows a few samples from a multivariate Gaussian that was fit to the first-layer kernel dataset.

We use a CNN architecture with 3 convolution layers, each followed by a max pooling layer (see Table 6.1). Hyperparameters including convolution window size, pool size, and filter counts were selected via randomized grid-search, using a validation set with examples from each class to score candidate values. A rectified linear unit (ReLU) nonlinearity is applied to the output of each convolution layer, as well as to the fully-connected layer. The network is trained 20 times using the Adam optimizer, each time with a unique random initialization. It achieves an average validation accuracy of 97.7% across the 20 trials, indicating substantial generalization.

Following the completion of phase 1 training, a kernel dataset is obtained for each convolution layer by consolidating the learned kernels for that layer from the 20 trials. Covariance matrices Σ_i for each layer i are obtained by fitting a multivariate Gaussian to the layer's kernel dataset. For a first-layer convolution with window size $K \times K$, this Gaussian has dimensionality $3K^2$, equal to the window area times RGB depth. We model the input channels as separate variables in layer 1 because these channels have a consistent interpretation as the RGB color channels of the input image. For remaining convolution layers, where the interpretation of input channels may vary from case to case, we treat each input channel as an independent sample from a Gaussian with dimensionality K^2. The kernel datasets for each layer are centered to ensure zero mean, typically requiring only a small perturbation vector.

To ensure that our multivariate Gaussians model the kernel data well, we computed the cross-

validated log-likelihoods of this estimator on each layer's kernel dataset and compared them to those of an i.i.d. Gaussian estimator fit to the same data. The multivariate Gaussian achieved an average score of 358.5, 413.3 and 828.1 for convolution layers 1, 2 and 3, respectively. In comparison, the i.i.d. Gaussian achieved an average score of 144.4, 289.6 and 621.9 for the same layers. These results confirm that our multivariate Gaussian provides an improved model of the kernel data. Some examples of the first-layer convolution kernels are shown in Fig. 6.5 alongside samples from our multivariate Gaussian that was fit to the first-layer kernel dataset. The samples appear structurally consistent with our phase 1 kernels.

In phase 2, we train our CNN on a new 10-way classification task, providing the network with just 3 examples per class for gradient descent training and 3 examples per class for validation. Colors are again added at random to each silhouette in the dataset. The network is initialized randomly, and we apply SK-reg to the convolution kernels of each layer during training using the covariance matrices obtained in phase 1. Our validation set is used to track and save the best model over the course of the training epochs (early stopping). A holdout set with 6 examples per class is used to assess the final performance of the model. A depiction of the train, validation and test sets used for phase 2 is given in Fig. 6.6. The validation and test images have been shifted, translated and flipped to make for a more challenging generalization test. Similar to phase 1, random shifts, rotations and horizontal flips are applied to the training images at each training epoch. As a baseline, we also train our CNN using standard L2 regularization.

The regularization weight λ is an important hyperparameter of both SK and L2 regularization. Before performing the phase 2 training assessment, we use a validated grid search to select the optimal λ for each regularization method, applying our train/validate sets.[2] The same weight λ is applied to each convolution layer, as done in phase 1.

Results. With our optimal λ values selected, we trained our CNN on the 10-way phase 2

[2]To yield interpretable λ values that can be compared between the SK and L2 cases, we normalize each covariance matrix to unit determinant by applying a scaling factor c, such that $\det(c\Sigma) = \det(I)$.

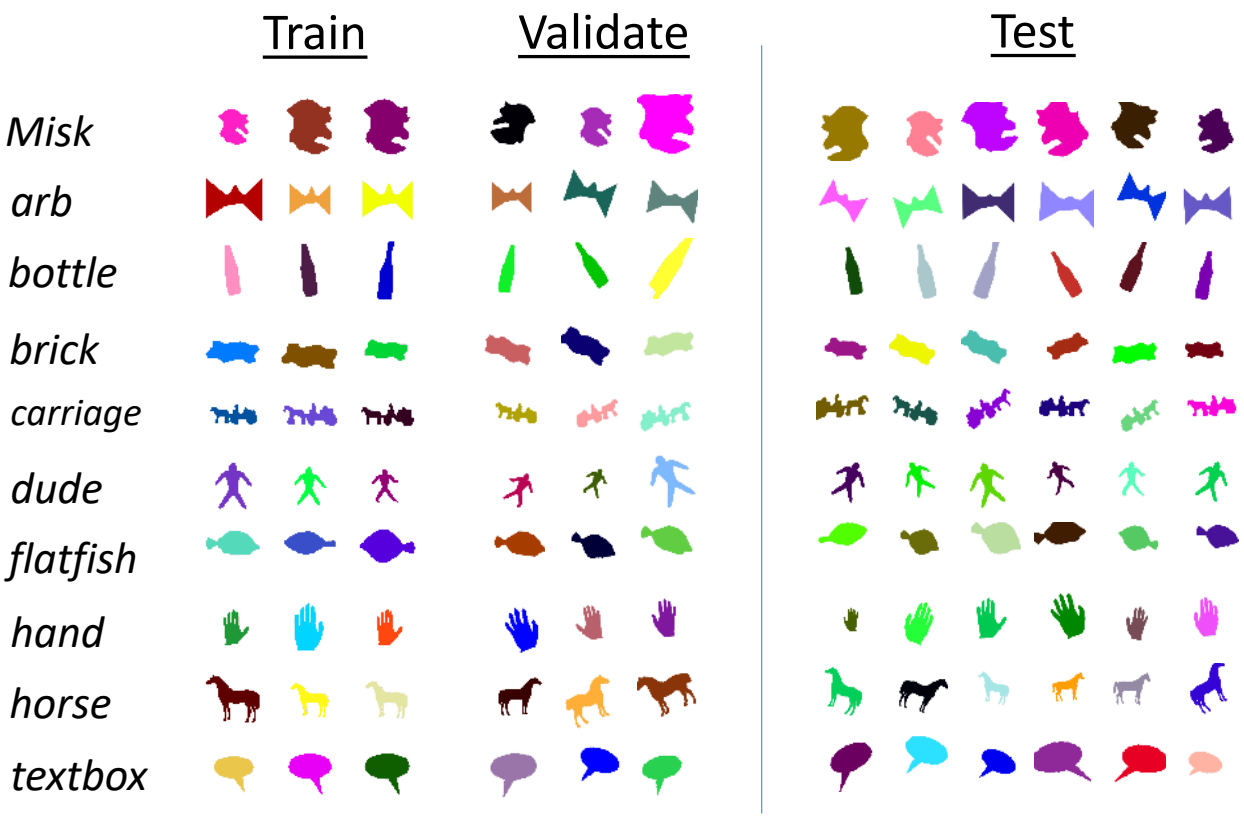


Figure 6.6: Silhouettes phase 2 datasets. 3 examples per class are provided in both the train and validation sets. A holdout test set with 6 examples per class is used to evaluate final model performance.

classification task of Fig. 6.6, comparing SK regularization to a baseline L2 regularization model. Average results for the two models collected over 10 training runs are presented in Table 6.2. Average test accuracy is improved by roughly 55% with the addition of SK reg, a substantial performance boost from 53.0% correct to 82.1% correct. Clearly, a priori structure is beneficial to generalization in this use case. An inspection of the learned kernels confirms that SK-reg encourages the structure we expect; these kernels look visually similar to samples from the Gaussian (e.g. Fig. 6.5).

Method	λ	Cross-entropy	Accuracy
L2	0.214	2.000 (+/- 0.033)	0.530 (+/- 0.013)
SK	0.129	0.597 (+/- 0.172)	0.821 (+/- 0.056)

Table 6.2: Silhouettes phase 2 results. For each regularization method, the optimal regularization weight λ was selected via grid-search. Results show the average cross-entropy and classification accuracy achieved on the holdout test set over 10 phase 2 training runs.

6.5.2 Tiny ImageNet

Our silhouette experiment demonstrates the effectiveness of SK-reg when the parameters of the regularizer are determined from the structure of CNNs trained on a similar image domain. However, it remains unclear whether these regularization parameters can generalize to novel image domains. Due to the nature of the silhouette images, the silhouette recognition task encourages representations with properties that are desirable for object recognition tasks in general. Categorizing silhouettes requires forming a rich representation of shape, and shape perception is critical to object recognition. Therefore, this family of representation may be useful in a variety of object recognition tasks.

To test whether our kernel priors obtained from silhouette training generalize to a novel domain, we applied SK-reg to a simplified version of the Tiny ImageNet visual recognition challenge, using covariance parameters fitted to silhouette-trained CNNs. Tiny ImageNet images were up-sampled with bilinear interpolation from their original size of 64×64 to mirror the Silhouette size 200×200. We selected 10 well-structured ImageNet classes that contain properties consistent with the silhouette images.[3] We performed 10-way image classification with these classes, using the same CNN architecture from Table 6.1 and applying the SK-reg soft constraint. The network is provided 10 images per class for training and 10 per class for validation. Because of the increased complexity of the Tiny ImageNet data, a larger number of examples per class is merited to achieve good generalization performance. A holdout test set with 20 images per class is used to evaluate performance. Fig. 6.7 shows a breakdown of the train, validate and test sets.

[3]Desirable classes have a uniform, centralized object with consistent shape properties and a distinct background.

Method	λ	Cross-entropy	Accuracy
L2	0.450	1.073 (+/- 0.102)	0.700 (+/- 0.030)
SK	0.450	0.956 (+/- 0.180)	0.776 (+/- 0.035)

Table 6.3: Tiny ImageNet SK-reg and L2 results. Table shows the average cross-entropy and classification accuracy achieved on the holdout test set over 10 training runs.

A few modifications were made to account for the new image data. First, we modified the phase 1 silhouette training used to acquire our covariance parameters, this time applying random colors to both the foreground and background of each silhouette. Previously, each silhouette overlaid a strictly white background. Consequently, the edge detectors of the learned CNNs would be unlikely to generalize to novel color gradients. Second, we added additional regularization to our covariance estimators to avoid over-fitting and help improve the generalization capability of the resulting kernel priors. Due to the nature of the phase 2 task in this experiment, and the extent to which the images differ from phase 1, additional regularization was necessary to ensure that our kernel priors could generalize. Specifically, we applied L1-regularized inverse covariance estimation (Friedman et al., 2008) to estimate each Σ_i, which can be interpreted as a hyperprior $p(\mathbf{\Sigma}; \beta)$ (Fig. 6.3) that favors a sparse inverse covariance (Lake & Tenenbaum, 2010).

Similar to the silhouettes experiment, the validation set is used to select weighting hyperparameter λ and to track the best model over the course of learning. As a baseline, we again compared SK-reg to a λ-optimized L2 regularizer.

Results. SK-reg improved the average holdout performance received from 10 training runs as compared to an L2 baseline, both in accuracy and cross-entropy. Results for each regularization method, as well as their optimal λ values, are reported in Table 6.3. An improvement of 8% in test accuracy suggests that some of the structure captured by our kernel prior is useful even in a very distinct image domain. The complexity of natural images like ImageNet is vast in comparison to simple binary shape masks; nonetheless, our prior from phase 1 silhouette training is able to influence ImageNet learning in a manner that is beneficial to generalization.

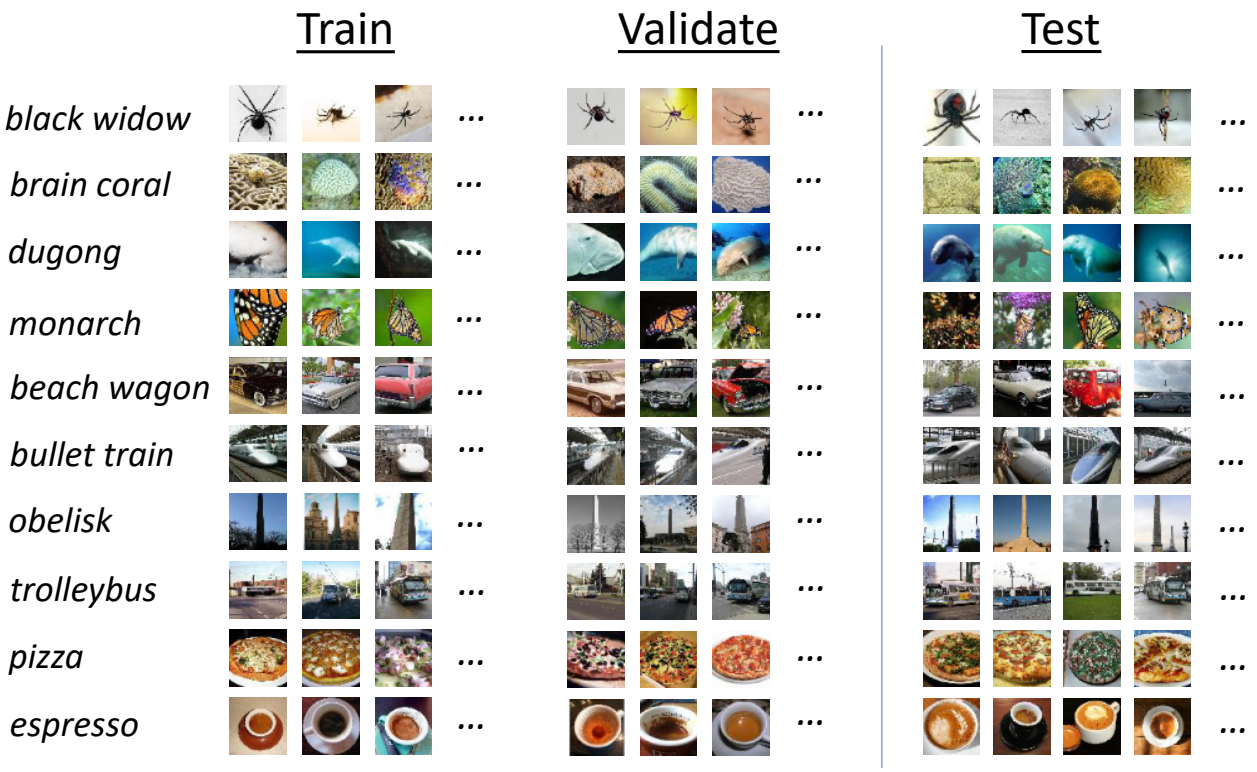


Figure 6.7: Tiny ImageNet datasets. 10 classes were selected to form a 10-way classification task. The train and validate sets each contain 10 examples per class. The holdout test set contains 20 examples per class.

6.6 Discussion

Using a set of controlled visual learning experiments, our work in this paper demonstrates the potential of structured receptive field priors in CNN learning tasks. Due to the properties of image data, smooth, structured receptive fields have many desirable properties for visual recognition models. In our experiments, we have shown that a simple multivariate Gaussian model can effectively capture some of the structure in the learned receptive fields of CNNs trained on simple object recognition tasks. Samples from the fitted Gaussians are visually consistent with learned receptive fields, and when applied as a model prior for new learning tasks, these Gaussians can help a CNN generalize in conditions of limited training data. We demonstrated our new regularization method in two simple use cases. Our silhouettes experiment shows that, when the parameters of

SK-reg are determined from CNNs trained on a similar image domain to that of the new task, the performance increase that results in the new task can be quite substantial–as large as 55% over an L2 baseline. Our Tiny ImageNet experiment demonstrates that SK-reg is capable of encoding generalizable structural principles about the correlations in receptive fields; the statistics of learned parameters in one domain can be useful in a completely new domain with substantial differences.

The Gaussians that we fit to kernel data in phase 1 of our experiments could be overfit to the CNN training runs. We have discussed the application of sparse inverse covariance (precision) estimation as one approach to reduce over-fitting. In future work, we would like to explore a Gaussian model with graphical connectivity that is specified by a 2D grid MRF. Model fitting would consist of optimizing the non-zero precision matrix values subject to this pre-specified sparsity. The grid MRF model is enticing for its potential to serve as a general "smoothness" prior for CNN receptive fields. Ultimately, we hope to develop a general-purpose kernel regularizer that does not depend on transfer learning.

Although a Gaussian can model some kernel families sufficiently, other families would give it a difficult time. The first-layer kernels of AlexNet–which are 11×11 and are visually similar to Gabor wavelets and derivative kernels–are not well-modeled by a multivariate Gaussian. A more sophisticated prior is needed to model kernels of this size effectively. In future work, we hope to investigate more complex families of priors that can capture the regularities of filters such as Gabors and derivatives. Nevertheless, a simple Gaussian estimator works well for smaller kernels, and in the literature, it has been shown that architectures with a hierarchy of smaller convolutions followed by nonlinearities can achieve equal (and often better) performance as those will fewer, larger kernels (Simonyan & Zisserman, 2015). Thus, the ready-made Gaussian regularizer we introduced here can be used in many applications.

Chapter 7

Conclusion and future directions

Human conceptual knowledge supports a variety of unique capabilities spanning perception, production and reasoning. The form of this knowledge is of great interest to cognitive scientists. One tradition has emphasized structured knowledge (Section 1.1), viewing concepts as embedded in intuitive theories or as organized by symbolic representations such as trees, grammars and programs. A second tradition has emphasized statistical knowledge (Section 1.2), viewing concepts as embodiments of patterns and correlations from observed data and as exemplified by training neural network models. Each of these traditions alone provides an incomplete account of human behavior, although their strengths are complementary.

This thesis presents a new synthesis of ideas from the structured and statistical traditions that helps understand and account for human concept learning. We have proposed a new framework for computational models of conceptual knowledge called Generative Neuro-Symbolic (GNS) Modeling and demonstrated how this framework can account for human behavior in a variety of concept learning tasks. GNS models represent concepts as probabilistic programs with neural network subroutines, leveraging the strengths of both structured symbolic representation and nonparametric statistical estimation. The control flow of a probabilistic program, coupled with symbolic primitives

and renderers, provides an explicit representation of the *causal* and *compositional* processes by which concepts are formed. This type of representation helps explain how concepts are applied flexibly to a variety of tasks, overcoming a shortcoming of purely-statistical approaches. In addition, by using neural networks to represent program subroutines, GNS is able to capture complex correlations and account for behaviors that are not well-explained by fully symbolic models.

Chapter 2 presents a first case study of GNS modeling designed to systematically evaluate the computational ingredients of the framework. In the study, we developed a generative model to account for the ways that people generate handwritten character concepts. Using likelihood of real human-drawn characters to measure the behavioral account of the model, we showed that our GNS model outperforms two alternative models with more generic neural network architectures. One of these alternatives, the Hierarchical LSTM (H-LSTM), retains an explicit notion of parts but uses a less constrained form of memory to propagate information and model statistical correlations between parts. The improvement of GNS over H-LSTM suggests that intermediate symbolic rendering and controlled canvas memory are valuable ingredients for models of human concepts. Unlike the fully-symbolic Bayesian Program Learning (BPL) model, our GNS model generates new characters that exhibit complex correlations and that appear stylistically consistent with human examples, evidencing the effectiveness of neural network submodules.

Chapter 3 builds on the work of Chapter 2 and develops a full hierarchical model of character concepts, as well as techniques to perform probabilistic inference with the model. Whereas Chapter 2 focused on the forward generative model in isolation, aiming to evaluate the GNS architecture in a controlled setting, Chapter 3 demonstrates how the generative model can be used to perform four unique concept learning tasks that people grapple with ease. By performing a variety of unique tasks with a single model, we provided an account for the *task-generality* of human concepts in ways that previous work has fallen short. In addition, our experiments in this chapter offer an account of how people generate new concepts and new exemplars that are novel yet structurally consistent with familiar ones. This human capability has been demonstrated in a number of different creative

domains (Ward, 1994; Jongejan et al., 2016), and yet it has presented a challenge for computational models (Lake et al., 2019). The GNS model produces new characters that are consistent with human generations, yet that are sufficiently dissimilar from their nearest training example. Lastly, our experiments in Chapter 3 demonstrate the importance of symbolic machinery to understanding and reproducing the ways that people quickly grasp new concepts and use them in a variety of tasks. Our GNS model relies on symbolic notions of pen actions (splines, stroke breaks, etc.) and a symbolic rendering engine to capture the latent causal process by which characters are formed. As demonstrated, this machinery provides clear improvements over alternative models that operate directly on image pixels.

Some of the evaluations used in Chapters 2 and 3 were qualitative; in particular the evaluations of new concepts and new examples generated by the model. These experiments would benefit from a series of visual Turing tests (Lake et al., 2015) that would further quantify the behavioral account of the model. Conducting these Turing tests is a primary interest for future work. In addition, comparisons with the fully-symbolic Bayesian Program Learning (BPL) model (Lake et al., 2015)—our most powerful benchmark and the only that performs a diversity of Omniglot tasks—are mostly qualitative and could benefit from further development. The BPL model does not provide a means to compute the marginal likelihood of a character, neither in its image or drawing form. This makes direct comparisons difficult, but further design could produce more thoughtful ways to compare the two models.

Alphabetical characters are only one of the many types of concepts that people encounter, and in Chapter 4, we showed how to extend the GNS framework to model a new class of human concepts with some distinguishing qualities. These concepts, dubbed “alien figures," have much richer intra-class variability, with tokens that can vary along a number of different attribute dimensions. We handled this change by shifting up one level in the generative hierarchy, using GNS to model different tokens of an individual concept. We showed that our GNS model can account for the ways that people quickly learn and generalize new alien figure concepts that embody different

types of composition ranging from fixed part structure to abstract relational rules. Compared to a fully-symbolic Bayesian model with strong domain knowledge, our GNS model provides a considerable improvement in likelihood of human few-shot learning behavior, and it provides an improved account of two salient inductive biases that human participants exhibit.

Although the current GNS model of alien figures is an important step, it relies on the Bayesian model and other synthetic generators for training data, and as result, it may include some of the same biases and shortcomings of these parametric distributions. In future work, we would like to scale up the human experiment to provide enough training data such that we can learn a GNS model more directly from human behavior. The current dataset includes only 155 trials in total, which is insufficient to train the neural network modules considered in this thesis. With a larger experiment and sufficient data to train on human data alone, we believe that GNS's behavioral account will strengthen even farther beyond the current benchmarks.

The architecture of generative neuro-symbolic (GNS) modeling imposes a powerful inductive bias over statistical neural network models by building in causal generative modeling and compositional representation. We see this framework as the most promising direction toward models that explain the dual structural and statistical characteristics of human concepts; however, it is not the only way to integrate inductive biases and neural networks. Chapters 5 and 6 explore other ways that inductive biases can be learned by and imposed upon neural network models, focusing on weaker inductive biases such as a preference to organize objects by shape and a smoothness prior. These experiments demonstrate how, with only minimal modifications, existing neural network toolkits can account for results from developmental psychology and can learn-to-learn new concepts in a more efficient manner. Although valuable as a complement, these experiments are secondary to the central focus and contribution of the thesis.

One general take-away from our work on GNS models is that causal generative models provide an effective means to understand and model the mind and brain. A number of human behavioral studies have suggested that people's generative knowledge influences the way they perceive and

interact with concepts. In a seminal work, Freyd (1983) showed that the way that people learn to draw character symbols influences subsequent recognition and categorization of the same symbols. Tse & Cavanagh (2000) further uncovered that prior production knowledge influences apparent motion when characters are presented stroke-by-stroke to human participants, and that the signal is strong enough to override perceptual grouping cues. Generative neuro-symbolic (GNS) models provide an effective means to understand these phenomenon in computational terms.

In addition to the behavioral evidence for generative models, neuroimaging studies have confirmed that a sensorimotor network is engaged upon the presentation of character stimuli (Longcamp et al., 2003b; James & Gauthier, 2006), offering further support for the role of generative production knowledge in the brain. A premotor area in the left dorsal precentral gyrus that is activated during handwriting, located roughly at the hand region of motor cortex, is also activated upon viewing characters. In addition, the character region of ventral visual stream is activated when participants write characters from memory with no visual stimulus. Researchers at Indiana University have found neuroimaging evidence that the functional specialization for characters may result from people's experience drawing characters, and that these circuits are representing a synthesis of visual information and stored production knowledge (James & Atwood, 2009; James, 2010). Together, these results suggest that generative models provide a promising means to understand and model the brain.

Another important take-away from GNS modeling is that human concepts are well-explained as including a synthesis of structural and statistical representation. Models from tradition 1 and tradition 2 emphasize only one of these two ingredients, and each has fallen short in accounting for components of human concept learning. Our experiments with GNS modeling improve over prior work from the two traditions and provide preliminary evidence that human concepts include both structural and statistical ingredients.

There are two important ways that our work on generative neuro-symbolic (GNS) models can be expanded to provide a more comprehensive account of human concepts. First, the generality of the

framework can be further supported with additional experiments that add breadth to our applications. Human concept learning is distinguished for having both a breadth and depth of applications (Murphy, 2002; Lake et al., 2019), and although our current results are a start, the experiments are limited to just two conceptual domains. We have designed the framework around general principles of concepts—namely, that concepts are composed of reusable parts and relations—and we see many new domains to which these ideas can extend. Second, the ability of GNS models to handle real-world complexity could be supported through further experiments. People learn new concepts directly from raw, high-dimensional sensory data, and they identify instances of known concepts embedded in similarly complex stimuli. Our current experiments are limited to simple concepts composed of only basic lines and shapes. In future work, we'd like to scale the framework up and develop GNS models of concepts with more real-world complexity, such as objects embedded in natural images or in 3D representations.

The remainder of this chapter explores ongoing and future directions that address the two interests of the preceding paragraph. The intention of these directions is to help demonstrate the broad range of concepts that GNS models can capture and provide an avenue toward modeling object concepts with real-world complexity. Some of these directions include preliminary experiments; however, this research is unpublished and not yet complete enough for submission. Section 7.1 presents preliminary experiments with structured block concepts that embody physical properties and abstract symbolic relations. Section 7.2 demonstrates that GNS can account for simplified renditions of real-world object concepts such as ice cream, house and building. Finally, Section 7.3 presents a proposal for a GNS model of 3D objects embedded in raw, high-dimensional stimuli. This final section is a future direction, as there are no experiments to discuss at the current time.

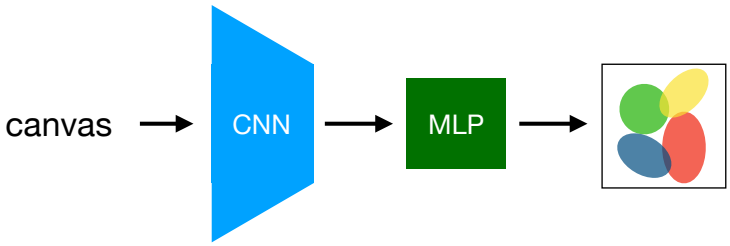


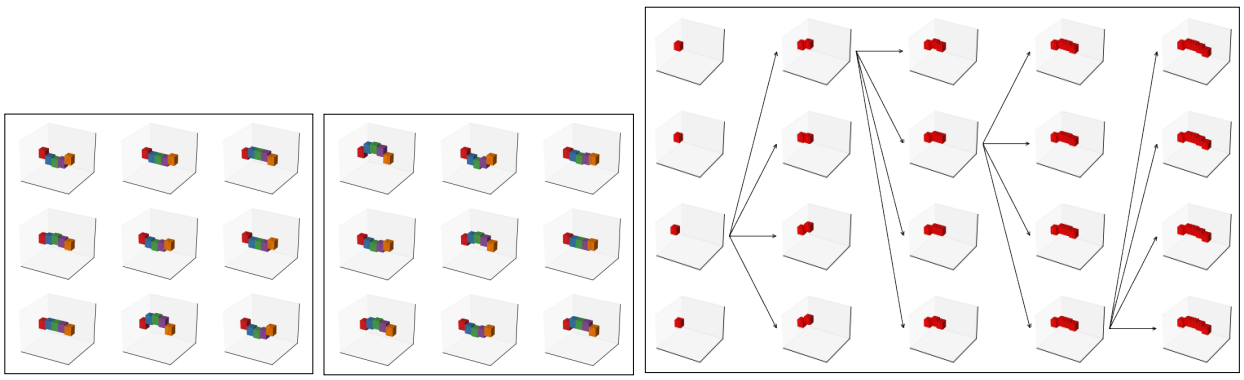
Figure 7.1: Neural network architecture of GNS subroutine `GeneratePart` for structured blocks concepts. The network first reads the current canvas (partial object) as a 3D voxel image and processes it with a 3D convolutional neural network (CNN) to form a hidden representation. This hidden layer is then fed to a multi-layer perceptron (MLP), followed by a mixture density output head (Graves, 2013) that predicts a distribution for the next part location $l_i \in \mathcal{R}^3$.

7.1 GNS model of structured blocks concepts

As another proof-of-concept of generative neuro-symbolic (GNS) modeling, we evaluated whether GNS models can represent two types of synthetic concepts that require reasoning about physical properties. In this domain, stimuli are 3D voxel images composed by stacking basic cube blocks with uniform shape and size to form various compound shapes. We endow the model with a simple part generator that generates the i^{th} part by specifying a location $l_i \in \mathcal{R}^3$. The part generator, depicted in Fig. 7.1, uses a mixture density network (Graves, 2013) similar to those from Chapters 2 & 3. The network outputs a distribution for l_i, and it is trained to maximize the log-likelihood of true part locations provided with the training data. We tested whether a GNS model could learn to represent highly structured physical concepts and relations using only this highly generalized, real-valued part representation. Importantly, stimuli in this experiment are represented as 3D voxel images, compared to the standard 2D color images used in previous chapters.

7.1.1 Parabolas

We first evaluated the GNS model on simple parabola stimuli formed from 5 blocks each (Fig. 7.2(a)). This first experiment tests whether GNS can learn to represent a structured physical concept



(a) Data (b) Model samples (c) Generating part-by-part

Figure 7.2: Learning to generate parabola concepts. (a) A collection of real examples from the parabola dataset. (b) A collection of examples from the GNS model trained to generate parabolas. (c) The GNS model generates new examples of parabolas part-by-part, visualized with a sequential sample tree.

that manifests as relationships between multiple parts of a compound object. In each stimuli, the relative location of block parts is dictated by a U-shaped parabola. Across the 5 parts, one location dimension, z, remains constant, while the other two dimensions relate to each other by the formula $y = a \cdot x^2$. The scalar a is sampled independently for each stimuli from a mean-centered normal distribution, $a \sim \mathcal{N}(0, \sigma^2)$. To successfully model these concepts, GNS must reason about the abstract mathematical formula that dictates the relationship between spatial locations of different block parts.

After training, the GNS model successfully generates new examples of parabolas, producing stimuli that are indistinguishable from real data to the eye (Fig. 7.2(b)). These results suggest that the model has learned to represent the symbolic relationship that governs different parts of the stimuli. Another view into the resulting model is provided by visualizing the part-by-part sample generation process (Fig. 7.2(c)).

In addition to producing high-fidelity samples of whole objects, the uncertainty of the GNS model at different phases of sample generation—conveyed by the variability of next-part samples for different partial canvases (Fig. 7.3)—matches with our intuition about the parabola domain. At

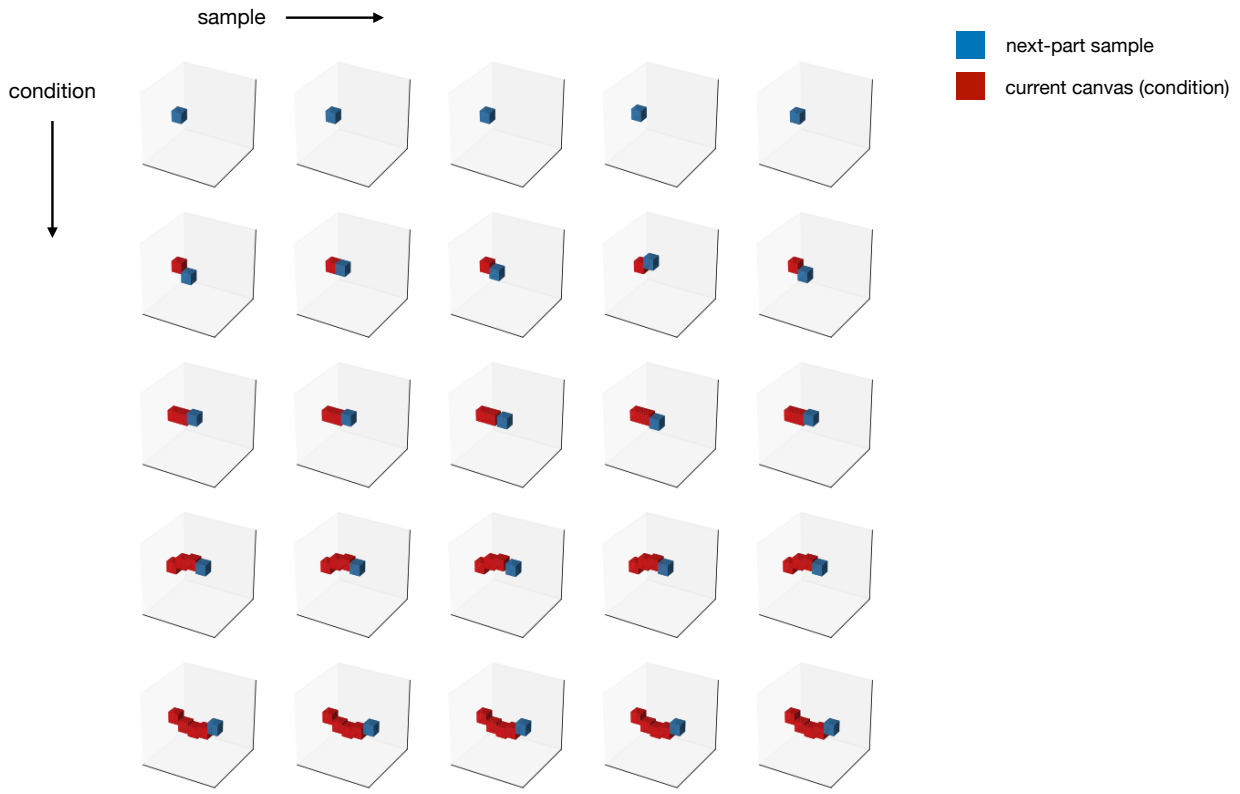
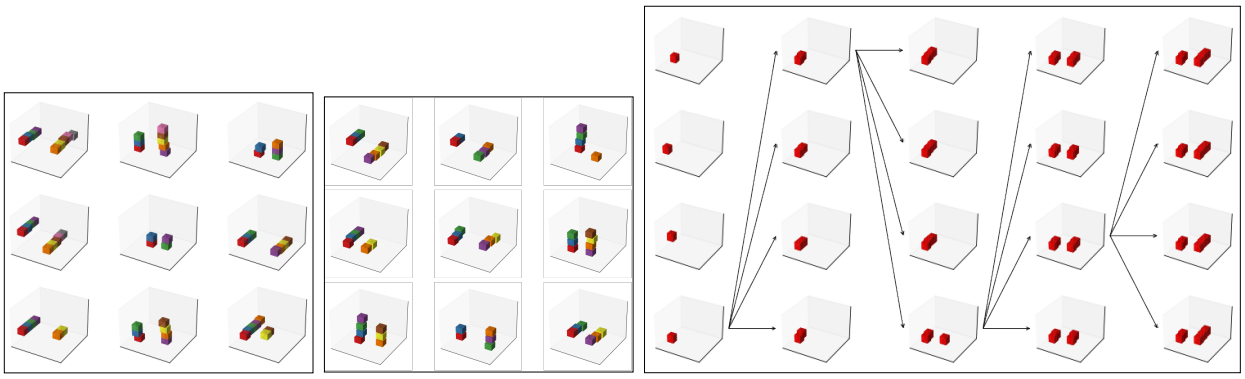


Figure 7.3: Next-part samples from the GNS model of parabola concepts. For each of 5 different partial-object canvases (rows), the model produces 5 unique samples of the next part (columns).

early phases of the sample, such as rows 1-3 of Fig. 7.3 where the canvas has only 0-2 previous parts, the model shows a high level of uncertainty about the next, conveyed by notable variance between its next-part samples. As we get further into the sample and the parabola becomes more clear, the model begins to collapse on a more confident and uniform prediction about the next part.

7.1.2 Parallel towers

This experiment tests whether GNS can learn a unique class of concepts with rigid and discrete structure that we denote *parallel towers* (Fig. 7.4(a)). Each stimuli consists of two block towers, and every block is connected to one of the two discrete towers. Moreover, towers are organized either vertically or horizontally, and the choice of orientation must match between the two towers. Understanding the parallel towers concepts requires reasoning about the concept *two*: every part



(a) Data (b) Model samples (c) Generating part-by-part

Figure 7.4: Learning to generate parallel towers concepts. (a) A collection of real examples from the parallel towers dataset. (b) A collection of examples from the GNS model trained to generate parallel towers. (c) The GNS model generates new examples of parallel towers part-by-part, visualized with a sequential sample tree.

must be connected to one of two discrete towers, and there must always be two (i.e., there must always be at least one block in each of two disconnected locations). In addition, it requires reasoning about discrete orientations: each stimuli has either a vertical or horizontal orientation, and both of the two block towers must match this orientation.

The architecture of `GeneratePart` is the same as for parabolas (Fig. 7.1) with one modification: for this domain, since concepts can have a variable number of parts, we add a termination predictor as an additional output head to the neural network. By predicting yes-or-no whether to continue with another part at each step, the model implicitly specifies a number of parts for the object.

Fig. 7.4(b) shows some examples of new parallel towers stimuli generated by the GNS model after training. The model successfully generates from the concept, producing stimuli that always contain two disconnected towers and that have either vertical or horizontal orientation. Moreover, the statistics of part count, and of horizontal-vs-vertical orientation, match closely with the parameters of the real data. Fig. 7.5 visualizes the GNS model's part-by-part generation of parallel towers stimuli with a sample tree diagram.

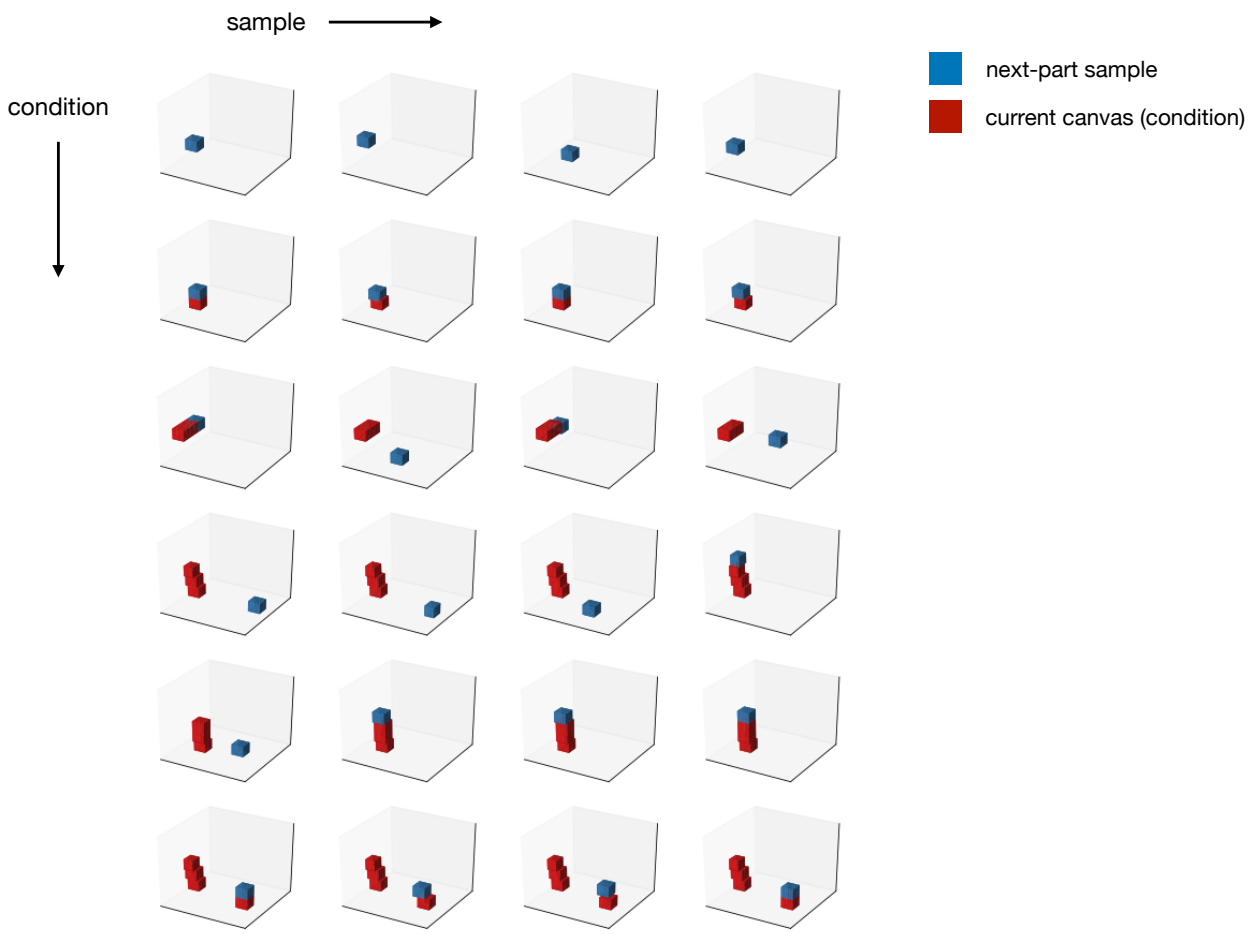


Figure 7.5: Next-part samples from the GNS model of parallel towers concepts. For each of 6 different partial-object canvases (rows), the model produces 4 unique samples of the next part (columns).

Much like with parabola concepts, the variability of GNS next-part samples for parallel towers concepts (Fig. 7.5) matches with our intuition about the uncertainty at different points of the generation process. When the first tower contains only a single part (row 2), the model is confident that the next part will go to the same tower, because it is unlikely to see a one-part tower. However, once the first tower has two parts (row 3), the model is 50/50 about whether the next part will start a new tower vs. add to the existing one.

7.2 GNS model of 2D objects

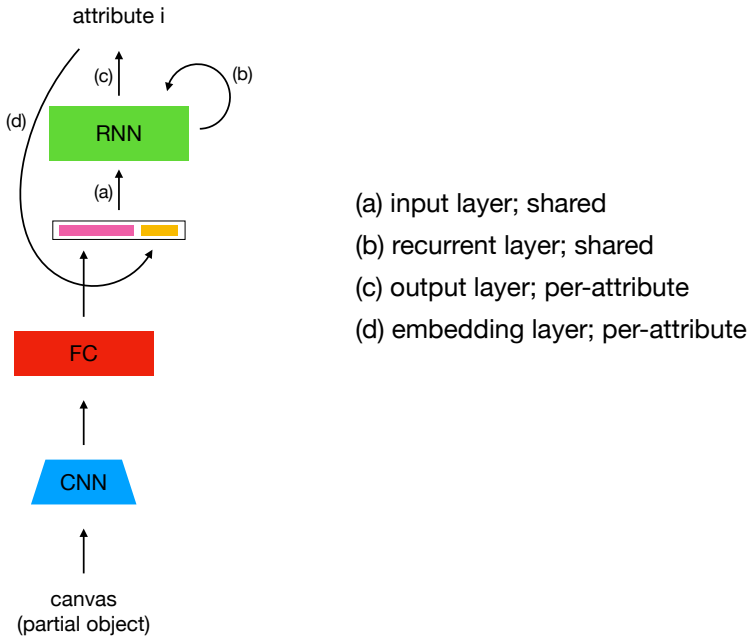


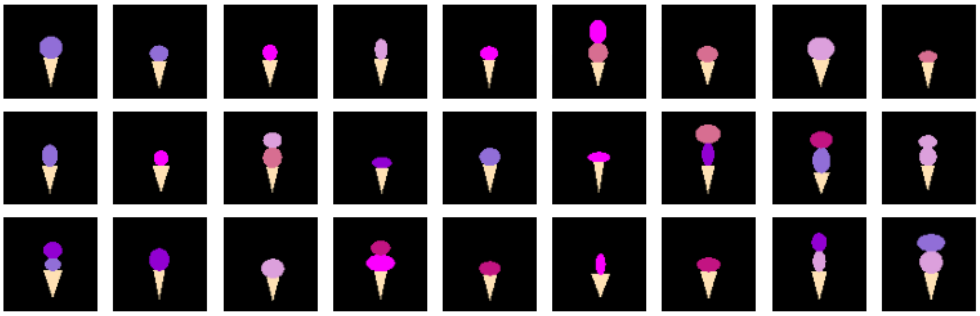
Figure 7.6: Neural network architecture of the GNS subroutine `GeneratePart` for 2D object concepts.

In addition to the structured block concepts discussed above, we also evaluated the GNS modeling framework for its ability to represent real-world objects. Our ultimate goal for GNS is to model object stimuli embedded in raw, high-dimensional data such as they occur in the natural world. Section 7.3 presents a proposal for how to achieve this vision using datasets with real-world complexity. In this section, we take a first step forward using simplified renditions of real-world object concepts including ice cream, houses, and buildings. Stimuli are 2D approximations of objects' real 3D form, constructed from a dictionary of basic polygon shapes that can vary in size and aspect ratio. An ice cream cone, for example, is composed of a single triangle and 1-2 ellipses, each with dimensions that can vary (Fig. 7.7(a)).

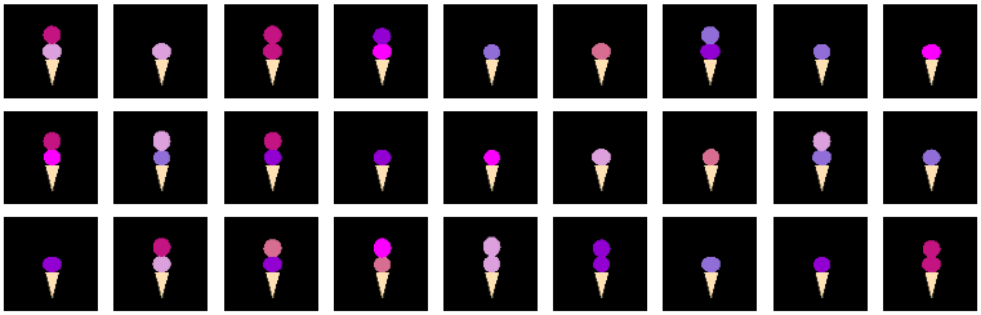
To generate instances of these 2D object concepts, we endow the model with a neural network part generator that samples a few basic attributes to specify the next part: 1) the primitive shape of

the part, 2) the color of the part, 3) its 2D location, and 4) its 2D size. These attributes are sampled in an autoregressive sequence using a recurrent neural network (Fig. 7.6). For part i, the generator first samples a categorical primitive ID, p_i, from a dictionary of 4 basic primitive polygons. It then samples a categorical color ID, c_i, that assigns a color from a dictionary of 14 basic colors. Finally, it samples a 2D location $l_i \in \mathcal{R}^2$ and a 2D size $s_i \in \mathcal{R}^2$ that specify the center and size of the part's outer bounding box in both x and y dimensions. The network outputs a distribution for each of these attributes, and it is trained to maximize the log-likelihood of true attribute labels provided with the training data for each category.

7.2.1 Ice cream



(a) Data



(b) Model samples

Figure 7.7: Ice cream concept. (a) A collection of real examples from the ice cream dataset. (b) An equal-size collection of examples from the GNS model trained to generate ice creams.

We first evaluated the GNS model for its ability to represent a 2D *ice cream* concept (Fig. 7.7(a)).

Stimuli are composed of 2-3 parts: a single cone part that manifests as a triangle, and 1-2 scoop parts that each manifest as ellipses. To successfully model these concepts, GNS must learn about the occurrence frequencies and relative placements of each part category. Placing each scoop so that it sits just atop the previous cone or scoop requires jointly reasoning about both location and size for the part.

Some examples of new ice cream stimuli generated by the GNS model after training are shown in Fig. 7.7(b). The trained GNS model successfully generates new examples of ice cream cones, producing stimuli that are mostly indistinguishable from real data to the eye (Fig. 7.7(b)). Moreover, the statistics of cones and scoops are a strong match with the real data provided in training. In addition, the uncertainty of the GNS model at different steps of generation—conveyed by the variability of next-part samples for different partial canvases (Fig. 7.8)—matches with our intuition about this domain.

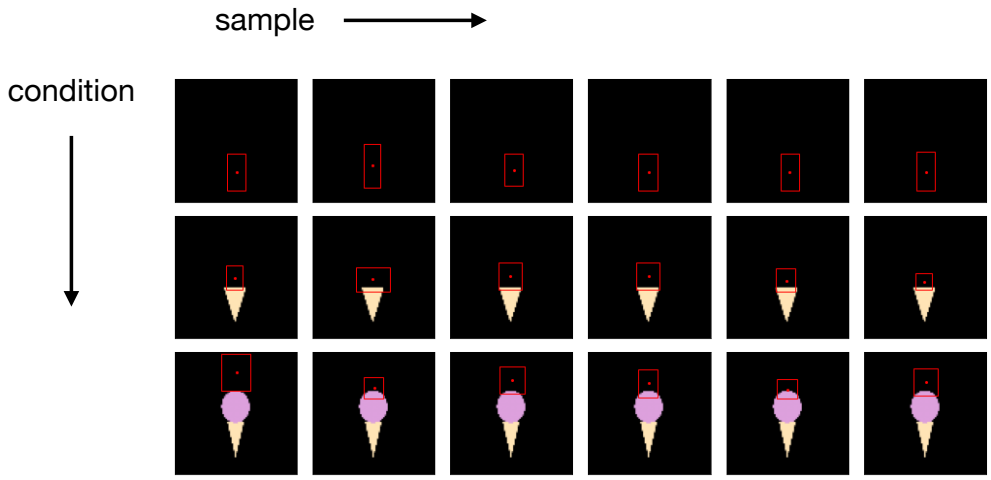


Figure 7.8: Next-part samples from the GNS model of ice cream concepts. For each of 3 different partial-object canvases (rows), the model produces 6 unique samples of the next part (columns).

7.2.2 House

Our second 2D object evaluation uses of a simple *house* concept (Fig. 7.9(a)). Each object is a house with exactly 3 parts: a core (rectangle), a roof (triangle) and a chimney (rectangle). This

(a) Data

(b) Model samples

Figure 7.9: House concept. (a) A collection of real examples from the house dataset. (b) An equal-size collection of examples from the GNS model trained to generate houses.

concept is the first of two building-related stimuli categories, and for starters we begin with very basic statistics that have little variability. The location and size of the core and the roof parts vary only slightly via a low-variance normal distribution. The location of the chimney has the largest variance, as it can appear at any x position along the roof.

Fig. 7.9(b) shows some examples of new house stimuli generated by the GNS model after the completion of training. In all examples, the model correctly generates one each of the three part categories and places them in their correct relative location such that all parts connect. Moreover, the model correctly matches the low variance of the core and roof parts seen in real data: the sizes and locations of these parts is relatively constant with only minor variability. Importantly, the model correctly matches the statistics of the “chimney" part, placing this part on either the left or right side of the roof and adding small variations on either side.

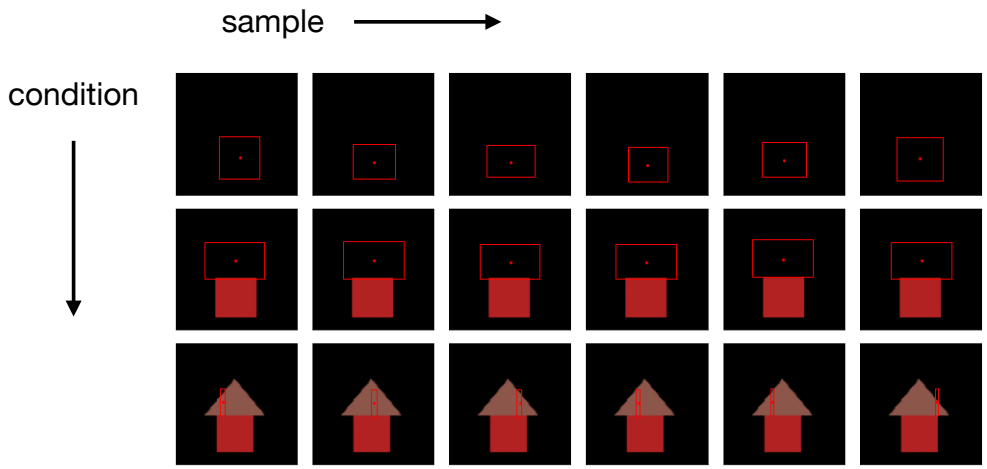
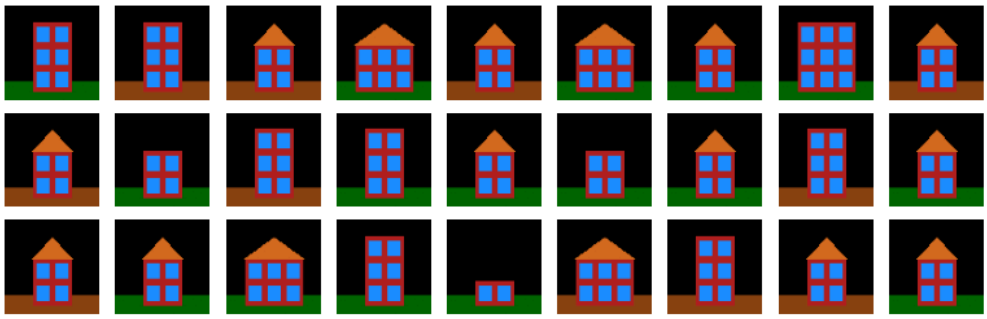


Figure 7.10: Next-part samples from the GNS model of house concepts. For each of 3 different partial-object canvases (rows), the model produces 6 unique samples of the next part (columns).

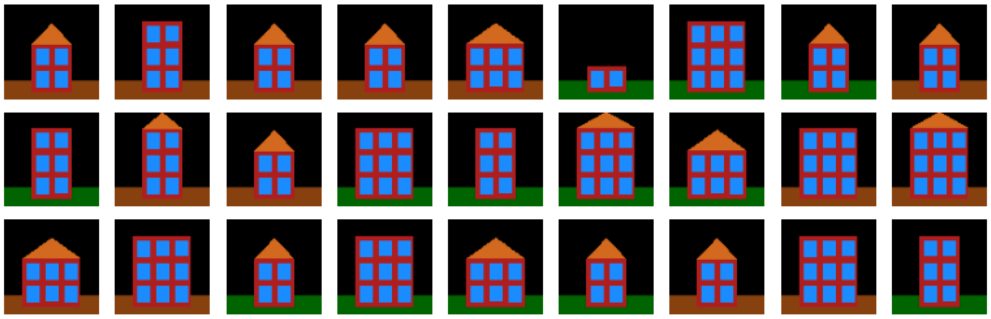
7.2.3 Building

Our third 2D object concept expands on the previous house with a more general *building* concept with richer variation (Fig. 7.11(a)). Like house, each building example has a core part, and some buildings also have a roof. Buildings, however, have a few additional variations that make for a richer class of stimuli. Each has an additional landscape part, which is a rectangle that takes one of two colors. Importantly, the core part of the *building* concept varies much more in size compared to *house*. There is an additional window part, and the number of windows present correlates with the size of the building core.

Fig. 7.11(b) shows some examples of new building stimuli generated by the GNS model after training. The model produces new examples that are near-indistinguishable from real data and that would likely pass a visual Turing test (Lake et al., 2015). Like the real data, each generated stimuli contains all of the required part categories: landscape, core, window, and (sometimes) roof. The landscape part is either blue or green with approximately 50% probability as per ground truth, as is the occurrence of a “roof" part. Importantly, the number of window rows and columns, which are each sampled between 1-3 in the real data, contains analogous statistics in the generated data.



(a) Data



(b) Model samples

Figure 7.11: Building concept. (a) A collection of real examples from the building dataset. (b) An equal-size collection of examples from the GNS model trained to generate buildings.

Moreover, the x and y sizes of the "core" part is correctly correlated with the number of window rows and columns in each generation.

7.3 Proposal: GNS model of 3D objects

The GNS modeling framework is designed to capture inductive biases for concept learning that generalize across different kinds of visual concepts. This same framework can potentially be used to model 3D object concepts, such chairs, vehicles and other categories from the ShapeNet library (Chang et al., 2015), again with a neuro-symbolic generative process for parts and relations. Here, we briefly review the path forward for training GNS models of 3D object concepts.

Everyday objects have more intra-class variability compared to the handwritten characters discussed in Chapters 2 & 3; for example, different chair tokens vary in the number of arm rests,

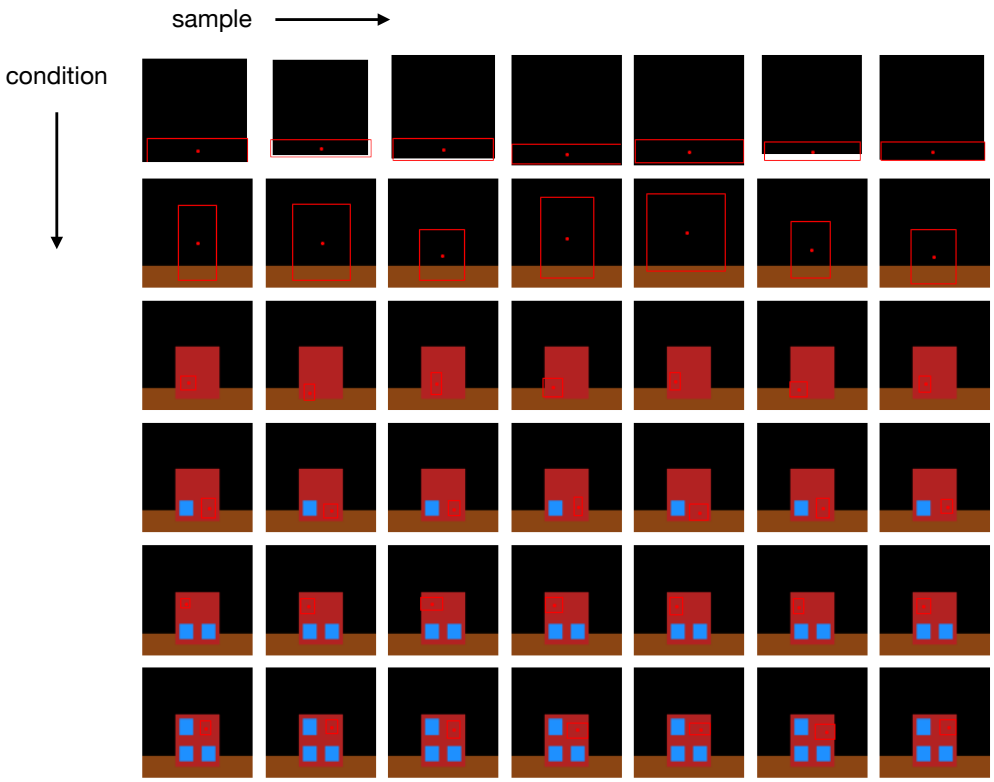


Figure 7.12: Next-part samples from the GNS model of building concepts. For each of 6 different partial-object canvases (rows), the model produces 7 unique samples of the next part (columns).

legs, and etc. much as different character types vary in their parts, although both domains are characterized by similar structural and statistical considerations. A similar case was evident with the alien figure concepts studied in Chapter 4. As done for alien figures, we again shift up one level in the generative hierarchy and design a token-level model for generating new exemplars of an individual concept (chairs, cars, etc.) that mirrors our type-level model for characters. The architecture and sampling procedure of a proposed GNS model for 3D object tokens is given in Fig. 7.13. As with characters, our object model produces samples one part at a time, using a 3D canvas C in place of the previous 2D canvas. The procedure `GeneratePart` consists of two neural network components: 1) a discriminitively-trained *category model* $p(k \mid C)$ that predicts the category label k of the next part given the current canvas (leg, arm, back, etc.), and 2) a generative *instance model* $p(x \mid k, C)$ that is trained with a variational autoencoder objective to sample an instance of the

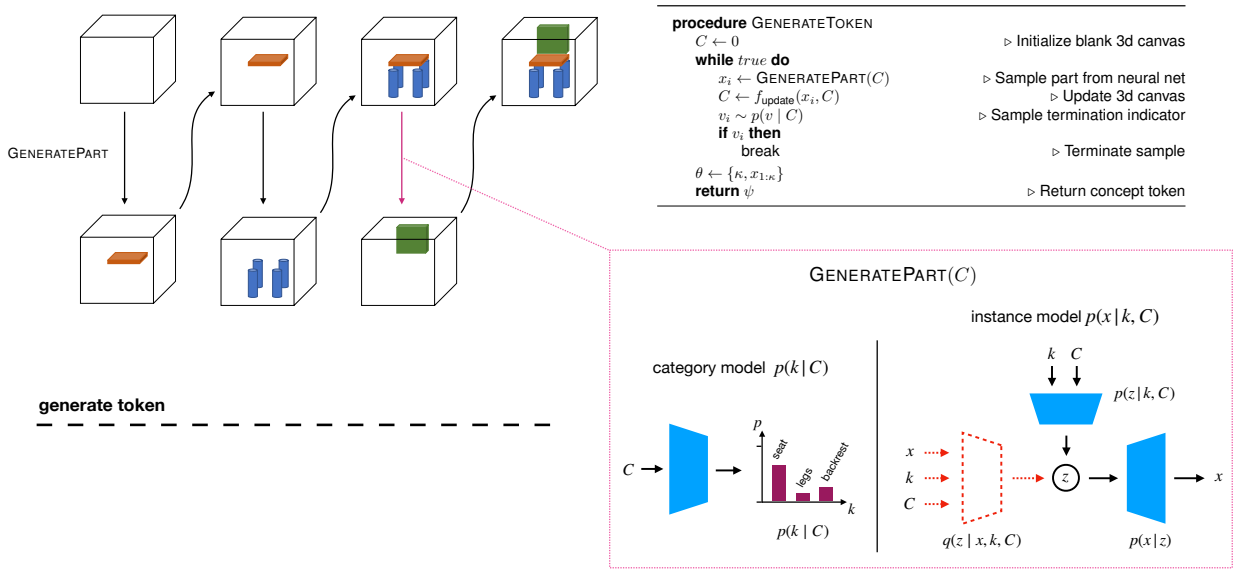


Figure 7.13: A proposed GNS model for the 3D object concept “chair."

next part x given the current canvas and predicted part category label. Objects and object parts are represented as 3D voxel grids, and all neural modules, including the category model and the encoder/decoder of the instance model, are parameterized by 3D convolutional neural networks. A function f_{update} is used to update the current canvas with the most recent part by summing the voxel grids. A GNS model for a particular concept is trained on examples of the 3D voxel grids with semantic part labels.

Appendix A

Additional details for Chapter 3

A.1 Hierarchical generative model

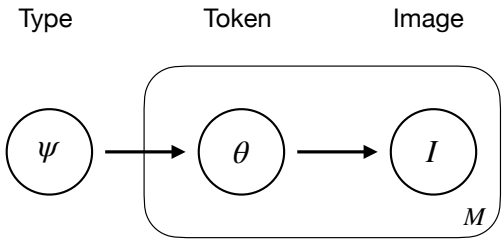


Figure A.1: The GNS hierarchical generative model.

The full hierarchical generative model of GNS is depicted in Fig. A.1. The joint density for type ψ, token $\theta^{(m)}$, and image $I^{(m)}$ factors as

$$P(\psi, \theta^{(m)}, I^{(m)}) = P(\psi)P(\theta^{(m)}|\psi)P(I^{(m)}|\theta^{(m)}). \tag{A.1}$$

The type ψ parameterizes a motor program for generating character tokens $\theta^{(m)}$, unique exemplars of the concept. Both ψ and $\theta^{(m)}$ are expressed as causal drawing parameters. An image $I^{(m)}$ is obtained from token $\theta^{(m)}$ by rendering the drawing parameters and sampling binary pixel values.

A.2 Training on causal drawing data

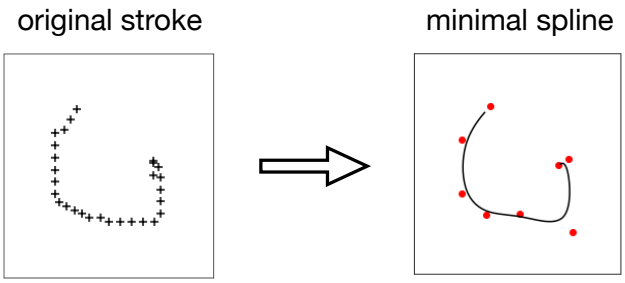


Figure A.2: Spline representation. Raw strokes (left) are converted into minimal splines (right) using least-squares optimization. Crosses (left) indicate pen locations and red dots (right) indicate spline control points.

To learn the parameters of $P(\psi)$ and $P(\theta^{(m)} \mid \psi)$, we fit our models to the human drawing data from the Omniglot background set. In this drawing data, a character is represented as a variable-length sequence of strokes, and each stroke is a variable-length sequence of pen locations $\{z_1, ..., z_T\}$, with $z_t \in \mathbb{R}^2$ (Fig. A.2, left). Before training our model on background drawings, we convert each stroke into a minimal spline representation using least-squares optimization (Fig. A.2, right), borrowing the B-spline tools from Lake et al. (2015). The number of spline control points depends on the stroke complexity and is determined by a residual threshold. Furthermore, we removed small strokes using a threshold on the trajectory length. These processing steps help suppress noise and emphasize signal in the drawings. Our generative models are trained to produce character drawings, where each drawing is represented as an ordered set of splines (strokes). The number of strokes, and the number of spline coordinates per stroke, are allowed to vary in the model.

A.3 Type prior

The type prior $P(\psi)$ represents a character as a sequence of strokes, with each stroke decomposed into a starting location $y_i \in \mathbb{R}^2$, conveying the first spline control point, and a stroke trajectory

$x_i = \{\Delta_1, ..., \Delta_N\}$, conveying deltas between spline control points. It generates character types one stroke at a time, using a symbolic rendering procedure called f_{render} as an intermediate processing step after forming each stroke. An image canvas C is used as a memory state to convey information about previous strokes. At each step i, the next stroke's starting location and trajectory are sampled with procedure `GeneratePart`. In this procedure, the current image canvas C is first read by the *location model* (Fig. 3.3), a convolutional neural network (CNN) that processes the image and returns a probability distribution for starting location y_i:

$$y_i \sim p(y_i \mid C).$$

The starting location y_i is then passed along with the image canvas C to the *stroke model*, a Long Short-Term Memory (LSTM) architecture with a CNN-based image attention mechanism.The stroke model samples the next stroke trajectory x_i sequentially one offset at a time, selectively attending to different parts of the image canvas at each sample step and combining this information with the context of y_i:

$$x_i \sim p(x_i \mid y_i, C).$$

After `GeneratePart` returns, the stroke parameters y_i, x_i are rendered to produce an updated canvas $C = f_{\text{render}}(y_i, x_i, C)$. The new canvas is then fed to the *termination model*, a CNN architecture that samples a binary termination indicator v_i:

$$v_i \sim p(v_i \mid C).$$

Both our location model and stroke model follow a technique from Graves (2013), who proposed to use neural networks with mixture outputs to model handwriting data. Parameters $\{\pi^{1:K}, \mu^{1:K}, \sigma^{1:K}, \rho^{1:K}\}$ output by our network specify a Gaussian mixture model (GMM) with

K components (Fig. 3.3; colored ellipsoids), where $\pi^k \in (0, 1)$ is the mixture weight of the k^{th} component, $\mu^k \in \mathbb{R}^2$ its means, $\sigma^k \in \mathbb{R}^2_+$ its standard deviations, and $\rho^k \in (-1, 1)$ its correlation. In our location model, a single GMM describes the distribution $p(y_i \mid C)$. In our stroke model, the LSTM outputs one GMM at each timestep, describing $p(\Delta_t | \Delta_{1:t-1}, y_i, C)$. The termination model CNN has no mixture outputs; it predicts a single Bernoulli probability to sample binary variable v_i. When sampling from the model at test time, we use a temperature parameter proposed by Ha & Eck (2018) (see (Ha & Eck, 2018, Eq. 8)) to control the entropy of the mixture density outputs.

A.4 Token model

procedure GENERATETOKEN(ψ)
- $\{\kappa, y_{1:\kappa}, x_{1:\kappa}\} \leftarrow \psi$ ▷ Unpack type-level variables
- **for** i = 1 ... κ **do**
  - $y_i^{(m)} \sim P(y_i^{(m)} \mid y_i)$ ▷ Sample token-level location
  - $x_i^{(m)} \sim P(x_i^{(m)} \mid x_i)$ ▷ Sample token-level part
- $A^{(m)} \sim P(A^{(m)})$ ▷ Sample affine warp transformation
- $\theta \leftarrow \{y_{1:\kappa}^{(m)}, x_{1:\kappa}^{(m)}, A^{(m)}\}$
- **return** θ ▷ Return concept token

Figure A.3: Token model sampling procedure.

Character types ψ are used to parameterize the procedure `GenerateToken`(ψ), a probabilistic program representation of token model $P(\theta^{(m)} \mid \psi)$. The psuedo-code of this sampling procedure is provided in Fig. A.3. The location model $P(y_i^{(m)} \mid y_i)$ and part model $P(x_i^{(m)} \mid x_i)$ are each zero-mean Gaussians, with standard deviations fit to the background drawings following the procedure of Lake et al. (2015) (see SM 2.3.3). The location model adds noise to the start of each stroke, and the part model adds isotropic noise to the 2d cooridnates of each spline control point in a stroke. In the affine warp $A^{(m)} \in \mathbb{R}^4$, the first two dimensions control global re-scaling of spline coordinates,

and the second two control a global translation of the center of mass. The distribution is

$$P(A^{(m)}) = \mathcal{N}([1, 1, 0, 0], \Sigma_A), \tag{A.2}$$

with the parameter Σ_A similarly fit from background drawings (see SM 2.3.4 in (Lake et al., 2015)).

A.5 Supplemental figures

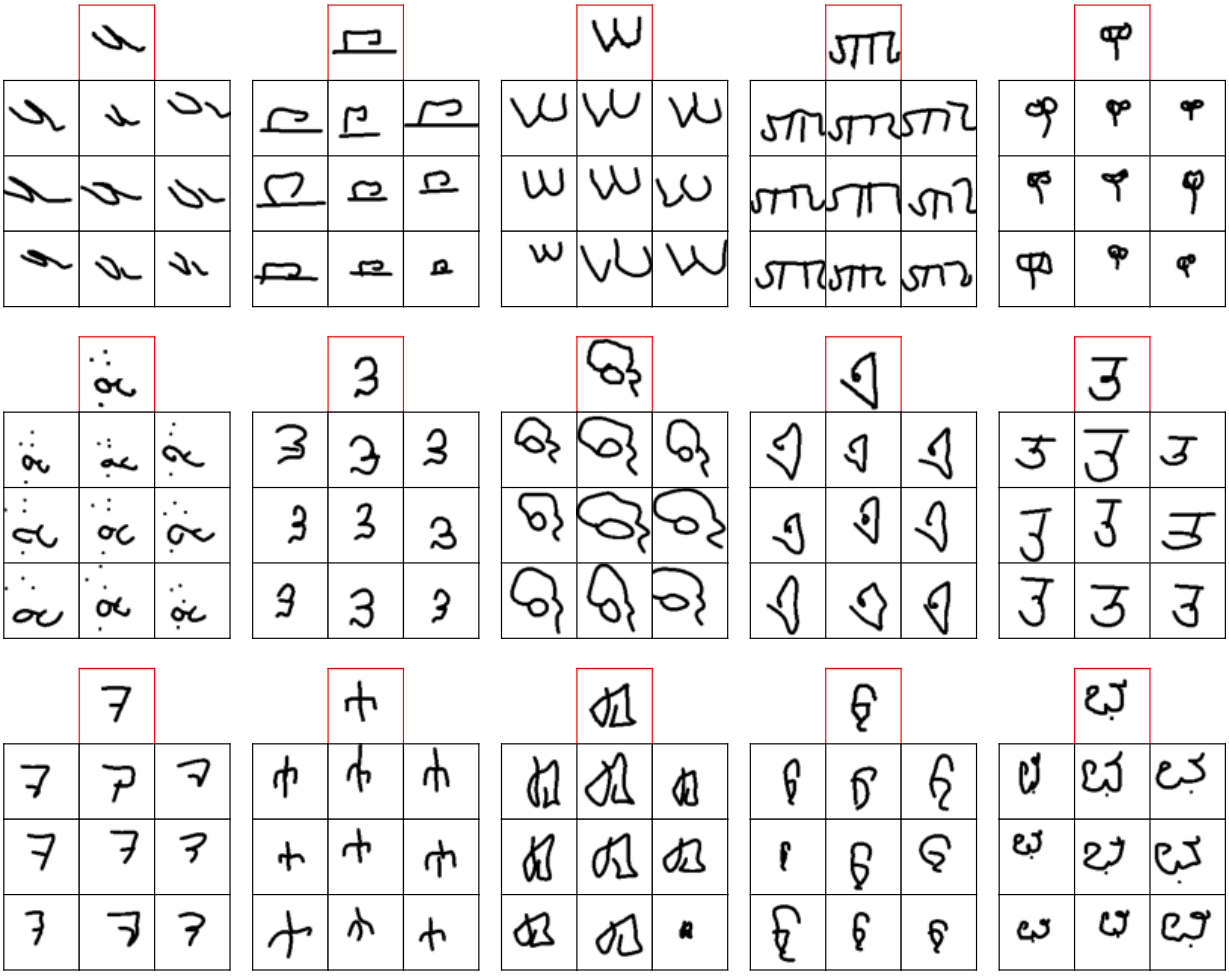


Figure A.4: Generating new exemplars with GNS. Twelve target images are highlighted in red boxes. For each target image, the GNS model sampled 9 new exemplars, shown in a 3x3 grid under the target.

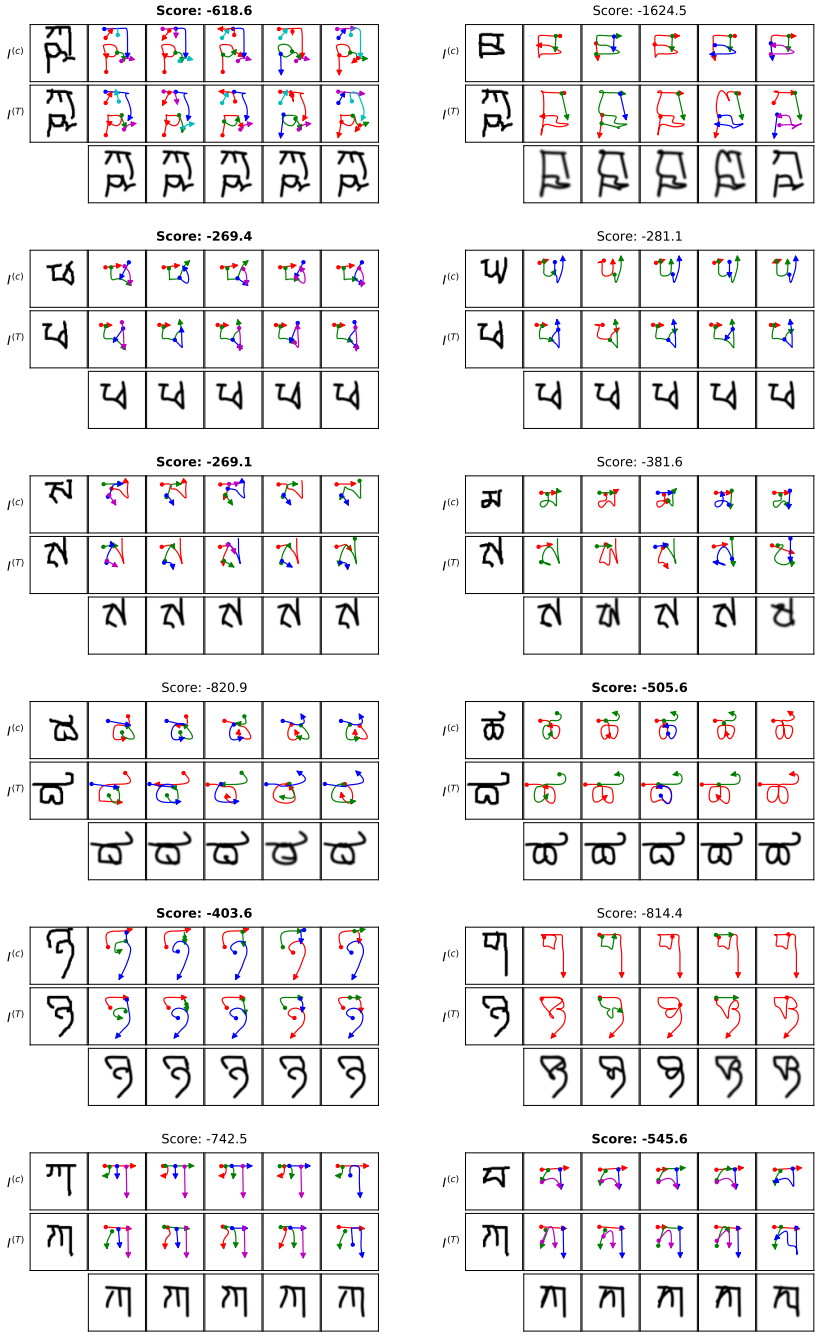


Figure A.5: Classification fits. Each row corresponds to one classification trial (one test image). The first column shows parses from the correct training image re-fit to the test example, and the second column parses from an incorrect training image. The two-way score for each train-test pair is shown above the grid, and the model's selected match is emboldened. The 4th and 6th row here are misclassified trials.

Appendix B

Additional details for Chapter 4

B.1 Relation architecture

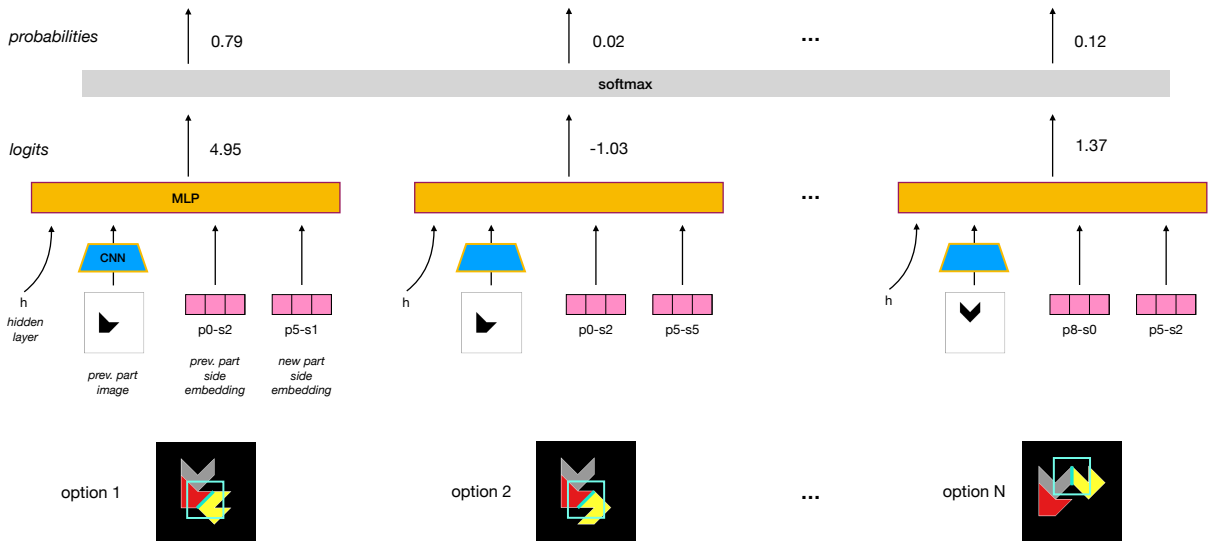


Figure B.1: Relation prediction architecture used in GNS subroutine `GenerateRelation`.

The GNS model uses polygon attachments as a model of relations between parts in an alien figure. Each relation $r_i = \{j, s_j, s_i\}$ encompasses 3 unique choices which together specify an attachment. The first is the choice of attachment part, represented by index j, selected from the set

of all previous parts. Second and third are the choice of polygon side for the attachment part s_j, and for the current part s_i. These choices convey which polygon sides will be touching when the two polygons are connected to one another.

To predict the next relation r_i, GNS uses a neural network as an energy function to score every combination of values $\{j, s_j, s_i\}$ (Fig. B.1). The choice of attachment part j is conveyed by a binary image of the isolated part, which is processed to a hidden embedding by a CNN. The side choices s_j and s_i are each conveyed by a discrete embedding from a learnable dictionary with one entry for every side of every primitive polygon, indexed as $e[c_i, s_i]$. Each of these inputs is concatenated and fed to the neural network, which returns a scalar energy that represents the unnormalized log-probability of choosing this combination.

B.2 Training the GNS model

Our full GNS model and all lesions are training using minibatches of 60 meta-learning episodes. The composition of data distributions for each lesion is provided in Table B.1. The number of support examples in an episode is sampled uniformly between 1-6 at each iteration, and the number of query examples is fixed at 5. Models are trained to maximize the log-likelihood (minimize log-loss) of the query examples conditioned on support. Training proceeds for 40,000 batch iterations using the Adam optimizer with cosine learning rate annealing. For each GNS model, we train 4 different models with different random initialization. In subsequent evaluations, we use the average log-likelihood from all 4 seeds as the overall log-likelihood.

Model	Composition
GNS (P)	60
GNS (P/R)	40/20
GNS (P/R/H)	30/15/15
GNS (P/R/H/C)	20/10/10/20

Table B.1: Minibatch compositions for GNS model training.

B.2.1 Data distribution C

The C distribution is designed to help teach the complete-the-pattern and reconfigure biases, two inductive biases that are relevant in trials with the partial-pattern property. To generate episodes from C, we first sample a trial type from the four partial-pattern types: Rotations-1, Rotations-2, Primitives-1, Primitives-2. Next we sample a support set S by selecting 3 tokens from the trial type that make a partial-pattern. Finally, to construct the query set Q we sample completion items with probability $p_a = 0.59$, reconfigure items with probability $p_b = 0.14$, and alternate "noise" tokens with the remaining probability mass. The values of p_a and p_b are set to mirror the empirical human frequencies for each bias.

B.3 Likelihood analysis

All token likelihoods that we report for the GNS model are marginal image likelihoods. By default, the GNS model computes the likelihood of a *latent program* or a *token string*, i.e. a sequence of parts and relations $\{c_1, r_1, ..., c_N, r_N\}$. There is a many-to-one mapping from these latent programs to images; to obtain the marginal likelihood of a token image, we sum the individual likelihoods from all programs that yield the target image.

For both the GNS model and the Bayesian model, we fit a lapse parameter α that mixes the model distribution $p(y \mid X)$ with a null distribution $q(y)$ to produce a final distribution $\tilde{p}(y \mid X) = (1-\alpha) \cdot p(y \mid X) + \alpha \cdot q(y)$. We use the complexity-based null distribution $q(y) = P^0(y)$ discussed in Zhou et al. (2023, Appendix D).

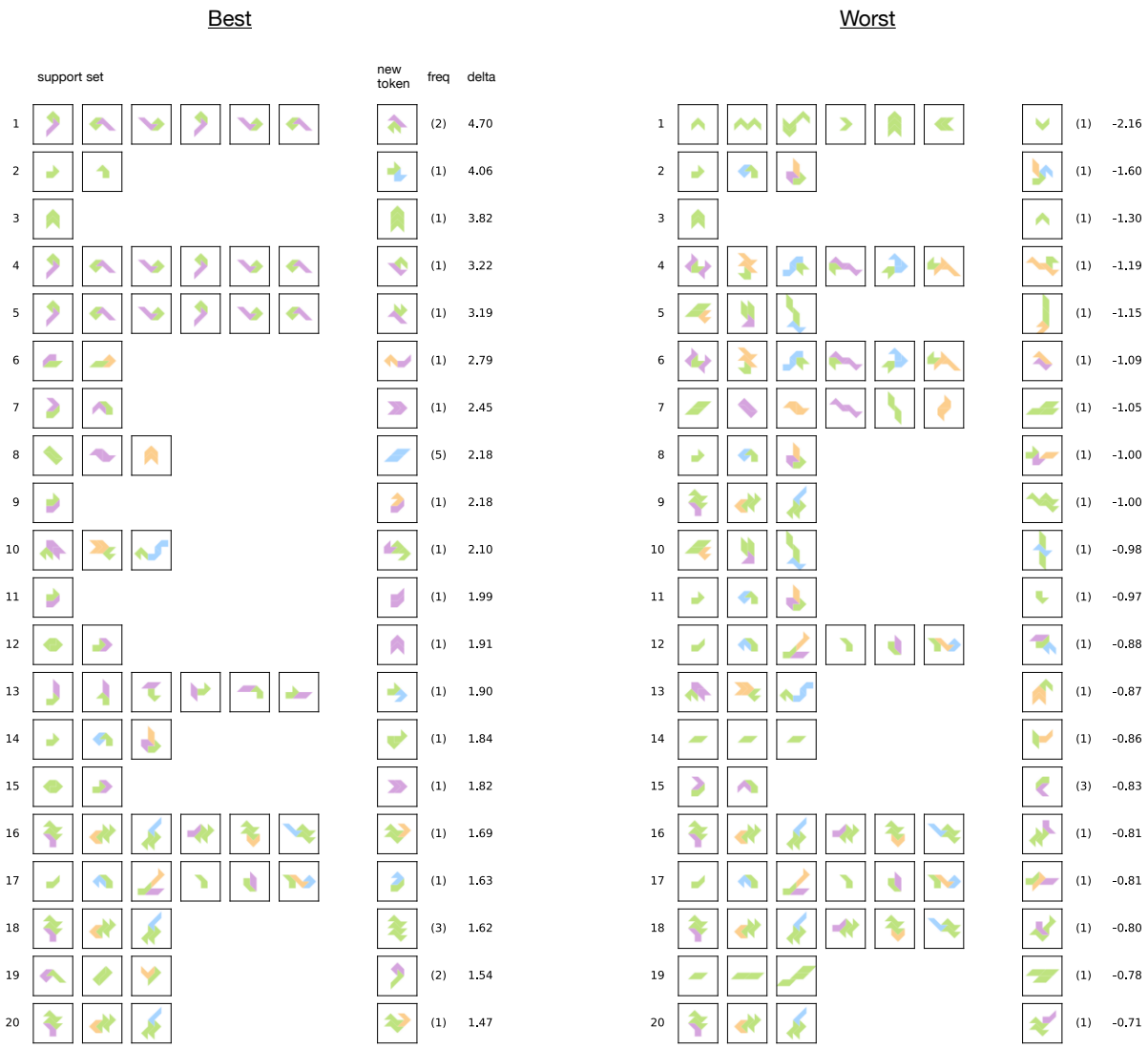


Figure B.2: Best and worst 20 human examples, measured by ℓ(GNS) - ℓ(Bayes).

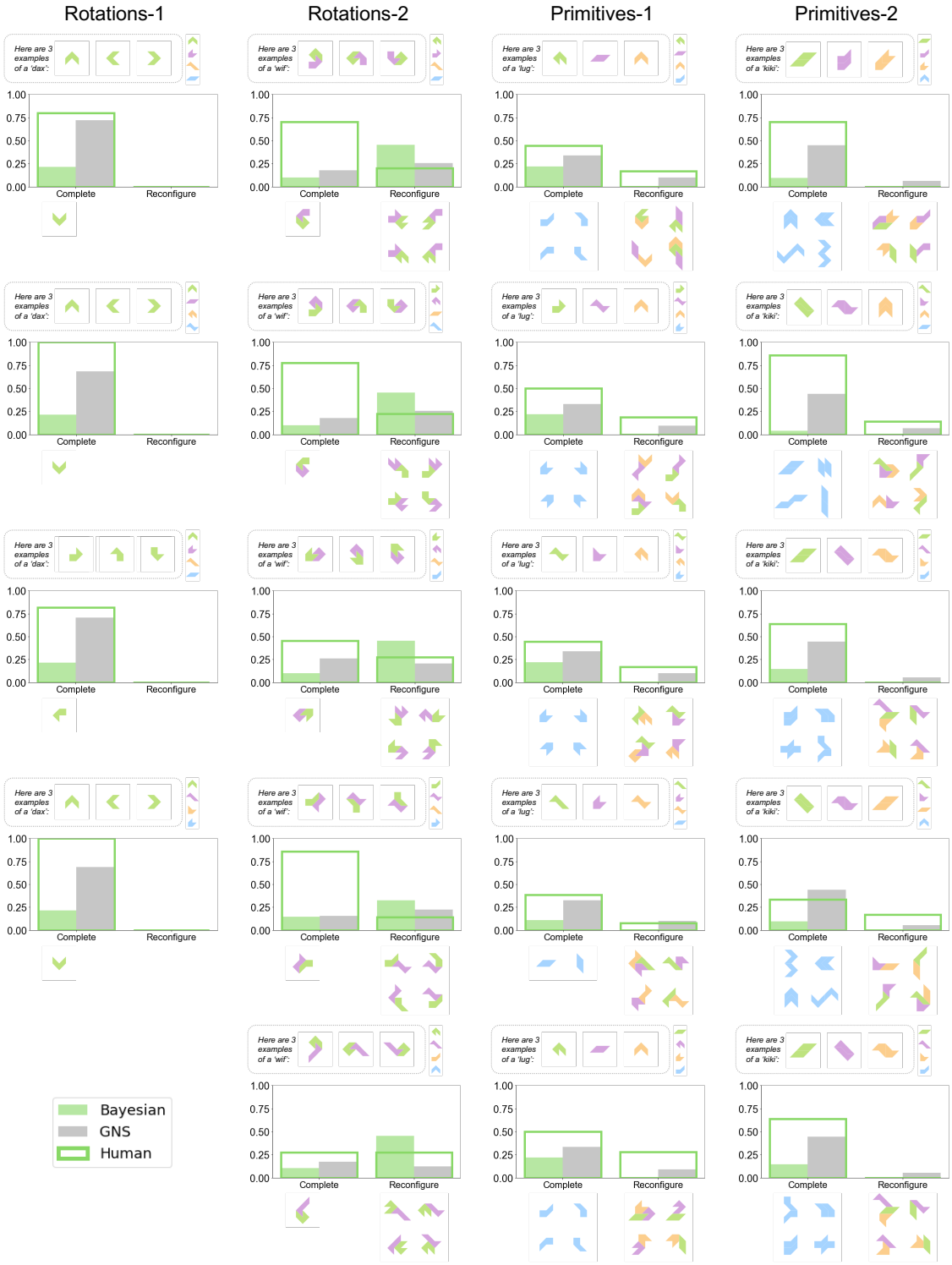


Figure B.3: Inductive biases captured by GNS and Bayesian models (exhaustive version).

Appendix C

Additional details for Chapter 5

C.1 Network architectures

The MLP architecture for Experiment 1 was chosen ad-hoc before running any experiments. The network receives a 60-dimensional input, and thus, we chose a hidden layer size of 30 units to reduce this dimensionality by a factor of 2. L2 regularization was critical to the performance of the network when small training sets were provided. For Experiment 2, we chose the minimal CNN architecture that could effectively learn the image classification task that it was assigned. We generated a large dataset of 30 categories and 20+ examples per category. Then, we started with a large CNN and iteratively reduced the number of parameters until the minimal architecture was found. L2 regularization was again critical to model performance.

C.2 Training parameters

For both the MLP and the CNN, we train the network to minimize negative log-likelihood loss, using stochastic gradient descent (SGD) with the RMSprop update rule and a typical batch size of

32. There are a few exceptions to this batch size: when the training set is very small, we adjust the batch size to ensure there are at least 5 training batches. Thus, for a training set with N categories and K examples per category (a total of $N * K$ training points), we use a batch size of min(32, $\frac{N*K}{5}$). The number of training epochs was chosen such that the network loss reaches an asymptote for each the MLP and CNN. Training loss is monitored and used to save the best model.

Bibliography

Atanov, A., Ashukha, A., Struminsky, K., Vetrov, D., & Welling, M. (2019). The deep weight prior. In *International Conference on Learning Representations (ICLR)*.

Babcock, M. K., & Freyd, J. (1988). Perception of dynamic information in static handwritten forms. *American Journal of Psychology*, *101*(1), 111–130.

Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, *1*(3), 295–311.

Bever, T. G., & Poeppel, D. (2010). Analysis by synthesis: A (re-) emerging program of research for language and vision. *Biolinguistics*, *4*, 174–200.

Bloom, P. (2000). *How children learn the meanings of words*. Cambridge, MA: MIT Press.

Botvinick, M., Barrett, D., Battaglia, P., de Freitas, N., Kumaran, D., & et al. (2017). Building machines that learn and think for themselves. *Behavioral and Brain Sciences*, *40*, e255.

Brodatz, P. (1966). *Textures: a photographic album for artists and designers*. New York, NY: Dover Publications.

Bruna, J., & Mallat, S. (2013). Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *35*(8), 1872–1886.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956). *A study of thinking*. New York: Wiley.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., ... Yu, F. (2015). ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*.

Chung, J., Ahn, S., & Bengio, Y. (2017). Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations (ICLR).*

Colunga, E., & Smith, L. B. (2005). From the lexicon to expectations about kinds: a role for associative learning. *Psychological Review*, *112*(2), 347–382.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science (JASIS)*, *41*(6), 391–407.

Devlin, J., Uesato, J., Bhupatiraju, S., Singh, R., Mohamed, A.-R., & Kohli, P. (2017). Robustfill: Neural program learning under noisy i/o. In *International Conference on Machine Learning (ICML).*

Dewar, K. M., & Xu, F. (2010). Induction, overhypothesis, and the origin of abstract knowledge: evidence from 9-month-old infants. *Psychological Science*, *21*(12), 1871–1877.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning (ICML).*

Dubuisson, M., & Jain, A. K. (1994). A modified hausdorff distance for object matching. In *Proceedings of the 12th International Conference on Pattern Recognition (ICPR)* (pp. 566–568).

Eden, M. (1962). Handwriting and pattern recognition. *IRE Transactions on Information Theory*, *8*, 160–166.

Ellis, K., Ritchie, D., Solar-lezama, A., & Tenenbaum, J. B. (2018). Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems (NeurIPS).*

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., & Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems (NeurIPS).*

Feinman, R., & Lake, B. M. (2018). Learning inductive biases with simple neural networks. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society (CogSci).*

Feinman, R., & Lake, B. M. (2019). Learning a smooth kernel regularizer for convolutional neural networks. In *Proceedings of the 41st Annual Conference of the Cognitive Science Society (CogSci).*

Feinman, R., & Lake, B. M. (2020). Generating new concepts with hybrid neuro-symbolic models. In *Proceedings of the 42nd Annual Conference of the Cognitive Science Society (CogSci).*

Feinman, R., & Lake, B. M. (2021). Learning task-general representations with generative neuro-symbolic modeling. In *International Conference on Learning Representations (ICLR).*

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML).*

Fodor, J. A. (1975). *The language of thought*. Cambridge, MA: Harvard University Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*, 3–71.

Freyd, J. J. (1983). Representing the dynamics of static form. *Memory & Cognition*, *11*(4), 342–346.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, *9*(3), 432–441.

Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S. M. A., & Vinyals, O. (2018). Synthesizing programs for images using reinforced adversarial learning. In *International Conference on Machine Learning (ICML).*

Garcia, V., & Bruna, J. (2018). Few-shot learning with graph neural networks. In *International Conference on Learning Representations (ICLR).*

Geisler, W. S., Perry, J. S., Super, B. J., & Gallogly, D. P. (2001). Edge co-occurence in natural images predicts contour grouping performance. *Vision Research*, *41*(6), 711–724.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., & Rubin, D. (2013). *Bayesian data analysis*. Boca Raton, FL: CRC.

Gelman, S. A. (2003). *The essential child: Origins of essentialism in everyday thought*. Oxford Cognitive Development.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*(1), 1–58.

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., ... et al. (2017). A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science*, *358*(6368).

Gershkoff-Stowe, L., & Smith, L. B. (2004). Shape and the first hundred nouns. *Child Development*, *75*(4), 1098–1114.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV).*

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Goodman, N. D., Tenenbaum, J. B., Feldman, J., & Griffiths, T. L. (2008). A rational analysis of rule-based concept learning. *Cognitive Science*, *32*(1), 108–154.

Goodman, N. D., Tenenbaum, J. B., & Gerstenberg, T. (2015). Concepts in a probabilistic language of thought. In *The Conceptual Mind: New Directions in the Study of Concepts* (pp. 623–653). Cambridge, MA: MIT Press.

Goodman, N. D., Ullman, T. D., & Tenenbaum, J. B. (2011). Learning a theory of causality. *Psychological Review*, *118*(1), 110–119.

Grant, E., Finn, C., Levine, S., Darrell, T., & Griffiths, T. (2018). Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations (ICLR).*

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., ... Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning (ICML).*

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. In *International Conference on Machine Learning (ICML).*

Ha, D., & Eck, D. (2018). A neural representation of sketch drawings. In *International Conference on Learning Representations (ICLR).*

Halle, M., & Stevens, K. (1962). Speech recognition: A model and a program for research. *IRE Transactions on Information Theory*, *8*(2), 155–159.

Harlow, H. F. (1949). The formation of learning sets. *Psychological Review*, *56*(1), 51–65.

Hewitt, L. B., Le, T. A., & Tenenbaum, J. B. (2020). Learning to learn generative programs with Memoised Wake-Sleep. In *Uncertainty in Artificial Intelligence (UAI).*

Hewitt, L. B., Nye, M. I., Gane, A., Jaakkola, T., & Tenenbaum, J. B. (2018). The Variational Homoencoder: Learning to learn high capacity generative models from few examples. In *Uncertainty in Artificial Intelligence (UAI).*

Hill, F., Hermann, K. M., Blunsom, P., & Clark, S. (2017). Grounded language learning in a 3d simulated world. *arXiv preprint arXiv:1710.09867*.

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2022). Meta Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154.

James, K. H. (2010). Sensori-motor experience leads to changes in visual processing in the developing brain. *Developmental Science*, *13*(2), 279–288.

James, K. H., & Atwood, T. P. (2009). The role of sensorimotor learning in the perception of letter-like forms: Tracking the causes of neural specialization for letters. *Cognitive Neuropsychology*, *26*(1), 91–110.

James, K. H., & Gauthier, I. (2006). Letter processing automatically recruits a sensory–motor brain network. *Neuropsychologia*, *44*, 2937–2949.

James, K. H., & Gauthier, I. (2009). When writing impairs reading: Letter perception's susceptibility to motor interference. *Journal of Experimental Psychology: General*, *138*(3), 416–31.

Jones, J. P., & Palmer, L. A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, *58*(6), 1233–1258.

Jongejan, J., Rowley, H., Kawashima, T., Kim, J., & Fox-Gieg, N. (2016). *The quick, draw!-a.i. experiment.* https://quickdraw.withgoogle.com/.

Keil, F. C. (1989). *Concepts, kinds, and cognitive development*. Cambridge, MA: MIT Press.

Kemp, C., Perfors, A., & Tenenbaum, J. B. (2007). Learning overhypotheses with hierarchical bayesian models. *Developmental Science*, *10*(3), 307–321.

Kemp, C., & Tenenbaum, J. B. (2008). The discovery of structural form. *Proceedings of the National Academy of Sciences (PNAS)*, *105*(31), 10687–10692.

Kemp, C., & Tenenbaum, J. B. (2009). Structured statistical models of inductive reasoning. *Psychological Review*, *116*(1), 20–58.

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

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS).*

Kruschke, J. K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review*, *99*(1), 22–44.

Kulkarni, T. D., & et al. (2015). Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Lake, B. M., & Baroni, M. (2018). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML).*

Lake, B. M., & Piantadosi, S. T. (2020). People infer recursive visual concepts from just a few examples. *Computational Brain & Behavior*, *3*(1), 54–65.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2019). The Omniglot challenge: A 3-year progress report. *Current Opinion in Behavioral Sciences*, *29*, 97–104.

Lake, B. M., & Tenenbaum, J. B. (2010). Discovering structure by learning sparse graphs. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society (CogSci).*

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, E253.

Landau, B., Smith, L. B., & Jones, S. S. (1988). The importance of shape in early lexical learning. *Cognitive Development*, *3*(3), 299–321.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Li, S. Z. (2009). *Markov random field modeling in image analysis*. New York, NY: Springer-Verlag.

Ling, W., Trancoso, I., Dyer, C., & Black, A. (2016). Character-based neural machine translation. In *International Conference on Learning Representations (ICLR).*

Longcamp, M., Anton, J. L., Roth, M., & Velay, J. L. (2003a). Visual presentation of single letters activates a premotor area involved in writing. *Neuroimage*, *19*(4), 1492–1500.

Longcamp, M., Anton, J.-L., Roth, M., & Velay, J.-L. (2003b). Visual presentation of single letters activates a premotor area involved in writing. *NeuroImage*, *19*(4), 1492–1500.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations (ICLR).*

Marcus, G. F. (2003). *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. Cambridge, MA: MIT Press.

Markman, E. M., & Wachtel, G. F. (1988). Children's use of mutual exclusivity to constrain the meaning of words. *Cognitive Psychology*, *20*(2), 121–157.

McClelland, J. L. (2010). Emergence in Cognitive Science. *Topics in Cognitive Science*, *2*(4), 751–770.

McClelland, J. L., Botvinick, M. M., Noelle, D. C., Plaut, D. C., Rogers, T. T., Seidenberg, M. S., & Smith, L. B. (2010). Letting structure emerge: Connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Science*, *14*, 348–356.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (2013). *Machine learning: an artificial intelligence approach*. Berlin: Springer Science+Business Media.

Mo, K., Zhu, S., Chang, A. X., Yi, L., Tripathi, S., Guibas, L. J., & Su, H. (2019). PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Murphy, G. L. (2002). *The Big Book of Concepts*. Cambridge, MA: MIT Press.

Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, *92*(3), 289–316.

Neisser, U. (1966). *Cognitive Psychology*. Appleton-Century-Crofts.

Nye, M. I., Solar-Lezama, A., Tenenbaum, J. B., & Lake, B. M. (2020). Learning compositional rules via neural program synthesis. *arXiv preprint arXiv:2003.05562*.

Perfors, A., Tenenbaum, J. B., & Regier, T. (2011). The learnability of abstract syntactic principles. *Cognition*, *118*(3), 306–338.

Piantadosi, S. T., Tenenbaum, J. B., & Goodman, N. D. (2016). The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, *123*(4).

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Reed, S., & de Freitas, N. (2016). Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR).*

Rehder, B. (2003). A causal-model theory of conceptual representation and categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *29*(6), 1141–1159.

Rehder, B., & Hastie, R. (2001). Causal knowledge and categories: The effects of causal beliefs on categorization, induction, and similarity. *Journal of Experimental Psychology: General*, *130*(3), 323–360.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., & Wierstra, D. (2016). One-Shot generalization in deep generative models. In *International Conference on Machine Learning (ICML).*

Ritter, S., Barrett, D. G. T., Santoro, A., & Botvinick, M. M. (2017). Cognitive psychology for deep neural networks: a shape bias case study. In *International Conference on Machine Learning (ICML)*.

Rogers, T. T., & McClelland, J. L. (2004). *Semantic Cognition: A Parallel Distributed Processing Approach*. MIT Press.

Rozenblit, L., & Keil, F. (2002). The misunderstood limits of folk science: an illusion of explanatory depth. *Cognitive Science*, *26*, 521–562.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. (1987). *Parallel distributed processing: Explorations in the microstructure of cognition. volume 1*. Cambridge, MA: MIT Press.

Salakhutdinov, R., Tenenbaum, J., & Torralba, A. (2012). One-shot learning with a hierarchical nonparametric bayesian model. *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 195–206.

Salakhutdinov, R., Tenenbaum, J. B., & Torralba, A. (2013). Learning with hierarchical-deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1958–1971.

Samuelson, L. K., & Smith, L. B. (1999). Early noun vocabularies: do ontology, category structure and syntax correspond? *Cognition*, *73*(1), 1–33.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *29*(3), 411–426.

Shyam, P., Gupta, S., & Dukkipati, A. (2017). Attentive recurrent comparators. In *International Conference on Machine Learning (ICML)*.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR).*

Smith, L. B., Jones, S. S., Landau, B., Gershkoff-Stowe, L., & Samuelson, L. (2002). Object name learning provides on-the-job training for attention. *Psychological Science*, *13*(1), 13–19.

Smolensky, P. (1987). Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1* (pp. 194–281). Cambridge, MA: MIT Press.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS).*

Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Simonsen, J. G., & Nie, J. Y. (2015). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM).*

Stuhlmuller, A., Tenenbaum, J. B., & Goodman, N. D. (2010). Learning Structured Generative Concepts. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society (CogSci).*

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS).*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., & Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, *331*(6022), 1279–1285.

Tse, P. U., & Cavanagh, P. (2000). Chinese and Americans see opposite apparent motions in a Chinese character. *Cognition*, *74*(3), B27–B32.

Valkov, L., Chaudhari, D., Srivastava, A., Sutton, C., & Chaudhuri, S. (2018). HOUDINI: Lifelong learning as program synthesis. In *Advances in Neural Information Processing Systems (NeurIPS).*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS).*

Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NeurIPS).*

Ward, T. B. (1994). Structured imagination: The role of category structure in exemplar generation. *Cognitive Psychology*, *27*, 1–40.

Xu, F., & Tenenbaum, J. B. (2007). Word learning as Bayesian inference. *Psychological Review*, *114*(2), 245–272.

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., ... Bengio, Y. (2016). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML).*

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., & Tenenbaum, J. B. (2018). Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems (NeurIPS).*

Yuille, A. L., & Liu, C. (2019). Deep nets: What have they ever done for vision? *International Journal of Computer Vision*, *129*, 781–802.

Zhou, Y., Feinman, R., & Lake, B. M. (2023). Compositional diversity in visual concept learning. *arXiv preprint arXiv:2305.19374*.